Notes on Social and Behavioral Networks

Cristian Di Pietrantonio

October 17, 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 $n = |V|$, $\Delta = \max_v \deg(\mathrm{v})$ 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.

ξ_1 = "A given edge $\{u, v\} \in E$ is inferred using the trace t."

The probability of ξ_1 is given by

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

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

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

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 exclude 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 $X = |E - L|$.

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

□

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

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

□

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 (Discrete derivative)**.** *Given a set function $f : 2^V \to \mathbb{R}$, $A \subseteq V$, $x \in V$, we define the* discrete derivative *of f at A with respect to x as*

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

In economics, Equation 3.16 is called *marginal return.* Putting together equations 3.15 and 3.16 we get the economics Law of diminishing returns.

Definition 3 (Marginal returns property)**.** *Given $B \subseteq A \subseteq V$ and an item $x \in V \setminus B$, adding x to B increase the value of that set more than adding that element to a bigger set A which includes B. Formally*

$$\Delta(x|B) \geq \Delta(x|A). \tag{3.17}$$

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

Claim 1. *Definition 3 implies Definition 1*

Proof. Let $S, T \subseteq V$ be given. There are two cases, let's look at the simple one first. Assume $S \subseteq T$, then $S \cup T = T$ and $S \cap T = S$. The same conditions of Definition 1 are present, so we can just go backwards on the steps we performed to get to Definition 3.

So assume $S \not\subseteq T$ and $T \not\subseteq S$. This means that $S - T = \{s_1, \ldots, s_k\}$ is not empty.

Lemma 5.

$$\forall 1 \leq i \leq k \;\; f((S \cap T) \cup \{s_1, \ldots, s_i\}) - f(S \cap T) \geq f(T \cup \{s_1, \ldots, s_i\}) - f(T) \tag{3.18}$$

Proof. If $i = 1$ then we get the marginal returns property. The following inequality shows the base step of the induction.

$$f((S \cap T) \cup \{s_1\}) - f(S \cap T) \geq f(T \cup \{s_1\}) - f(T) \tag{3.19}$$

This lemma is all about applying marginal returns property to longer and longer stripes of elements.

What we want to do in the next step is adding s_2. So we have

$$f((S \cap T) \cup \{s_1, s_2\}) - f((S \cap T) \cup \{s_1\}) \geq f(T \cup \{s_1, \dots, s_2\}) - f(T \cup \{s_1\}) \quad (3.20)$$

This is true by the marginal returns property. In fact, all this inequalities will be true up until

$$f((S \cap T) \cup \{s_1, \dots, s_{i+1}\}) - f((S \cap T) \cup \{s_1, \dots, s_i\}) \geq \\ f(T \cup \{s_1, \dots, s_{i+1}\}) - f(T \cup \{s_1, \dots, s_i\}) \quad (3.21)$$

We are assuming that these inequalities hold by induction. Now we want to add up them, and we see that the terms cancel out. We are left with

$$f((S \cap T) \cup \{s_1, \dots, s_{i+1}\}) - f(S \cap T) \geq f(T \cup \{s_1, \dots, s_{i+1}\}) - f(T) \quad (3.22)$$

And so the claim is true. □

Now observe what happens at the last step.

$$\begin{aligned} &f((S \cap T) \cup \{s_1, \dots, s_k\}) - f(S \cap T) \geq f(T \cup \{s_1, \dots, s_k\}) - f(T) \\ &f((S \cap T) \cup (S - T)) - f(S \cap T) \geq f(T \cup (S - T)) - f(T) \\ &f(S) - f(S \cap T) \geq f(T \cup S) - f(T) \\ &f(S) + f(T) \geq f(S \cap T) + f(S \cup T) \end{aligned}$$

And we have proved the equivalence between the two definitions. □

Claim 2. *If f and g are submodular, then $f + g$ is submodular.*

Proof. $\forall S, T \subseteq V$

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$$

and

$$g(S) + g(T) \geq g(S \cup T) + g(S \cap T)$$

Define $h(S) = f(S) + g(S)$ and then sum the inequalities. The result is still submodular (the inequality holds). □

If f is submodular, $-f$ is submodular? Only when the property holds at equality (hence, we don't switch sign).

Claim 3. *If f is submodular, αf is submodular, $\forall \alpha \geq 0$.*

Claim 4. *If f is submodular and g is modular then $f - \alpha g$ is submodular.*

Proof. g is modular, so multiplying it for a negative number results in a modular function, which is also submodular. Finally, the sum of two submodular function is submodular. □

3.3 Facility location

Now we are going to give an example of submodular function optimization problem. Let $\mathcal{F} = \{F_1, \dots, F_n\}$ be a set of facilities and $\mathcal{U} = \{U_1, \dots, U_m\}$ a set of users. Maybe $\mathcal{F}$ is the set of universities in Rome and $\mathcal{U}$ the set of (wannabe) students in Rome. Student U_i gets a gain $M_{ij} \geq 0$ for starting a program at University F_j. So what a student would do? He would go to the facility that gives the maximum gain. If thats the protocol, it is all very easy. But suppose you are the government and you can't open all the n universities, but only k of them. We have to select $S = \{i_1, \dots, i_k : U_{i_k} \in \mathcal{U}\}$ facilities so to maximize $\sum_{i=1}^{m} \max_{j \in S} M_{ij}$.

What is the function we are going to prove to be submodular?

$$f(S) = \sum_{i=1}^{m} \max_{j \in S} M_{ij} \tag{3.23}$$

The function is pretty intricate, but we are going to use the properties we just saw to prove it being submodular. Thanks to Claim 2, we can focus only on proving

$$f_i(S) = \max_{j \in S} M_{ij} \tag{3.24}$$

to be submodular.

Claim 5. $\forall S, T \subseteq \mathcal{F}$

$$f_i(S) + f_i(T) \geq f_i(S \cup T) + f_i(S \cap T). \tag{3.25}$$

Proof. Let's concentrate on $f_i(S \cup T)$. Observe that, $f_i(S \cup T)$ is going to be larger or at least as large as the other f_is in the inequality 3.25 because f_i is just a max over the elements of the set. Suppose the maximum value of $f_i(S \cup T)$ is realized by some $j^\star$. This element can be in S, T, or both.

$$j^\star \in S \implies f_i(S) = f_i(S \cup T) \tag{3.26}$$

$$j^\star \in T \implies f_i(T) = f_i(S \cup T) \tag{3.27}$$

We are trying to prove that the left hand side is no larger than the left hand side. We have seen that $f_i(S \cup T)$ is the biggest part in the right hand side, and we have shown that, no matter what, there will be one element in the LHS that has the same value. Now we have to show that the other part of the RHS is not larger than the other element in the LHS.

Suppose $k^\star$ achieves the maximum at $f_i(S \cap T)$. What do we know? $k^\star$ is going to be in both S and T. So the max at S is going to be at least as large as the max obtained with $k^\star$. It follows that

$$f_i(S) \geq f_i(S \cap T) \tag{3.28}$$

and

$$f_i(T) \geq f_i(S \cap T) \tag{3.29}$$

so

$$\min(f_i(T), f_i(S)) \geq f_i(S \cap T). \tag{3.30}$$

□

At this point we can state that K-MAX-COVER is FACILITY LOCATION with $M_{ij} \in \{0,1\}$. Think of users as elements $E_1, \ldots, E_m$ and facilities as sets $S_1, \ldots, S_n$.

$$M_{ij} = \begin{cases} 1 \ \textit{if}\ E_i \in S_j \\ 0 \ \textit{otherwise.} \end{cases} \tag{3.31}$$

Here, $f_i(S)$ is going to be the maximum across all the sets in the solution S and is going to be 1 if at least one set in S contains element i; zero otherwise. Finally, we are summing up over all the elements we picked.

3.4 Approximation of submodular functions

We are going to see an algorithm that returns a $\left(1 - \frac{1}{e}\right)$ approximation of the optimal solution for a submodular function under a cardinality constraint. It is going to be similar to the one seen for k-max cover.

```
Input: $f, k$
$S_0 \leftarrow \emptyset$;
for $j = 1, \ldots, k$ do
    Let $v_i \in ([n] - S_{i-1})$ be the element maximizing $\Delta(v_i | S_{i-1})$;
    $S_i \leftarrow S_{i-1} \cup \{v_i\}$;
end
return $S_k$;
```

Algorithm 4: Greedy algorithm to approximate submodular functions.

Authors in [4] proved the following theorem.

Theorem 4. *If f is submodular, non-negative and monotone then Algorithm 4 returns a $\left(1 - \frac{1}{e}\right)$ approximation of* $\max_{S \in \binom{[n]}{k}} f(S)$.

Proof. Let $S^\star = \{v_1^\star, \ldots, v_k^\star\} \in \binom{[n]}{k}$ be the optimal solution.

Lemma 6.

$$\forall i \leq k-1 \ f(S^\star) \leq f(S_i) + k \overbrace{(f(S_{i+1}) - f(S_i))}^{\textit{new step gain}} \tag{3.32}$$

Proof. We look at the gain obtained by adding a new element at step i. In the following lines we are *telescoping* the first line to get intermediate steps.

$$\begin{aligned} f(S^\star) &\le f(S^\star \cup S_i) && \text{(monotonicity)} \\ &= f(S^\star \cup S_i) - f(\{v_1^\star, \dots, v_{k-1}^\star\} \cup S_i) \\ &+ f(\{v_1^\star, \dots, v_{k-1}^\star\} \cup S_i) - f(\{v_1^\star, \dots, v_{k-2}^\star\} \cup S_i) \\ &+ f(\{v_1^\star, \dots, v_{k-2}^\star\} \cup S_i) - \dots \\ &\dots \\ &+ f(\{v_1^\star, v_2^\star\} \cup S_i) - f(\{v_1^\star\} \cup S_i) \\ &+ f(\{v_1^\star\} \cup S_i) - f(S_i) + f(S_i) \end{aligned}$$

On each row we can see a diminishing return:

$$\begin{aligned} &\Delta(v_k^\star | S_i \cup \{v_1^\star, \dots, v_{k-1}^\star\}) \\ &+ \Delta(v_{k-1}^\star | S_i \cup \{v_1^\star, \dots, v_{k-2}^\star\}) \\ &+ \dots \\ &+ \Delta(v_2^\star | S_i \cup \{v_1^\star\}) \\ &+ \Delta(v_1^\star | S_i) \\ &+ f(S_i) \end{aligned}$$

So we have

$$\begin{aligned} f(S^\star) &\le f(S^\star \cup S_i) && \text{(monotonicity)} \\ &= f(S_i) + \sum_{j=1}^{k} \Delta(v_j^\star | S_i \cup \{v_1^\star, \dots, v_{j-1}^\star\}) \\ &\le f(S_i) + \sum_{j=1}^{k} \Delta(v_j^\star | S_i) && \text{(diminishing returns)} \\ &\le f(S_i) + \sum_{j=1}^{k} \Delta(v_{i+1} | S_i) && \text{(greedy choice)} \\ &= f(S_i) + k\Delta(v_{i+1} | S_i) \\ &= f(S_i) + k(f(S_{i+1}) - f(S_i)) \end{aligned}$$

□

Define $\delta_i = f(S^\star) - f(S_i)$ and also, for simplicity, $\delta_{i+1} = f(S^\star) - f(S_{i+1})$. Remember that we want to claim we don't lose so much in going from one step to the next. By looking at $f(S_{i+1}) - f(S_i)$ we can't say much since we don't know the function f. However, let's look at the difference between δ_i and δ_{i+1}; essentially we used the diminishing return property to forget about the function. Now we can just think about the costs with respect to the optimum that we pay at each step.

$$\begin{aligned}
\delta_i &= f(S^\star) - f(S_i) && (3.33)\\
&\leq k(f(S_{i+1}) - f(S_i)) && \text{(by Lemma 6)}\\
&= k(\delta_i - \delta_{i+1}) && (3.34)
\end{aligned}$$

Now we want to show that the cost goes down.

$$k\delta_{i+1} \leq (k-1)\delta_i \implies \delta_{i+1} \leq \left(1 - \frac{1}{k}\right)\delta_i \tag{3.35}$$

So

$$\begin{aligned}
\delta_0 &= f(S^\star) - f(S_0) \leq f(S^\star) && \text{(f non-negative)}\\
\delta_1 &\leq \left(1 - \frac{1}{k}\right)\delta_0 = \left(1 - \frac{1}{k}\right) f(S^\star) && (3.36)\\
\dots & && (3.37)\\
\delta_k &\leq \left(1 - \frac{1}{k}\right)\delta_{k-1} \leq, \dots, \leq \left(1 - \frac{1}{k}\right)^k \delta_0 && (3.38)\\
&= \left(1 - \frac{1}{k}\right)^k f(S^\star) \leq \frac{1}{e} f(S^\star). && (3.39)
\end{aligned}$$

□

This can be though as an example of how one can start by solving a specific problem, creating a solution, and then think of it as a general solution. One can asks what the created algorithm, or the analysis for proving it works, can say about a more general problem.

Do we need the properties we assumed, i.e. submodularity, non-negativity and monotonicity?

So, non-negativity is not that important. For one thing, if the function can be negative, what does it mean to optimize it and to give a constant approximation? Maybe the optimal is negative, in which case, say, a $\left(1 - \frac{1}{e}\right)$ approximation would be impossible. In the context of maximization, multiplying the optimal (which we assume negative) for a constant $0 < x < 1$ gives a greater value than the optimal (contradiction). Thus, multiplying by a constant can change whether you are maximizing or minimizing. For what we have just said, non-negativity is necessary. On the other hand, you could remove that assumption keeping, in this particular case, a bound with $f(S^\star) - f(S_0)$. It won't yield a constant approximation, because it doesn't make sense with negative functions, but you would get something that depends on this gap. Essentially, non-negativity is used to get a nice, clean theorem.

What about monotonicity? That actually helps. If it removed, then what we can get is a $\frac{1}{2}$ approximation, under some computational assumptions, but not a $\left(1-\frac{1}{e}\right)$ approximation. It would require a different way of reasoning about the problem and a much more complicated algorithm. So removing it means getting a worst approximation. Finally, we don't want to remove submodularity, we started from there.

Other questions are about constraints. Why do we care about optimizing the value of the function at sets of cardinality k? One could say "I care about other sets". If the function is monotone and we want to maximize it without any constraint on the set, what should I do? Return the full set. Monotonicity is actually very important if the optimization problem comes with some constraint. As example, what if we do not want exactly k elements but at most k elements. In this case not having monotonicity is an important issue (maybe the empty set is the one that has more value).

3.5 Learning a submodular function

Algorithm 4 is able to approximate the value of the function by looking at $O(kn)$ values instead of $\Theta(2^n)$. It performs a number of steps that is logarithmic in the size of the input. In cases when there is no need to look at all the input, the algorithm is said to be sublinear.

We saw that linear (modular) functions can be understood by only looking at singleton sets (plus the empty set), that is, $n+1$ values. It is natural to think that we can't do the same for submodular functions, and what follows is the idea to prove it.

What is a construction that can actually prove that, while there is an algorithm that efficiently optimize the function, there is no algorithm that can learn the function efficiently? As usual we want to use easy building blocks, and then mix up them according to the claims we saw earlier (summing them up, essentially).

For $T \subseteq [n]$, define the function

$$f_T(S) = \begin{cases} 0 \ \textit{if } S \subseteq T, \\ 1 \ \textit{otherwise.} \end{cases} \tag{3.40}$$

The function f_T is a building block that we want to use to build the final submodular function F. Consider

$$\mathcal{S} \subseteq U = \binom{[n]}{\frac{n}{2}} = \left\{ Q : Q \subseteq [n] \wedge |Q| = \frac{n}{2} \right\}$$

F will depend on $\mathcal{S}$, but the fact we don't know actually what $\mathcal{S}$ is will lead us to the result we want to prove. So, let's define

$$F_{\mathcal{S}}(A) = \sum_{T \in \mathcal{S}} f_T(A). \tag{3.41}$$

There are exponentially many functions f_T, which are all the possible building blocks. We are looking for a subset of them, and one subset is be picked without it being disclosed; now, the value of the submodular function F is the sum of the values of the functions at a particular set.

Claim 6. *f_T is submodular.*

Proof. Let's pick two sets $A, B \subseteq [n]$, and apply the definition of submodularity.

$$f_T(A) + f_T(B) \geq f_T(A \cup B) + f_T(A \cap B). \tag{3.42}$$

There are four cases to be considered.

1. $A, B \subseteq T$. Then we have $0 + 0 \geq 0 + 0$, which holds.
2. $A \subseteq T$ and $B \not\subseteq T$ (and the specular case). Computing the function values we have that $0 + 1 \geq 0 + 1$.
3. $A, B \not\subseteq T$. In this case we have $1 + 1 \geq 0 + x$. We can't tell exactly the value x of the function computed at intersection, but it can be at most 1, so the inequality holds.

□

Since the sum of submodular functions is submodular, $F_{\mathcal{S}}(A)$ is submodular. We want to prove that knowing F requires a lot of queries. Doing it formally would require many definitions, but it is a pretty easy algorithm. Your algorithm's call is to learn F by querying it. The answer it gets at each query is some integer representing the number of functions f_T that evaluates to 1 at set A.

How big and how small that number can be? The function F is nonnegative, so the minimum is 0 and the maximum is the cardinality of S, which can be as large as 2^n. $F_S(A)$ is an integer from 0 to 2^n, and thus requires n bits for each query. We will show the that the number of bits needed to represent F is much larger, exponential. Why is it the case? How big is the set of sets U?

$$|U| = \binom{n}{\frac{n}{2}} = \Theta\left(\frac{2^n}{\sqrt{n}}\right). \tag{3.43}$$

Since we are picking any subset $\mathcal{S}$ of this big set of sets, to be able to know which subset $\mathcal{S}$ is, we actually have to know for each set in U whether i picked it or not. One bit is needed for each set, summing up to $|U|$ bits. At each query we get only n bits, therefore at least $\frac{|U|}{n}$ queries are needed (i.e., an exponential number).

Why do we need to know exactly S? Well, if we don't know S, the value of F can't be known.

This is why learning submodular functions is hard. But we can optimize them with efficient algorithms, doing leverage on certain properties as we saw.

Chapter 4

Random Graph Models

We are going to start random graph models that would represent properties of social networks. The one model we are going to look at is the preferential attachment model, a very famous one. It was introduced so to explain one strange behavior of social network distributions.

4.1 Preferential Attachment Model

It has been observed that the degree distribution in big social graphs, like the one of Facebook or Twitter, it is a really close to a *power law* distribution.

Definition 4. *A probability distribution P follows a power law if*

$$P(x) = x^{-\alpha}, \alpha > 1. \tag{4.1}$$

In the social network context it means that it is exponentially more likely to pick "normal people" with few friends or followers rather than popular profiles, called "celebrities" or "authorities".

It turns out that the Preferential Attachment Model is able to capture this Social Networks' aspect, whereas others random models like $G(n,p)$ (the one proposed by Edgar Gilbert) do not show this regularity. In $G(n,p)$, for example, we expect each node to have the degree very close to the mean, because of the Chernoff Bound.

Nodes with high degree in a social network are few, but they exists. So, as an example, an advertisement agency could pay those celebrities to publicize a product, enabling a spread of information due to the high number of connections those nodes have.

$$X_i^{(t)} = \text{“\textit{Number of nodes with degree } } i \text{ \textit{in} } G_i\text{”}. \tag{4.2}$$

The following properties are always true:

$$X_2^{(1)} = 1,\ X_i^1 = 0\ \forall i\ i \neq 2. \tag{4.3}$$

$$|E(G_t)| = t,\ |V(G_t)| = t \tag{4.4}$$

Furthermore. the sum of all degrees in the graph is twice the number of edges.

Can we compute the expected number of nodes with degree 1? That number depends on the past, which we can represent as $\overline{X^{(t-1)}}$, the vector representing the degree distribution at $t-1$ time step.

Define

$$\xi_1 = \textit{“The new node chooses a degree } 1 \textit{ neighbor”.} \tag{4.5}$$

Then

$$\begin{aligned}
E[X_1^{(t)}|\overline{X_1^{(t-1)}}] &= (X_1^{(t-1)}+1)(1-Pr\{\xi_1\}) + X_1^{(t-1)}Pr\{\xi_1\} \qquad (4.6)\\
&= (X_1^{(t-1)}+1)\left(1-\frac{X_1^{(t-1)}}{2(t-1)}\right) + X_1^{(t-1)}\frac{X_1^{(t-1)}}{2(t-1)} \qquad (4.7)\\
&= X_1^{(t-1)} - \frac{(X_1^{(t-1)})^2}{2(t-1)} + 1 - \frac{X_1^{(t-1)}}{2(t-1)} + \frac{(X_1^{(t-1)})^2}{2(t-1)} \qquad (4.8)\\
&= \left(1-\frac{1}{2(t-1)}\right)X_1^{(t-1)} + 1. \qquad (4.9)
\end{aligned}$$

So now

$$\begin{aligned}
E[X_1^{(t)}] &= \sum_{c\geq 0} Pr\left\{X_1^{(t-1)}=c\right\} E[X_1^{(t)}|X_1^{(t-1)}=c]\\
&= \sum_{c\geq 0} Pr\left\{X_1^{(t-1)}=c\right\}\left(\left(1-\frac{1}{2(t-1)}\right)c+1\right)\\
&= \sum_{c\geq 0} Pr\left\{X_1^{(t-1)}=c\right\}\left(1-\frac{1}{2(t-1)}\right)c + \sum_{c\geq 0} Pr\left\{X_1^{(t-1)}=c\right\}\\
&= \left(1-\frac{1}{2(t-1)}\right)\sum_{c\geq 0} Pr\left\{X_1^{(t-1)}=c\right\}c+1\\
&= \left(1-\frac{1}{2(t-1)}\right)E[X_1^{(t-1)}]+1.
\end{aligned}$$

Lemma 7.

$$E[X_1^{(t)}] = \left(1-\frac{1}{2(t-1)}\right)E[X_1^{(t-1)}]+1. \tag{4.10}$$

We want to derive a similar lemma for $i>1$.

$$
\begin{aligned}
E[X_i^{(t)}|\overline{X^{(t-1)}}] &= X_i^{(t-1)} + Pr\left\{X_i^{(t)} - X_i^{(t-1)} = 1\right\} - Pr\left\{X_i^{(t)} - X_i^{(t-1)} = -1\right\} \\
&= X_i^{(t-1)} + \frac{i-1}{2(t-1)} X_{i-1}^{(t-1)} - \frac{i}{2(t-1)} X_i^{(t-1)} \\
&= \left(1 - \frac{i}{2(t-1)}\right) X_i^{(t-1)} + \frac{i-1}{2(t-1)} X_{i-1}^{(t-1)} \\
&= E[X_i^{(t)}|X_i^{(t-1)}, X_{i-1}^{(t-1)}].
\end{aligned}
$$

So

$$
\begin{aligned}
E[X_i^{(t)}] &= \sum_{c,d\geq 0} Pr\left\{X_i^{(t-1)} = c \wedge X_{i-1}^{(t-1)} = d\right\} E[X_i^{(t)}|X_i^{(t-1)} = c, X_{i-1}^{(t-1)} = d] \\
&= \sum_{c,d\geq 0} Pr\left\{X_i^{(t-1)} = c \wedge X_{i-1}^{(t-1)} = d\right\} \left(\left(1 - \frac{i}{2(t-1)}\right) c + \frac{i-1}{2(t-1)} d\right) \\
&= \sum_{c\geq 0} Pr\left\{X_i^{(t-1)} = c\right\} \left(1 - \frac{i}{2(t-1)}\right) c + \sum_{d\geq 0} Pr\left\{X_{i-1}^{(t-1)} = d\right\} \frac{i-1}{2(t-1)} d \\
&= \left(1 - \frac{i}{2(t-1)}\right) E[X_i^{(t-1)}] + \frac{i-1}{2(t-1)} E[X_{i-1}^{(t-1)}].
\end{aligned}
$$

Lemma 8.

$$
\forall i \ i \geq 2, \ E[X_i^{(t)}] = \left(1 - \frac{i}{2(t-1)}\right) E[X_i^{(t-1)}] + \frac{i-1}{2(t-1)} E[X_{i-1}^{(t-1)}]. \tag{4.11}
$$

To solve this is difficult, so we give the theorem and prove that it is correct, without deriving it.

Theorem 5. *Let* $E_i = \frac{4}{i(i+1)(i+2)}$.

$$
E_i t - 1 \leq E[X_i^{(t)}] \leq E_i t + 1 \tag{4.12}
$$

Prove this by double induction.
First induction on t. Assume $i = 1$
If $t = 1$ then

$$
E[X_1^{(1)}] = 0 \tag{4.13}
$$

$$
\frac{2}{3} - 1 \leq E[X_1^{(1)}] \leq \frac{2}{3} + 1. \tag{4.14}
$$

Now we assume Equation 4.12 is true up until $t-1$. We prove the upper bound first.

$$
\begin{aligned}
E[X_1^{(t)}] &= \left(1 - \frac{1}{2(t-1)}\right) E[X_1^{(t-1)}] && (4.15)\\
&\leq \left(1 - \frac{1}{2(t-1)}\right) \left(\frac{4}{6}(t-1) + 1\right) + 1 && \text{(By inductive hypotesis)}\\
&= \frac{2(t-1)}{3} - \frac{2(t-1)}{6(t-1)} - \frac{1}{2(t-1)} + 1 && (4.16)\\
&= \frac{2(t-1)}{3} + \left(1 - \frac{1}{3}\right) + \left(1 - \frac{1}{2(t-1)}\right) && (4.17)\\
&= \frac{2}{3}t + \left(1 - \frac{1}{2(t-1)}\right) \leq \frac{2}{3}t + 1. && (4.18)
\end{aligned}
$$

Now the lower bound.

$$
\begin{aligned}
E[X_1^{(t)}] &= \left(1 - \frac{1}{2(t-1)}\right) E[X_1^{(t-1)}] && (4.19)\\
&\geq \left(1 - \frac{1}{2(t-1)}\right) \left(\frac{4}{6}(t-1) - 1\right) + 1 && \text{(By inductive hypotesis)}\\
&= \frac{2(t-1)}{3} - \frac{2(t-1)}{6(t-1)} + \frac{1}{2(t-1)} + 1 && (4.20)\\
&= \frac{2}{3}t - \frac{2}{3} - \frac{1}{3} + \frac{1}{2(t-1)} + 1 && (4.21)\\
&= \frac{2}{3}t + \frac{1}{2(t-1)} \geq \frac{2}{3}t - 1. && (4.22)
\end{aligned}
$$

Now we have proved for the base case $i = 1$. Assume Equation 4.12 is true up until $i - 1$. Let's now prove the upper bound.

$$
\begin{aligned}
E[X_i^{(t)}] &= \left(1 - \frac{i}{2(t-1)}\right) E[X_i^{(t-1)}] + \frac{i-1}{2(t-1)} E[X_{i-1}^{(t-1)}] && (4.23)\\
&\leq \left(1 - \frac{i}{2(t-1)}\right) (E_i(t-1)+1) + \frac{i-1}{2(t-1)} (E_{i-1}(t-1)+1) && (4.24)\\
&= \left(1 - \frac{i}{2(t-1)}\right) \left(\frac{4(t-1)}{i(i+1)(i+2)} + 1\right) && (4.25)\\
&\quad + \frac{i-1}{2(t-1)} \left(\frac{4(t-1)}{(i-1)(i)(i+1)} + 1\right) && (4.26)\\
&= \frac{4(t-1)}{i(i+1)(i+2)} + 1 - \frac{2}{(i+1)(i+2)} - \frac{i}{2(t-1)} + \frac{2}{i(i+1)} + \frac{i-1}{2(t-1)} && (4.27)\\
&= \frac{4(t-1)}{i(i+1)(i+2)} + 1 + \frac{4}{i(i+1)(i+2)} - \frac{1}{2(t-1)} && (4.28)\\
&= \frac{4t}{i(i+1)(i+2)} + 1 - \frac{1}{2(t-1)} \leq E_i t + 1. && (4.29)
\end{aligned}
$$

The lower bound now.

$$
\begin{aligned}
E[X_i^{(t)}] &= \left(1 - \frac{i}{2(t-1)}\right) E[X_i^{(t-1)}] + \frac{i-1}{2(t-1)} E[X_{i-1}^{(t-1)}] && (4.30)\\
&\geq \left(1 - \frac{i}{2(t-1)}\right) (E_i(t-1)-1) + \frac{i-1}{2(t-1)} (E_{i-1}(t-1)-1) && (4.31)\\
&= \left(1 - \frac{i}{2(t-1)}\right) \left(\frac{4(t-1)}{i(i+1)(i+2)} - 1\right) && (4.32)\\
&\quad + \frac{i-1}{2(t-1)} \left(\frac{4(t-1)}{(i-1)(i)(i+1)} - 1\right) && (4.33)\\
&= \frac{4(t-1)}{i(i+1)(i+2)} - 1 - \frac{2}{(i+1)(i+2)} + \frac{i}{2(t-1)} + \frac{2}{i(i+1)} - \frac{i-1}{2(t-1)} && (4.34)\\
&= \frac{4(t-1)}{i(i+1)(i+2)} - 1 + \frac{4}{i(i+1)(i+2)} + \frac{1}{2(t-1)} && (4.35)\\
&= \frac{4t}{i(i+1)(i+2)} - 1 + \frac{1}{2(t-1)} \geq E_i t - 1. && (4.36)
\end{aligned}
$$

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).

[4] Nemhauser, Wolsey, and Fisher. “An analysis of approximations for maximizing submodular set functions”. In: *Mathematical Programming* (1978).