Social and Behavioural Networks

Uniroma1 2018-19 and 2019-20

Andrea Proietto, Giovanni ficarra

September 27, 2019

Contents

Chapter 1

Preamble

Questi appunti utilizzano una notazione personale che sto portando avanti da anni, con l'obiettivo di essere il meno ambiguo possibile. Un caso comune è il seguente: per definire una funzione, utilizzo questa sintassi:

$$f \in A \to B : a \mapsto f(a) \tag{1.1}$$

dove f è la funzione, $A \to B$ è il **tipo** di funzione, specificato dal dominio A e dal codominio B, ed è trattato come insieme di tutte le funzioni da A a B, giustificando l'uso dell'operatore di appartenenza. La parte che segue i due punti definisce la funzione attraverso la notazione di mappatura. Ad esempio, l'espressione seguente:

$$d \in \mathbb{N} \to \mathbb{N} : n \mapsto 2 \cdot n \tag{1.2}$$

si leggerebbe: "d è una funzione che va dai naturail ai naturali tale che ad n associa 2n"

Chapter 2

Introductory pills

2.1 Objectives of the course

The main objective of the course are:

1. Find efficient solutions to analyze social networks;
2. Define models that describe users' behavior;
3. Find ways to infer missing data in a social network.

Our models have to adapt well to real cases but also to be understandable.

Our algorithms have to be efficient: they need few memory and execution time, and they can easily be executed in parallel on more computers, using frameworks such as Pregel or Map-Reduce.

2.2 Basic ideas

Here we enumerate some preliminary notes and formulas that should be already known and will be useful during the course:

- *Note*: "pick" and "sample" are used interchangeably;
- We call *node* an entity on the web that can be a blog, a website or a social network profile
- A useful property of logarithms:
$$x^{\ln x} = x \tag{2.1}$$
- A recurrent inequality:
$$1 - x \leq e^{-x} \tag{2.2}$$
- Let E_i be mutually independent events, then
$$Pr\left\{\bigcap_{i=0}^{n} E_i\right\} \leq \prod_{i=0}^{n}(Pr\{E_i\})) \tag{2.3}$$

Then we have some useful definitions.

Definition 2.2.1 (Expected value)**.** *$E(X)$ is called the expected value because it means the midpoint where all outcomes, compounded with their weight, contribute to the equilibrium.*

Question 2.2.1. Can we still talk about expected value when the outcomes are not numeric? What could be a good bijection between a generic sample space Ω and $\mathbb{N}$?

Definition 2.2.2 (Sampling)**.** *It is a simple algorithmic approach used for abstracting and estimating the percentage of elements of a certain type in a large set.*

Example 2.2.1. If there is a huge number of black and white marbles in a (very big) bowl and you'd like to know the ratio of black marbles, you can't count all of them, so you extract a sample of marbles many times, until you have sufficient confidence in the obtained result.

Definition 2.2.3 (Bayes' Theorem)**.** *Bayes' theorem (or law or rule) describes the probability of an event, based on prior knowledge of conditions that might be related to the event.*

$$\Pr\{(A \cap B)\} = \Pr\{(A)\} \cdot \Pr\{(B|A)\} \tag{2.4}$$

2.2.1 Pregel framework

Brief description:

- Each node can communicate only with its neighbors;
- The computation si asynchronous;
- Each node can use few local memory (since there are many nodes);
- The nodes of a network (graph) are distributed on many computers, so it's important to group them based on community to reduce the number of communications among computers, and to accurately choose a threshold between the memory to assign to each node and the number of nodes we want to put into each single machine.

Example 2.2.2 (Find connected components of a graph using Pregel). Each node contains a unique number at the beginning (an ID), then every one send its ID to its neighbors, and when a node receives a new number replace its ID with the received one, if it's geater than its own.
When the values stop changing every connected component has the same value. The maximum number of iterations is given by the diameter of the graph.

2.2.2 Union bound

Definition 2.2.4 (Union Bound)**.** *Let A and B be two events, then*

$$\Pr\{(A \vee B) \leq \Pr\{(\} A) + \Pr\{(\} B)\} \tag{2.5}$$

Proof. The theorem directly follows from the fact that probabilities are non-negative and the following property:

$$Pr\{(A \vee B) = Pr\{(\} A) + Pr\{(\} B)\} - Pr\{(A \wedge B)\}$$

□

Definition 2.2.5 (Generalized Union Bound)**.** *Let E_i be a countable set of events, then*

$$\Pr\left\{\bigcup_{i=0}^{n} E_i\right\} \leq \sum_{i=0}^{n} (\Pr\{E_i\}) \tag{2.6}$$

If the events are mutually disjoint, then the equality is strict.

2.2.3 Markov inequality

Let X be a non-negative random variable (RV) in a sample space Ω, and a an outcome in Ω, then:

$$Pr\{(\} X \geq a) \leq \frac{E[X]}{a} \tag{2.7}$$

Proof:

$$\begin{aligned} E(X) &= \int_0^\infty x Pr\{(\} X = x) dx \\ &\geq \int_a^\infty x Pr\{(\} X = x) dx \\ &\geq \int_a^\infty a Pr\{(\} X = x) dx \\ &= a Pr\{(\} X \geq a) \end{aligned}$$

Informally, the Markov inequality let us tell something about an extreme event in a probability distribution of which we know only the expectation: if $E[X]$ is small, then X is unlikely to be very large.

More about this here or here.

2.2.4 Moment generating functions

Moments, in a mathematical-analytical sense, are "quantitative measures" that describe characteristic of a shape. E.g.: a generic function's first moment is its slope (first derivative).
Given a random variable X, then its moment generating function is

$$M_X(t) = e^{tX}, t \in \mathbb{R} \tag{2.8}$$

To be continued...

Reminder! Power series of the e^x function: $e^x = \sum_{i=0}^{\infty} \frac{x^n}{n!}$

Chapter 3

Modeling users' behaviors

If we want to infer some information about a huge graph, such as the ones behind the web or Facebook, we usually can't perform our computation directly on the real graph, so we use models of the graph.

Some properties we usually are interested in when dealing with a network (or a graph) are the following:

- the overall number of nodes,
- the average degree,
- the number of nodes with a certain degree,
- the size of the communities,
- the degree distribution.

3.1 Erdős–Rényi model

This is a model for generating random graph, that has many applications due to its simplicity, but doesn't fit real world networks: each node has more or less the same degree, based on a fixed probability, while real networks have a gaussian distribution of the degrees.

3.2 Preferential attachment

This is another model for generating random graphs, but it follows the rule "the rich get richer", meaning that nodes with higher degree will see their degree becoming higher and higher.

The rule used to build a graph is the following: when you have built a graph with $N-1$ nodes, you add the N-th node with an edge that goes from N to a node i chosen accordingly to the following probability:

$$Pr\{\text{neighbor of } N \text{ is } i\} = \frac{deg(i)}{\sum_{k=1}^{N} deg(k)}$$

where the denominator is a normalization factor.

There exist many variants of this model, for example with each node creating k edges instead of one, with no self-loop, etc.

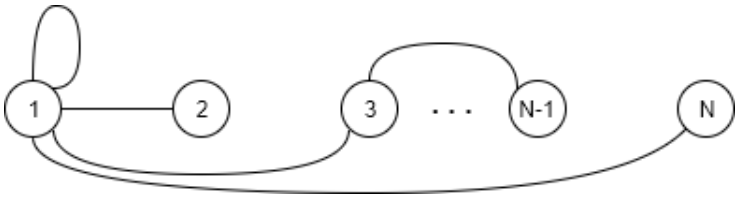

Figure 3.1: Preferential attachment

The graphs generated by the Preferential attachment model follow a power law degree distribution, like the one observed in many real world networks, indeed, the fraction of nodes of degree x approaches x^{-3}.
This characteristic causes that this model fits well with social and biological networks, so it can be used to develop efficient algorithms that actually work in practice.

Now we give a general definition of power law:

Definition 3.2.1 (Power law)**.** *A power law is a functional relationship $y = ax^{-c}$ between two quantities, where one quantity varies as a power of the other.*

As a consequence of the definition, we get that a power law appears as a line in a log log scale plot, as can be seen in fig. 3.2.

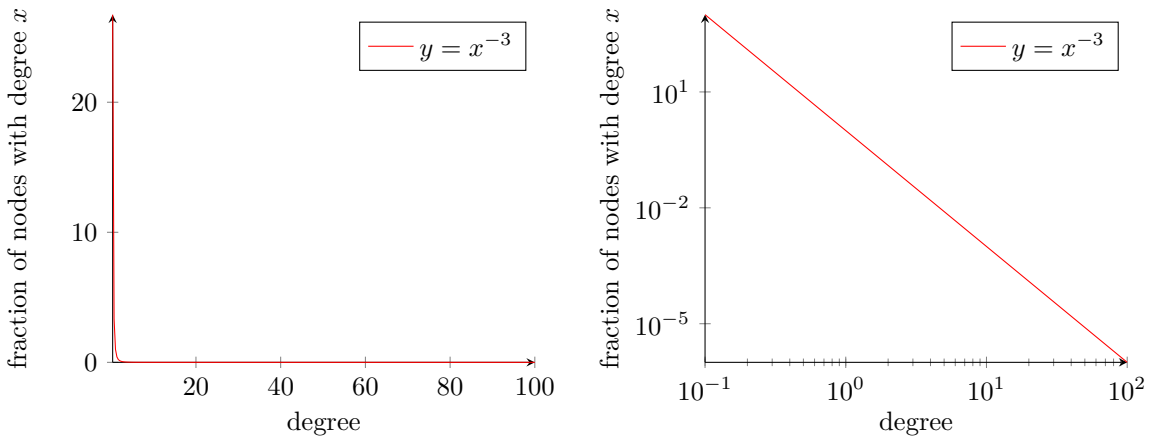

Figure 3.2: Power law distribution with linear and log log scale

Chapter 4

Information spreading [1]

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:

Definition 4.0.1 (Meme)**.** *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 some mathematical models that describes the spread of information.
Let's note that in this case, unlike in the models seen in section [3], the models work on the graph without changing it.

4.1 Push

First, a source node picks a neighbor UAR and sends to it the meme.

Then, each node that contains the meme choose a random neighbor and sends to it the meme, iteratively.

4.2 Pull

As in the previous model, the information starts in a single node.

Each node x in the network, that doesn't contain the meme, chooses a neighbor y UAR; if y has the information, x obtains the information too.

The procedure repeats many times, but the upper bound of the number of iterations is given by the number of rounds necessary to spread the ifnromation in the whole graph, that depends on the structure of the graph. For example, a star will require much less time than a chain.

This model can be useful to choose the best node to advertise a product.

[1] Most of this chapter is taken from this repo by Cristian Di Pietrantonio.

4.3 The Trace Spreading Model

In this model, nodes can receive or take information from other nodes, enabling the spread of the meme. They 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 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.

Definition 4.3.1 (Trace)**.** *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?

4.4 The First-Edge Algorithm

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

```
FirstEdge(T) // the set of traces
        E ← {} //the set of edges in the graph
        for{t ∈ T}{
                u ← t[0] //the first node in t
                v ← t[1] //the second node in t
                E ← E ∪ {(u, v)}
        }
```

Listing 4.1: The first-edge algorithm

Consider Algorithm 4.1. 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 4.4.1. *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 $n = |V|$, $\Delta = \max_v \deg(v)$ 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

$$Pr\{\xi_1\} = Pr\{t[0] = x\}Pr\{t[1] = y, y \in \{u, v\} \setminus \{x\} \mid t[0] = x\},\ x \in \{u, v\} \tag{4.1}$$
$$= \frac{2}{n}\frac{1}{\deg(x)}, x \in \{u, v\} \tag{4.2}$$
$$\geq \frac{2}{n\Delta}. \tag{4.3}$$

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.

$$Pr\{\textit{An edge in the graph is not inferred}\} = \left(1 - \frac{2}{n\Delta}\right)^{|T|} \leq \left(1 - \frac{2}{n\Delta}\right)^{cn\Delta \log n} \tag{4.4}$$

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

Lemma 4.4.2.

$$1 - x \leq e^{-x} \tag{4.5}$$

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

$$Pr\{\textit{Edge \{u, v\} in the graph is not inferred}\} = \left(1 - \frac{2}{n\Delta}\right)^{|T|} \tag{4.6}$$
$$\leq \left(1 - \frac{2}{n\Delta}\right)^{cn\Delta \log n} \tag{4.7}$$
$$\leq \left(e^{-\frac{2}{n\Delta}}\right)^{cn\Delta \log n} \tag{4.8}$$
$$= e^{\log n^{-2c}} \tag{4.9}$$
$$= \frac{1}{n^{2c}}. \tag{4.10}$$

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

$$\begin{aligned}
Pr\{\textit{The algorithm fails}\} &= Pr\left\{\bigcup_{\{a,b\}\in E} \xi_{a,b}\right\} && (4.11)\\
&\leq \sum_{\{a,b\}\in E} Pr\{\xi_{a,b}\} && \text{(by union bound)}\\
&= |E| Pr\{\xi_{a,b}\} && (4.12)\\
&\leq n^2 \frac{1}{n^{2c}} && (4.13)\\
&= \frac{1}{n^{2(c-1)}}. && (4.14)
\end{aligned}$$

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

4.5 Chernoff bound

The Chernoff bound allows you to control how much a RV is close to its expected value.

Let $X_1, \ldots, X_n \in \mathcal{B}er(p)$ IID, i.e. $X_1, \ldots, X_n$ are n RV such that
$Pr\{X_i = 1\} = p$ and $Pr\{X_i = 0\} = 1 - p$.
In other words, they are n independent throws of the same coin.

Let $X = \sum_{i=1}^{n}(X_i)$. X is essentially a binomial RV: $X \in \mathcal{B}inom(\mathbb{2}^n, p)$, thus $E[X] = p \cdot n$. It represents the number of times you get head with the aforesaid n throws.

So, given an error limit ε, we have the following definition:

Definition 4.5.1 (Chernoff bound)**.**

$$\Pr\{|X - E[X]| \geq t\} \leq e^{-2t^2/n} \tag{4.15}$$

where $t := \varepsilon n$.

In the binomial case, [4.15] translates to:

$$Pr\left\{\left|\sum_{i=1}^{n}(X_i) - p \cdot n\right| > t\right\} = Pr\left\{\left|\frac{1}{n}\sum_{i=1}^{n}(X_i) - p\right| > \frac{t}{n}\right\}$$

Then, replacing t with εn, we get

$$Pr\left\{\left|\frac{1}{n}\sum_{i=1}^{n}(X_i) - p\right| > \varepsilon\right\} \leq 2e^{-2n^3\varepsilon^2} \tag{4.16}$$

Example 4.5.1. If you want to know how many Facebook users have more than 100 friends, you can't simply count, but you can obtain a good approximation sampling a reasonable number of profiles.

It's important to observe that the size of the sample we need in order to obtain a good approximation depends on the error we tolerate, not on the total population.

There are many possible formulations of the Chernoff bound, another one that can be useful is the following:

$$Pr\{|X - E[X]| \geq \varepsilon \cdot n\} \leq 2e^{-\frac{\varepsilon^2}{3}n} \tag{4.17}$$

Example 4.5.2. Let be $E[X] = \frac{50}{100}$ and $\varepsilon = \frac{1}{100}$, the probability that $X \geq \frac{51}{100}$ or $X \leq \frac{49}{100}$ is less then or equal to $2e^{-\frac{1}{30000}n}$.

Chapter 5

Locality Sensitive Hashing

The goal of hashing techniques is to reduce a big "object" to a small "signature" or "fingerprint".
In general, what happens in locality sensitive hashing (or LSH) is to have some notion of similarity, and then define a "scheme" which computes it. The process of creating a scheme usually involves some sort of preprocessing step, and a function family which, by choosing one or another function according a probability distribution, statistically classifies the objects in the same way as the similarity function does.
The bottom line of LSH schemes is: similar objects hash to similar values (this is "*locality*").

Here are some common similarities:

Definition 5.0.1 (Jaccard similarity). *Given two sets of objects A and B, their Jaccard similarity is defined as follows:*

$$\mathcal{J}acc(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.1}$$

Definition 5.0.2 (Hamming similarity). *Given two sets of objects A and B taken from a universal set U, their Hamming similarity is defined as follows:*

$$\mathcal{H}amm(A,B) = \frac{|A \cap B| + |\complement_U(A \cup B)|}{|U|} \tag{5.2}$$

5.1 A case study: Web-page indexing

A search engine crawls periodically the whole Internet and stores valuable information in its own index for search optimization purposes.

Observation 5.1.1. Some kinds of documents, that are very similar to each other, are stored sparsely through the net; to save storage space, only one of a kind of document's info is stored in the index, whereas all others are linked to the first one, because of their similarity.

To find a useful hashing scheme, A. Broder came up with an idea, and he succeeded in reducing the storage space needed by Altavista by a factor of 10.
First off, let us fix some definitions:

- U: the set of all words, i.e. the English vocabulary
- U^*: the set of all strings composed of English words

The starting point is to treat web pages as strings:

T_1 : *"The black board is black"*

T_2 : *"The white board is white"*

Then, let $distinct(T)$ return the set of distinct words appearing in a string (ref. Bag of Words model).

So for example, by using the Jaccard similarity:

$$\mathcal{J}acc(distinct(T_1), distinct(T_2)) = \frac{3}{5} \tag{5.3}$$

Over a half: they look close. If we used the Hamming distance instead, we would (almost always) get a number very close to 1, because we're using a minuscule part of the universe set (in our case, the English dictionary), thus (almost) all words are absent from the sets.

Now, our objective is to construct a scheme over web-pages that implement the Jaccard similarity. Our pre-processing step: Choose a permutation (or total ordering) $\pi \in \mathcal{P}erm(A \cup B)$ UAR. To construct said order is a simple task, as we can see with the following algorithm.

Algorithm for sampling a UAR permutation π of $[n]$, where $[n]$ is a numeric representation of U, and using it to compute the hash of A:

```
MinHash(A):
    // preprocessing
    S := [n]
    π := empty sequence
    while S ≠ ∅:
        pick i ∈ S UAR
        append i to π
        remove i from S
    end
    // computing hash
    h_π(A) := minimum element of A, accordig to π
    return h_π(A)
```

Listing 5.1: min hash or shingles algorithm

Proof of uniform choice of π.

$$\begin{aligned} Pr\{X = \pi\} &= \frac{1}{|S|} \cdot \frac{1}{|S|-1} \cdot \cdots \cdot 1 \\ &= \frac{1}{|S|!} \Rightarrow X \in \mathcal{U}nif(\mathcal{P}erm(A \cup B)) \end{aligned}$$

□

So, from π, we obtained the following definition:

Definition 5.1.1 (Hashing function).

$$h_\pi \in \mathcal{P}(U) \to U : h_\pi(A) = min_\pi(A) \tag{5.4}$$

In other words, we take the "minimum" in A according to the ordering specified by π.

Observation 5.1.2. A simple but useful observation would be:

$$\forall A \subseteq U \Rightarrow h_\pi(A) \in A \tag{5.5}$$

Example 5.1.1.

$$\pi = (black, the, is, white, board) \wedge A = \{black, board\} \Rightarrow h_\pi(A) = black$$

Thus, we say that A is similar to B iff $h_\pi(A) = h_\pi(B)$.
Recall that A and B are fixed, π is the focus of this definition. What can be said about $Pr\{h_\pi(A) = h_\pi(B)\}$? Looking at a corresponding Venn diagram:

- $A \cap B = \emptyset \Rightarrow Pr\{h_p(A) = h_p(B)\} = 0$; they have no words in common, so their hashes must be different, independently of the chosen order;
- $A = B \Rightarrow Pr\{h_p(A) = h_p(B)\} = 1$; this time, all words are in common, so their hashes must coincide, again, independently of the chosen order;
- Otherwise, since π is chosen UAR, the probability that the hashes are equal has the same meaning of the probability of finding the lowest element of A and B in the intersection with respect of the union (and not in U as a whole, as our previous observation suggests), which is the Jaccard similarity of A and B by its very definition:

$$\begin{aligned} Pr\{h_\pi(A) = h_\pi(B)\} &= \frac{Pr\{(\} \min(A) \in A \cap B \wedge \min(B) \in A \cap B)}{Pr\{(\} \min(A) \in A \cup B \wedge \min(B) \in A \cup B)} \\ &= \frac{|A \cap B|}{|A \cup B|} = \mathcal{J}acc(A, B). \end{aligned}$$

Possible question about third point: Why not respect to the universal set? because A and B will have hashes which, as we observed earlier, do not live outside the union: the union between A and B is our true set of outcomes when hashing either A or B.

Now, if h_π is evaluated only once over a given permutation, only a binary response can be obtained. In order to obtain the probability value without resorting to compute unions and intersections, we can repeat evaluation over different permutations; this can be regulated by the **Chernoff-Hoeffding bound**:

Let $A, B \subseteq U$, and $X_{1\dots n} \in \mathcal{B}er(p)$ IID, with $Pr\{X_i = 1\} = p$ and $Pr\{(\} X_i = 0) = 1 - p$, with each X_i defined over a distinct element of $\Pi \subseteq \mathcal{P}erm(U)$, such that $X_i \mapsto 1 \Leftrightarrow A \sim_{\pi_i} B, 0$ *otherwise*, then:

$$Pr\{|avg_{i=1}^{n}(X_i - p)| \geq \varepsilon\} \leq 2e^{-n\varepsilon^2} \tag{5.6}$$

In other words, the difference between the average of the X_i (i.e., the average of the empirically observed results) and their exact probability is greater then ε only with a very small probability.

So, how many trials (evaluations, observations) are needed to have a good estimate of the similarity? That is, what is a good value for n?
Let $X_i = \begin{cases} 1 & \text{if } h_{\pi_i}(A) = h_{\pi_i}(B) \\ 0 & \text{otherwise} \end{cases}$ and $Pr\{X_i = 1\} = \mathcal{J}acc(A, B) = p$; we can apply the Chernoff bound on X_i to compute our n.
If our database has m pages (sets) to store, we can chose $n = \frac{\lg \frac{2m}{\delta}}{\varepsilon^2}$ to get a high probability of making zero errors; δ and ε are parameters we can set to adjust the size of n: even if the bound gives us a high probability for a quite small n, we can choose an even smaller n if we can accept big errors for very few pages.

We can now observe that the min hash algorithm [5.1] if efficient: instead of comparing two entire pages, it only compares n integers.

5.2 A case study: Comparing DNAs

In the previous case study, we considered small subsets of the universe set: each web page has only few words with respect to the whole English language.
However, there are cases in which the overlays between two subsets are often relevant, for example, if we want to compare the DNA of two people.
In such cases, the Hamming similarity is preferable (more significant) to the Jaccart similarity.

We can describe two DNA sequences A and B as two arrays of size n, in which each position corresponds to a certain component of the DNA, and contains a 1 if that component is present in that sequence and a 0 if it's absent.

$$\begin{aligned} \mathcal{H}amm(A, B) &= \frac{|A \cap B| + \overline{|A \cup B|}}{n} \\ &= \frac{\text{number of common } 1s + \text{number of common } 0s}{n} \end{aligned}$$

To get our hash function we pick an index $i \in [n]$ UAR, and we define
$h_i(A) = \begin{cases} 1 & \text{if } i \in A \\ 0 & \text{otherwise} \end{cases}$.

Example 5.2.1. We have two DNA sequences A and B such that
$A := \boxed{0}\boxed{0}\boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{1}$, $B := \boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{1}\boxed{1}\boxed{1}$ and $n := 8$, so we can write A and B as $A = \{4, 5, 8\}, B = \{2, 3, 6, 7, 8\}$ using the positions with a 1 inside (starting from 1).
If we randomly choose $i = 8$, we obtain $h_i(A) = h_i(B) = 1$, so we can conclude that A and B are similar.

Now we can see that the probability that two hashes are equal is the Hamming similarity:

$$\begin{aligned} Pr\{h_i(A) = h_i(B)\} &= \frac{Pr\{A[i] = B[i]\}}{n} \\ &= \frac{Pr\{(A[i] \text{ and } B[i] \text{ are both 1}) \vee (A[i] \text{ and } B[i] \text{ are both 0})\}}{n} \\ &= \frac{|A \cap B| + \overline{|A \cup B|}}{n} = \mathcal{H}amm(A, B) \end{aligned}$$

It's possible to obtain a good estimate of the similarity by repeating the test an appropriate number of time, given by the Chernoff Bound.

5.3 LSH formalization

First let us focus around the hash function as an object: its true purpose in a scheme is to classify objects based on how much they "look like", whatever this means in the chosen similarity's terms. Therefore, in our theoretical analysis, the codomain of a hash function is not that important; what is important is how the function partitions its own domain, U. In a sense, we're interested only in the partitions of U themselves, not in the functions that generate them.

Why have we dealt with functions back then? Moving from a purely mathematical perspective to a more computational one, what is usually done for measuring similarities is sampling some object's characteristics, and observe how "distant", or else "similar" they are. This is done by means of some program; and programs are (oh so) easily associated with functions. The computational approach gives a more intuitive vision of the problem we're confronting ourselves with.

Still, what could happen, is to have a couple of functions that map values into wildly different codomains, but partition U in exactly the same way! And in our journey, we're just interested in classifying objects; so these kind of "duplicate" functions are, well, useless (unless we delve in complexity studies, but that's out of our scope).

So, let us reform the foundations by taking as our core object a universe partition, instead of a universe-domained hash function. First, though, we need to formalize what a similarity is, and to get to a good definition, we have to carefully select them from their space $U^2 \to [0, 1]$. It should be noted that the codomain might very well be $\mathbb{R}$ itself, but to get some bearings we'll treat an image of 1 as a complete equivalence between two objects, and 0 for complete difference, with the interval expressing the degree of similarity.

Let U be a set, and $S \in U^2 \to [0, 1]$ a symmetric function; then S is called a **similarity** over U.

Tidbit: Let $f \in A^n \to B$, then f is **symmetric** iff *argument order does not change the image.*

Example 5.3.1.

$$\begin{aligned} U &= 2^{[n]} = \{A | A \subseteq [n]\} \\ S &= \mathcal{J}acc \\ S\left(\{1, 2\}, \{2, 3\}\right) &= \frac{1}{3} \end{aligned}$$

A **LSH scheme** over U is a probability distribution over the partitions of U.

Example 5.3.2. We can apply the min-hash scheme [5.1] to the Jaccard Similarity. With $U = [3]$ and $\pi = 1 < 2 < 3$, the function maps each subset of U to a hash as follows:

$$\begin{aligned} \emptyset &\mapsto \perp \text{ (the hash of the empy set is a special symbol)} \\ \{1\}, \{2, 1\}, \{1, 3\}, \{1, 2, 3\} &\mapsto 1 \text{ (all sets containing 1 have hash 1)} \\ \{2\}, \{2, 3\} &\mapsto 2 \text{ (all remaining sets containing 2 have hash 2)} \\ \{3\} &\mapsto 3 \text{ (all remaining sets containing 3 have hash 3)} \end{aligned}$$

thus defining 4 partitions over U.

Example 5.3.3. Similarly, we can apply the function based on the Hamming Similarity we saw in [5.2] to the same set $U = [3]$ and see we get new partitions. We choose $i = 2$ UAR, the function maps each subset of U to a hash as follows:

$$\{2\}, \{2.1\}, \{2, 3\}, \{1, 2, 3\} \mapsto 1 \text{ (all sets containing } i \text{ have hash 1)}$$
$$\emptyset, \{1\}, \{3\}, \{1, 3\} \mapsto 0 \text{ (all remaining sets have hash 0)}$$

thus defining 2 partitions over U.

Let's make some other considerations about what we have just seen.

Observation 5.3.1. Given a similarity ϕ, a LSH scheme is a family of hash functions H, coupled with a probability distribution D over H such that, chosen a function h from the family H according to D, h satisfies the property $Pr\{h(a) = h(b)\} = \phi(a, b) \forall a, b \in U$.
In other words, let $S \in U^2 \to [0, 1]$ be a similarity, and H be a RV over a family of hash functions over U, then H is a LSH scheme iff $Pr\{H(a) = H(b)\} = S(a, b) \forall a, binU$

Observation 5.3.2. Preprocess and hash function (aka a scheme) determine the similarity function (most people attempt to do the reverse)

Observation 5.3.3. In the previous webpage example [5.1], we're not dealing with a single hashing function, but with a family of functions each built with its own word permutation: the scheme distributes over the permutations of the union!

Before going on, let's recap what we have discussed so far:
A LSH scheme for a similarity S is a prob. dist. over U's partitions such that

$$\forall A, B \in U \Rightarrow Pr\{A \sim_p B\}_p = S(A, B) = Pr\{h(A) = h(B)\}_h \tag{5.7}$$

where p is a partitioning of U and $\sim_p$ means A and B are in the same partition.

Another possible definition: Given $s \in U^2 \to [0, 1]$ a similarity, then we define X to be a LSH scheme for s as the following:

$$X \in \mathcal{R}and(\mathcal{P}(U)) : \forall A, B \in U \implies \mathcal{E}(A \sim_X B) = S(A, B) \tag{5.8}$$

Challenge: Can we find a LSH scheme for an arbitrary S function? NO

Example 5.3.4. Given a universe set $U = \{a, b, c\}$ and a similarity function S s. t. $S \in U^2 \to [0, 1]$: $S(a, b) \mapsto 1, S(b, c) \mapsto 1, S(a, c) \mapsto 0$ we don't have an LSH, since we're violating transitivity: Translating into probabilities and using equality's transitivity, we obtain: $Pr\{h(a), h(c)\} = 1$, which contradicts the third mapping.

5.4 More distances

Let $A \triangle B = (A - B) \cup (B - A)$.

Dice similarity:

$$D(A, B) = \frac{|A \cap B|}{|A \cap B| + \frac{1}{2}|A \triangle B|} \tag{5.9}$$

Anderberg similarity:

$$An(A, B) = \frac{|A \cap B|}{|A \cap B| + 2|A \triangle B|} \tag{5.10}$$

Generalizing Jaccard, Dice and Anderberg:

$$S_\gamma(A, B) = \frac{|A \cap B|}{|A \cap B| + \gamma|A \triangle B|} \tag{5.11}$$

By this third definition we can obtain $\mathcal{J}acc$ with $\gamma = 1$, D with $\gamma = \frac{1}{2}$, and An with $\gamma = 2$.

Question: for which values of γ does an LSH exist for S? The next lemma helps us finding an answer.

Lemma 5.4.1. *(by Charikar) If a similarity S admits an LSH, then the function $1 - S$ must satisfy the triangular inequality, i.e.* $\forall A, B, C \in U$
$d(A, B) \leq d(A, C) + d(B, C)$*, where the distance $d(A, B) = 1 - S(A, B)$ is our function $1 - S$.*

Proof: Let E_{XY} be the event "$h(X) \neq h(Y)$"; since S admits a LSH, we have:

$$d(A,B) = 1 - S(A,B) = \Pr_h\{E_{AB}\} = p_1 + p_2 + p_3 + p_4$$
$$d(A,C) = 1 - S(A,C) = \Pr_h\{E_{AC}\} = p_1 + p_3 + p_5 + p_7$$
$$d(B,C) = 1 - S(B,C) = \Pr_h\{E_{BC}\} = p_1 + p_2 + p_5 + p_6$$

with the probabilities p_i defined as in the following table, where an X under the "may exist" column means that the corresponding probability is 0 (it happens because transitivity generates contradictions, see example [5.3.4]):

	E_{AB}	E_{AC}	E_{BC}	*may exist*
p_1	T	T	T	✓
p_2	T	T	F	✓
p_3	T	F	T	✓
p_4	T	F	F	X
p_5	F	T	T	✓
p_6	F	T	F	X
p_7	F	F	T	X
p_8	F	F	F	✓

Hence we have

$$d(A,C) + d(B,C) = 2p_1 + p_2 + p_3 + 2p_5 \geq p_1 + p_2 + p_3 = d(A,B)$$

and so $1 - S$ doesn't comply with the triangular inequality, QED.

Observation 5.4.1. Similarities are actually defined in most cases as the inverses of measures, which in turn give (oh so surprisingly!) a notion of distance.

Corollary 5.4.1.1. *By Charikar's lemma, we can prove that Dice's similarity cannot admit a LSH scheme.*

Proof by counterexample: Assume $A = \{1\}, B = \{2\}, C = \{1,2\}$, then use the triangular inequality over the distances:
$D(A,C) = \frac{2}{3},\ D(B,C) = \frac{2}{3},\ D(A,B) = 0$

$$d(A,B) = 1 - D(A,B) = 1 > \frac{2}{3} = (1 - D(A,C)) + (1 - D(B,C)) = d(A,C) + d(B,C)$$

hence it doesn't comply with the triangular inequality, QED.
(Note that D stands for Dice similarity and d for distance)

Observation 5.4.2. Parameterizing this counterexample with S_γ, we obtain a bounds for γ: let $A = \{1\}, B = \{2\}, C = \{1,2\}$ and
$S_\gamma(A,C) = \frac{1}{1+\gamma},\ S_\gamma(B,C) = \frac{1}{1+\gamma},\ S_\gamma(A,B) = 0$, thus

$$1 = 1 - S_\gamma(A,B) = d(A,B) > d(A,C) + d(B,C) = (1 - S_\gamma(A,C)) + (1 - S_\gamma(B,C)) = 2\left(1 - \frac{1}{1+\gamma}\right) = \frac{2\gamma}{1+\gamma}$$

from which we obtain $1 > \frac{2\gamma}{1+\gamma} \Rightarrow \gamma < 1$.
Hence, if $\gamma < 1$, the triangular inequality doesn't hold, so no LSH can exist, as in the case of the Dice similarity.

5.5 Probability generating functions

Intuition: A probability generating function (PGF) is a power series representation of a given probability distribution

Definition: Given a (discrete) RV X, its **PGF** is the function:

$$\mathcal{G}en_X(\alpha) = \sum_{x=0}^{\infty} Pr\{X = x\}\, \alpha^x \tag{5.12}$$

(note that all outcomes appear by their probability).

How to get back to pmf: $Pr\{X = x\} = \dfrac{\mathcal{D}^x(\mathcal{G}en_X(0))}{x!}$

In other terms, a PGF f is a function:

$$f(x) = \sum_{x=0}^{\infty} \left(p_i x^i\right) \tag{5.13}$$

s.t. $p_i \geq 0 \forall i$ and $\sum_{x=0}^{\infty} p_i = 1$.

Theorem 5.5.1. *: If a similarity S admits a LSH and a given function f is a PGF, then $f(S)$ admits a LSH.*

$$f(S(A,B)) = (f(S))(A,B) =: T(A,B)$$

Before going into the proof of the theorem we show a consequence of it.

Observation 5.5.1. Applying the theorem to the Jaccard similarity:
Our function is f_γ, with $\gamma > 1$, defined as

$$f_\gamma(x) = \frac{x}{x + \gamma(1-x)} \tag{5.14}$$

In order to proof f_γ is a PGF, we have to demonstrate that the coefficients represent a probability distribution: i.e., they are all positive and they sum to zero.
By applying the Taylor series expansion to [5.14], we get
$f_\gamma(x) = \sum_{x=1}^{\infty} \left(\frac{\left(1 - \frac{1}{\gamma}\right)^i}{\gamma - 1} x^i \right)$; now f_γ is a power series, so all the coefficients are positive.
In order for the sum of the coefficients to be equal to 1, the numerator must be equal to the denominator:
$\sum_{i=1}^{\infty} \left(\frac{\left(1 - \frac{1}{\gamma}\right)^i}{\gamma - 1} \right) = 1 \Rightarrow \sum_{i=1}^{\infty} \left(\left(1 - \frac{1}{\gamma}\right)^i \right) = \gamma - 1.$
Indeed we have:

$$\begin{aligned}
\sum_{i=1}^{\infty} \left(1 - \frac{1}{\gamma}\right)^i &= \left(1 - \frac{1}{\gamma}\right) \sum_{i=0}^{\infty} \left(1 - \frac{1}{\gamma}\right)^i \\
&=^{(*)} \left(1 - \frac{1}{\gamma}\right) \frac{1}{1 - \left(1 - \frac{1}{\gamma}\right)} \\
&= \frac{\gamma - 1}{\gamma} \frac{1}{1/\gamma} = \gamma - 1
\end{aligned}$$

where the step marked with $(*)$ is due to the equality $\sum_{i=0}^{\infty} \alpha^i = \dfrac{1}{1-\alpha}$.

And with this we proved that f_γ is a PGF.

Now we can apply PGF to a similarity S_γ:

$$\begin{aligned} f_\gamma(S_\gamma(A,B)) &= f_\gamma\left(\frac{|A\cap B|}{|A\cup B|}\right) \\ &= \frac{\frac{|A\cap B|}{|A\cup B|}}{\frac{|A\cap B|}{|A\cup B|}+\gamma\frac{|A\cup B|-|A\cap B|}{|A\cup B|}} \\ &= \frac{|A\cap B|}{|A\cap B|+\gamma|A\vartriangle B|} = S_\gamma(A,B) \end{aligned}$$

which in turn is LSH-able

5.6 More about PGF

Recap: Given a universe U, a function $S \in U^2 \to [0,1]$ is said to be a **LSHable similarity** iff exists a prob. distr. over (a family/subset of) the hash functions in U, such that:

$$\forall\{X,Y\} \in \binom{U}{2} \;\; Pr_h\left\{h(X)=h(Y)\right\} = S(X,Y) \tag{5.15}$$

Theorem 5.6.1. *If a similarity S is LSH-able and f is a PGF, then $f(S)$ is LSHable.*

Equivalent statement:

$$f(S) := T \in U^2 \to [0,1] \text{ s.t. } \forall\{A,B\} \in \binom{U}{2} \;\; T(A,B) = f(S(A,B)) \tag{5.16}$$

Lemma 5.6.2. *(L1) The similarity $O \in U^2 \to [0,1]$ s.t. $O(A,B) \mapsto 1$ $\forall\{A,B\} \in \binom{U}{2}$ admits a LSH.*

Proof: Give probability 1 to the constant hash function h: $h(A) = 1$ $\forall A \in U$
$\Pr\{h(A)=h(B)\} = 1 = O(A,B)$ $\forall\{A,B\} \in \binom{U}{2}$.

Purpose: This will be the base case for theorem proof...

Lemma 5.6.3. *(L2) If S and T similarities over U have a scheme, then $S \cdot T : (S\cdot T)(A,B) = S(A,B)\cdot T(A,B)$ has a scheme.*
I.e. LSHability is preserved upon composition/multiplication.

Proof by construction (Algorithm):

- *Sample hash functions for $S \cdot T$ as follows:*
  1. *first, sample h_S from the LSH for S;*
  2. *then, sample h_T independently from the LSH for T;*
- *Return the hash function h s.t. $H(A) = (h_S(A), h_T(A))$ $\forall A \in U$;*
- *Thus, $\forall\{A,B\} \in \binom{U}{2}$, we have:*

$$\begin{aligned} \Pr_h\{h(A)=h(B)\} &= \Pr_{h_S}\{h_S(A)=h_S(B)\}\cdot\Pr_{h_T}\{h_T(A)=h_T(B)\} \\ &= S(A,B)\cdot T(A,B) \end{aligned}$$

by independency.

Lemma 5.6.4. *(L3) if S is LSHable, then S^i is LSHable, $\forall i \in \mathbb{N}$.*

Proof by induction:

- *Base: $i = 0$ and $S^0 = O$*
  True for L1 [5.6.2];
- *Inductive hypothesis: S^i is LSHable;*
- *Inductive step: S^{i+1} is LSHable*
  True because $S^{i+1} = S \cdot S^i$ is LSHable by L2 [5.6.3], since S has a scheme by def and S^i has a scheme by inductive hypothesis.

Lemma 5.6.5. *(4) If $p_0, p_1, ..., p_i, ...$ is s.t. $\sum_{i=0}^{\infty} p_i = 1$, $p_i \geq 0$ $\forall i$, and $S_0, S_1, ..., S_i, ...$ are lshable similarities, then $\sum_{i=0}^{\infty} p_i S_i$ is lshable.*

Proof by scheme:

1. *First, sample $i*$ at random from $\mathbb{N}$ with probability $p_0, ..., p_i, ...$;*
2. *Then, sample h from the hash functions (LSH) of S_{i*} (that is, we are choosing which LSH to use);*
3. *Thus we have:*

$$\begin{aligned}\Pr_h\{h(A) = h(B)\} &= \sum_{i=0}^{\infty}(p_i S_i(A,B)) \\ &= \sum_{i=0}^{\infty}(\underbrace{\Pr\{(\}\, i = i*)}_{p_i} \cdot \underbrace{\Pr_h\{h(A) = h(B) | i = i*)\}}_{S_i(A,B)}\end{aligned}$$

Observation 5.6.1. This lemma is useful if we need a weighted average $W(A,B) = \sum_{i=0}^{\infty}(p_i S_i(A,B))$.

Proof of the theorem [5.6.1]:

- We want to prove that $\sum_{i=0}^{\infty} p_i S^i$ has a scheme (is LSHable);
- By L3 [5.6.4] we know S^i has a scheme;
- By L4 [5.6.5] we know the sum is lshable;
- Proven.

Observation 5.6.2. This theorem tells us how to build a LSH: by concatenating the results of different hash functions, keeping the output small.
I.e. PGF is an approach for making schemes for similarities from other schemes.

Example 5.6.1. $\sum_{i=1}^{a}(2^{-i} \cdot x^i)$.

5.7 A mention of the sketches

Like LSH, they're a means for simplify the storage of data; they are slower than LSH but allow you to keep interesting information in a small space.

Definition: a sketch is a representation of big objects with small images. They can be used to retrieve interesting information about original objects without regain the whole original objects (i.e. they are not compression algorithms).

Example 5.7.1. It's possible to go back to $|A \cap B|$ from only $|A|, |B|, h(A), h(B)$, $\mathcal{J}acc(A,B) \pm \varepsilon$. Note that $\mathcal{J}acc(A,B) \pm \varepsilon$ can be obtained from $h(A), h(B)$ by applying the Chernoff Bounf [4.15] and the other data need only few bits to be stored.

Example 5.7.2. It's possible to approximate $\dfrac{|A \cap B|}{|A \cap B| + \frac{1}{2}|A \triangle B|}$ with $\dfrac{1}{2} \cdot |A \cup B| + \dfrac{1}{2} \cdot |A \cap B|$.

5.8 Approximation of non-LSHable functions

What can we do if there is no LSH for the similarity we want/need to use? We can approximate a function without a LSH with an LSHable one.

Example 5.8.1. We can approximate S_γ with $\mathcal{J}acc$ with an error of $\dfrac{1}{\gamma}$.

Theorem 5.8.1. *(???) Let f be a PGF, $\alpha \in [0,1]$.*

$\alpha f = \alpha f(S) \mid (1-\alpha)T$

$T \in U^2 \to [0,1] : \forall t, t' \in \binom{U}{2}\ T(t,t') = 0$ *but* $T(t,t) = 1$

$h(t) = -t$

For the 1 *case we wanted a banal partition, now with* 0 *we want a punctual partition, so we need a hash function that assigns a distinct value to each argument. (You can actually use the identity)*
Not a good scheme, because we're not shrinking data.

***Definition**: Let $S : U^2 \to [0,1]$ be a similarity, then its **distortion** is the minimum* $\delta \geq 1$ s.t. $\exists$ an LSHable similarity S' s.t. $\forall A, B \in \binom{U}{2}$*
$\dfrac{1}{\delta} \cdot S(A,B) \leq S'(A,B) \leq S(A,B)$*, where minimum* is the lower bound (if U isn't finite, there is no actual minimum).*

Observation 5.8.1. S' is the LSHable similarity closest to S; the more δ is near to 1, the closer S' is to S (i.e. δ is the *approximation factor*); if δ tends to 1, then S is LSHable.

Example 5.8.2. In the example 5.8.1, where we used $\mathcal{J}acc$ to approximate S_γ, we had $\delta \to \dfrac{1}{\gamma}$.

Question: How can I know if there is a better approximation than the one I found?

Lemma 5.8.2. *(Center lemma) Let S be a LSHable similarity s.t. $\exists \mathcal{X} \subseteq U : \forall \{x, x'\} \in \binom{\mathcal{X}}{2}\ S(x,x') = 0$, then $\forall y \in U\ avg_{x \in \mathcal{X}}(S(X,Y)) \leq \dfrac{1}{|\mathcal{X}|}$.*

Observation 5.8.2. $\mathcal{X}$ is a set of the most different objects in U and y is a center in U; so the meaning of the lemma is that the average of the distances between each point and the center is $\leq \dfrac{1}{|\mathcal{X}|}$.

Observation 5.8.3. $\mathcal{X}$ is actually a (possibly incomplete) section of the partition of U induced by h.

Observation 5.8.4. (important) From the lemma, it trivially follows that
$\exists\ x^* \in \mathcal{X} : S(x^*, Y \leq \frac{1}{|\mathcal{X}|})$.

Proof of the Center lemma:

- Fix $y \in U$;
- If the hash function h has positive probability in the (chosen) LSH for S, then $\forall x, x' \in \binom{\mathcal{X}}{2}\ h(x) \neq h(x')$ (otherwise we would have $S(x, x') > 0$);
- Thus, $\forall$ hash functions with positive probability, there can exist at most one $x \in \mathcal{X}$ s.t. $h(x) = h(y)$ (by transitivity of equality);
- Than we have:

$$\begin{aligned}\sum_{x\in\mathcal{X}} S(x,y) &= \sum_{x\in\mathcal{X}} Pr_h\{h(x) = h(y)\} \\ &= \sum_{x\in\mathcal{X}}\sum_{h} Pr\{h \text{ is chosen}\} \cdot \underbrace{[h(x) = h(y)]}_{\text{pr. that } x=y \text{ with the choosen } h} \\ &= \sum_{h} Pr\{h \text{ is chosen}\} \cdot \sum_{x\in\mathcal{X}} [h(x) = h(y)] \\ &\leq \sum_{h} Pr\{h \text{ is chosen}\} = 1\end{aligned}$$

  (Note that he square brackets here are a boolean evaluation operator);
- Thus, $\sum_{x\in\mathcal{X}} S(x, y) \leq 1$, that implies that $avg(S(X, Y)) \leq \frac{1}{|\mathcal{X}|}$, and the lemma is proven.

Example 5.8.3. We will now prove what we said about S_γ's approximation in examples [5.8.1] and [5.8.2], using the lemma we have just demonstrated [5.8.2]:

- Let us give some definitions:
  - $S := S_\gamma$, with $0 < \gamma < 1$,
  - $U := 2^{[n]} = \{S | S \subseteq [n]\}$,
  - $\mathcal{X} := \mathcal{P}_1([n]) = \{\{1\}, \{2\}, ..., \{n\}\}$;
- So, by definition of S_γ, we have $S_\gamma(\{i\}, \{j\}) = 0\ \forall i \neq j \in [n]$, since $\{i\} \cap \{j\} = \emptyset$;
- Let us assume that T (that will be our S') finitely distorts S_γ, and T is LSHable;
- Then $T(\{i\}, \{j\}) = 0\ \forall \{i, j\} in \binom{[n]}{2}$, since $S' \leq S_\gamma = 0$;

- Since T is LSHable we can apply the center lemma:

  $\exists\{i\} \in \mathcal{X}$ s.t. $T(\underbrace{\{i\}}_{\text{our } x^*}, \underbrace{[n]}_{\text{our } y}) \leq \frac{1}{|\mathcal{X}|} = \frac{1}{n}$

  $S_\gamma(\{i\}, [n]) = \frac{1}{1 + \gamma \cdot (n-1)} = \frac{1}{\gamma \cdot n + (1-\gamma)}$

  so $\exists i \in [n]$ s.t. $S_\gamma(\{i\}, [n]) \geq \frac{1}{\gamma \cdot n + (1-\gamma)}$

  but $T(\{i\}, [n]) \leq \frac{1}{n}$

  thus $\frac{1}{\delta} \frac{1}{\gamma \cdot n + (1-\gamma)} = \frac{1}{\delta} S_\gamma(\{i\}, [n]) \leq T(\{i\}, [n]) \leq \frac{1}{n}$

  hence $\frac{1}{\gamma + \frac{1-\gamma}{n}} = \frac{n}{\gamma \cdot n + (1-\gamma)} \leq \delta$

  $\liminf_{n \to \infty} \delta \geq \frac{1}{\gamma}$ QED

  i.e. the more n grows, the more δ approaches $\frac{1}{\gamma}$;

- Now we know that the distortion of S_γ is $\frac{1}{\gamma}$.

Example 5.8.4. Now we would like to apply the same method to the **cosine similarity** (a.k.a. *inner product similarity*).

- Let's start again with some definitions:
  - $U := \{\vec{x} \mid \vec{x} \in \mathbb{R}^n_+,\ ||\vec{x}||_2 = 1\}$ (i.e., U is a positive hypersphere with the center in the origin and the radius equal to 1),
  - $C \in U^2 \to [0, 1]$ s.t. $C(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle = \sum_{i=1}^{n} x_i y_i$;
- Now, we want to know if C has an LSH and, if not, what is its distortion;
- Another definition:
  $\mathcal{X} = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_n\}$ where $x_i(j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$
  i.e. $\vec{x}_i = (0, ..., 0, 1, 0, ..., 0)$ with the only 0 in position i;
- So we have that $||\vec{x}||_2 = \sqrt{1^2 + 0^2(n-1)} = 1$;
- Moreover $C(\vec{x_i}, \vec{x_j}) = 0\ \forall i \neq j$;
- Let $\vec{y} = \left(\underbrace{\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, ..., \frac{1}{\sqrt{n}}}_{\text{n times}} \right)$, then $||\vec{y}||_2 = \sqrt{\sum_{i=1}^{n} y_i^2} = \sqrt{\sum_{i=1}^{n} \frac{1}{n}} = 1$;
- So, $C(\vec{x_i}, \vec{y}) = \sum_{j=1}^{n} (x_i(j) \cdot y_i(j)) = x_i(i) \cdot y_i(i) = \frac{1}{\sqrt{n}}$;
- Then $\delta \geq \sqrt{n}$, so, unlike before, the distortion is big and grows bigger with n.

Example 5.8.5. **Weighed Jaccard** is a generalization of $\mathcal{J}acc$ for vectors:
$\mathcal{WJ} = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)}$.

Example 5.8.6. **Sim Hash** is an LSH scheme similar to the cosine similarity:

- $\mathcal{CS}(\vec{x}, \vec{y}) = \cos(\theta_{\vec{x},\vec{y}})$;
- $\mathcal{SH}(\vec{x}, \vec{y}) = 1 - \frac{\theta_{\vec{x},\vec{y}}}{\pi}$;
- $\mathcal{SH}$ is high if the angle θ is small;
- $\frac{\theta_{\vec{x},\vec{y}}}{\pi}$ is the probability that an hyperplane divides $\vec{x}$ and $\vec{y}$, where the hyperplane is a threshold between similar and dissimilar elements of the universe, so it creates a partition of the universe in two sets.

Chapter 6

181015

First lesson reprise - Flow of memes

spread of memes/viruses from a website to another

so, we get traces

goal is to reconstruct graph from traces

Trace spreading model (from virology)

source chosen UARand edge traversal time chosen by (usually) Exp(lambda) trace starts from source

obs. the first two nodes of a trace are connected

pr a source is chosen: 1/n pr the edge's other endpoint: 1/deg(source) because of how traces work

n numero di tracce: $pr = (1 - 1/deltan)^(3deltanlogn) = e^(- 3lnn) simeq 1/n^3$

6.0.1 another information spreading process

NPR chain letter

important observations: first name is fake, the others are plausibly real

asks to add own's name, and then forward to all friends

someone breaks the rules, and decides to publish their copy of the mail

those published mails reveal a subtree of the whole chainletter tree

revealed tree MUCH smaller, and extended in-depth

Let T be a tree, each node has a chance of being "exposed" by a probability p, and if so, reveals all ancestors

estimate the size of the tree

ALGO throw a coin on all the nodes of the chain tree, don't throw if node is parent of a revealed node (no need) := special nodes

special nodes are all iid from exposed ones; delta = n of special nodes/special nodes; revealed nodes = delta dot n (it's the expected value!!) -> from there estimate n

delta is the actual exposure probability

the two estimates can go wrong...

examples in nature suggest some bounds

Theorem: the algorithm correctly estimates n if $n > omega(max(1/delta^2, k/delta))$

IMPORTANT: Estimates can be redone, but reconstructing another revealed tree is impractical, so we're estimating a characteristic of a tree by observing a sample revealed one and making estimates

Single-child fraction

Fix a bound: a node's maximum degree must be lesser than k

partition unknown tree into subforests, each subforest has 1/delta nodes and median height $omega(log_(k-1)^d elta^-1)$

by exposing a node in a subforest's lower half, we're exposing a number of nodes equal to the sf's median height

Insight: δ is unknown too

Recap of revealed subtrees

Exposure prob: $\delta > 0$

$E(\#exposednodes) = \delta n$

Let $x \in RV$ be the set of nodes of the revealed tree. Let $Y \in Coin(p)$, where $Y_v = 1$ if v is exposed, 0 otherwise.

$E(Y_v) = \delta$

Let $Y = \sum_{v \in X} Y_v \Rightarrow E(Y) = \delta|X|$

OUTPUT: $\hat{\delta} = \dfrac{Y}{|X|}$

Chernoff bound is applicable, the Ys are IID

Note: we can do the first substitution because of linearity (?)

"Multiplicative approximation": $Pr\left\{|\hat{\delta} - \delta| \geq \varepsilon\delta\right\} = Pr\{(\} \, |\frac{Y}{|X|} - \delta| \geq \varepsilon\delta) = Pr\{(\} \, |Y - \delta|X|| \geq \varepsilon\delta|X|) = Pr\{(\} \, |Y - E(Y)| \geq \varepsilon E(Y)) \leq 2e^{-\frac{\varepsilon^2}{3}E(Y)}, \forall \varepsilon \in (0,1)$

This is the Multiplicative Chernoff Bound: Let y1 ... yn IID Coins, then (see above, last inequality)

we need E(y) to be large in order to obtain a useful bound, for E close to 1, the bound itself goes over 1, becoming useless

$2e^{-\frac{\varepsilon^2}{3}E(Y)} = 2e^{-\frac{\varepsilon^2}{3}\delta|X|}$

If $|X| \geq \dfrac{3}{\varepsilon^2\delta} \ln 2/\eta$, then $2e^{-\frac{\varepsilon^2}{3}\delta|X|} \leq 2e^{-\ln 2/\eta} = 2\eta/2 = \eta$

$Y' = \sum_{v \in T} Y_v \Rightarrow E(Y') = \delta n$

$Pr\{Chernoff\ on\ Y'\} \leq 2e^{-\frac{\varepsilon^2}{3}\delta n}$

Using the bound over $|X|$, then $\leq \eta$

Final output: $\frac{Y'}{\not{\theta}}(\leq \frac{(1+\varepsilon)\delta n}{(1-\varepsilon)\delta} = (1+O(\varepsilon))n)$

Lower bound inverts the sign of bigOepsilon

Insight: $\frac{Y'}{\not{\theta}} = \frac{Y'}{Y}|X|$

"An additive approximation is useless"

goodnes sof alogrithm:

- nodes of unknown tree less than delta: revealed tree =0 almost surely - tree is a star -> almost surely get a bad approximation

what characteristics should the unknown tree have?

Claim: If the number of internal nodes of the revealed tree is at least 3/(epsilonSQRdelta)ln(2/eta) then our guess n̂ is going to satisfy (1-oe)n <= n̂ <= (1+oe)n with probability at least 1-2eta

the three greek letters can change the goodness of our result

Next goal: find the probability that a node of the unknown tree will be an internal node of the revealed tree

Assumption: The unknown tree is such that none of its nodes has more than K children.

Let v be a node of the unknown tree with K children, then $Pr\{$at least 1 children in K is exposed$\} \geq 1-\varepsilon^{-1}min(1,\delta K_v)$

K_v: expected number of children revealed

Let $Z_v \in Coin(p)$ where it is 1 if at least one child of v is exposed

$E(Z_v) = 1-(1-\delta)^{K_v} = 1-((1-\delta)^{1/\delta})^{\delta}K_v$

$\lim_{\varepsilon\to 0^+}(1-\varepsilon)^{1/\varepsilon} = 1/e$

$1-((1-\delta)^{1/\delta})^{\delta}K_v \geq 1-e^{-\delta K_v}$

- if deltakv geq 1 then E(Zv) geq 1-1/e
- else , geometric intuition... ?????????? ... E(Zv) geq 1-ePOW(-deltakv) geq deltakv(1 - 1/e)

Lemma is proven

Former insight: for revelation we care about children, not all of the descendants

By summing all children of all nodes, we get all the edges -> n-1

Lemma: Let Z be the set of nodes of which at least 1 child is exposed; then $Pr\{|Z| \geq 1/2(1-1/e)\min(K^{-1},\delta)(n-1)\}$ $1-e^{-\Theta(n\min(1/K,\delta))}$

Proof: Let I be the set of internal nodes of T; Let D subset I contain nodes with at least 1/delta children; $E(|Z|) = \sum_{v \in I} E(Z_v) \geq (1 - 1/e) \min(1, K_v\delta) = (...)(|D| + \delta \sum_{v \in I-D} K_v)$

obs1: sumkvforvinI = n-1

obs2: $|D| \geq \frac{\sum_{v \in D} K_v}{|K|}$

then : $(1 - 1/e)(|D| + \delta \sum_{v \in I-D} K_v) \geq (1 - 1/e)(\sum K_v/K + \delta \sum_{v \in I-D} K_v) \geq (...)min(1/K, \delta) \sum = (...)min(...)(n-1)$

Proven

see emergency notes!!

6.1 181022

-combinatoric optimization-

graph independent set - impractical example

An implicit opt problem.

unknown - partly known function, exp in the codomain

ex: describe SAT by means of truth tables - rows are exponential in variable number

“Modular set functions”

given $X = [n] f \in 2^X \to \mathbb{R}$

$n = 3, w_1 = 1, w_2 = 5, w_3 = 1, f(S) = \sum_{i \in S}$

max set: all domain

max set w neg values: exclude neg values

se ho un oracolo che mi calcola f(s), ed io non conosco f, posso interrogare l’oracolo sui singoletti, e poi applico le logiche precedenti

(Matroid) Ex tutte parti con al più k elementi (a partire da U)

“Ad placement” arriva un utente sul sito tipo Google, scegli quali ad mostrare in modo da guadagnare il più possibile

modello Independent Cascade Model ogni ad rappresentata da una coppia $AD_i = (v_i, p_i)$ accordo tra pubblicità e Google, $v_i \in \$, p_i$ probobilità di click

$Ad_{samsung} : (1\$, 0.1) \Rightarrow E[X_i] = 1 \cdot 0.1$

Strategy: put highest expectation ads on top

third value,: satisfaction factor (or the will to look other ads)

how to optimize f in ordder to maximize it

caso particolare “submodular set function” - analogo discreto delle funzioni convesse - can be misleading

given X, f is submodular iff $\forall S, T \in X, f(S) + f(T) \geq f(S \cap T) + f(S \cup T)$

modular is submodular: $\forall S, T \in X, f(S) + f(T) = f(S \cap T) + f(S \cup T)$

quanti bit servono per rappresentare tale funzione?

LAPSE

COVERAGE (function, submodular): $X = \{S_1, S_2 ... S_m\}, S_i \subseteq [n]$

$Y \subseteq X : c(Y) = |\bigcup_{S_i \in Y} S_i|$

$s(Y) = \sum_{S_i \in Y} |S_i|$

proof that f (or c?) is submodular base case n=1: couple S T can assume 4 combinations: $cases, t = \{1\} \Rightarrow unionanintersectionf(\cdot) = 11inT, 1notinSthen1inunion, 1notinintersectiondualisduals, t = \emptyset$

induction: split a coverage function into two coverage functions: $c(Y) = |\bigcup_{S_i \in Y} S_i| = |(\bigcup_{S_i \in Y} S_i) - \{n+1\}| + |(\bigcup_{S_i \in Y} S_i) \cap \{n+1\}|$

Nemauser-W

If f is submodular nonnegative and monotone, it can be approx to $max_{\{}|S| = K\}(f(S))$ to a factor of $1 - \frac{1}{e}$

represent modulars as points of convex multidimensional functions

“maxcover” data una classe di sottoinsiemi di X,. S-1, si trovino k insiemiinX tail che , insieme, coprano il massinmo numero di elementi

Before proving, ALGORITHM (greedy), parameterized by f, X and k: trovare k sottoinsiemi di X che massimizzano f

$S_0 < -\emptyset fori = 1- > kletx_i = maximizeroff(\{x_i\} \cup S_{i-1})S_i < -\{x_i\} \cup S_{i-1}returnS_k$

again before proving, observation: diminishing returns (formalization)

$\delta(x|S) = f(\{x\} \cup S) - f(S)$

let f be submodular, $B \subseteq A, x \notin B \Rightarrow \Delta(x|A) \geq \Delta(x|B)$

submodular functions exhibit diminishing returns property

Now: $S = A \cup \{x\}, T = B \Rightarrow f(A \cup \{x\}) + f(B) \geq f(A) + f(B \cup \{x\}) \Rightarrow f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B) \Rightarrow \Delta(x|A) \geq \Delta(x|B)$

proof of approximation:

take an optimal solution $S* = \{x_1^*, ..., x_k^*\}$

$\forall i, f(S^*) \leq f(S^* \cup S_i)$

$= ...f(S_i) + \sum_{j=1}^{k} \Delta(x_j^*|S_i \cup \{x_1^*, ..., x_{j-1}^*\}) \leq (bydiminishingreturns) f(S_i) + \sum_{j=1}^{k} \Delta(x_j^*|S_i) - > \leq (bygreediness) f(S_i) +$

$\sum_{j=1}^{k} \Delta(x_{i+1}|S_i) = f(S_i) + k(f(S_i \cup \{x_{i+1}\}) - f(S_i)) = kf(S_{i+1}) - (k-1)f(S_i)$

So: $f(S^*) - f(S_i) \leq k(f(S_{i+1} - f(S_i)))$

Def: $\delta_i = f(S^*) - f(S_i)$

Then: $\delta_i \leq k(\delta_i - \delta i + 1) \Rightarrow k\delta_{i+1} \leq (k-1)\delta_i \Rightarrow \delta_{i+1} \leq (1 - \frac{1}{k})\delta_i$

$\delta_0 = f(S^*) - f(S_0) \leq f(S^*)$

. . .

going to definition of e by its limit form $(1 - 1/k)^k$

6.2 181024

recap of yesterday:

given a ground set, and $f \in 2^X \rightarrow$, then f is modular iff $\forall S, T \subseteq X(f(S) + f(T) \geq f(s \cup t) + f(s \cap t))$

E.g.: coverage functions, X is a set system $c(X) = \left| \bigcup_{S \in X} S \right|$

"given a subset of sets, the coverage value would be the cardinality of the union"

claim: coverage function is submodular proof: $c(S) + c(T) \geq c(s \cup t) + c(s \cap t)$

proven case-by-case in the scope of where the elements are inside the subsets

if $f_1, ..., f_k \in 2^X \rightarrow \mathbb{R} are submodular, \alpha_1, ..., \alpha_k \geq 0 \Rightarrow \sum_{i=1}^{k} \alpha_i f_i is submodular$

proof: the i-th case is easily proven, then sum up; nonnegativity is essential here

algorithm for optimization (?): $GREEDY_f(X, K)$: $S_0 < -\emptyset for i = 1 - > k let x_i = maximizer of f(\{x_i\} \cup S_{i-1}) S_i < -\{x_i\} \cup S_{i-1} return S_k$

if f is submodular, nonnegative and monotone, then $f(S_K) \geq (1 - 1/e) max_{S \in \binom{X}{K}} f(S)$

another algorithm: $GREEDY_{\varepsilon,f}(X, K)$: $S_0 < -\emptyset for i = 1 - > k let x_i be : f(S_{i-1} \cup \{x_i\}) - f(S_{i-1}) \geq (1 - \varepsilon) max_{x \in X}(f(S_{i-1} \cup \{x\}) - f(S_{i-1})) S_i < -\{x_i\} \cup S_{i-1} return S_k$

the difference is: we don't need to find the exact maximizer, but something (1-epsilon) close to it; and it may be MUCH easier to find such something instead of the actual maximizer.

the core of the discussion is approximating a (usually impracticable to compute) function to a (poly-

nomial) one

Tidbit: proving correctness of this algorithm proves the previous one

Trail: KKT model

Hint(!): We don't need to find the maximum in this algorithm (otherwise, we are actually in the former case, why not use it directly?)

Theorem: $(\forall \varepsilon \in [0,1))$, if f is submodular, nonnegative and monotone, then $f(S_k) \geq (1-\varepsilon)(1-1/e) max_{S \in \binom{X}{K}} f(S)$

Proof: $S^* = argmax\ f(S), S \in \binom{X}{K}$

$\forall i \in \mathbb{k}$

$$f(S^*) \leq f(S_i) + \sum_{j=1}^{k} (f(S_i \cup \{x_1^*, ..., x_j^*\}) - f(S_i \cup \{x_1^*, ..., x_{j-1}^*\}))$$

[defining diminishing returns (or marginal increase of 1 having 2)]

$$\rightarrow = f(S_i) + \sum_{j=1}^{k} \Delta(x_j^* | S_i \cup \{x_1^*, ..., x_{j-1}^*\})$$

Because of submodularity, $\rightarrow \leq \sum_{j=1}^{k} \Delta(x_j^* | S_i)$

-brainlapse-

...at the end of the day $f(S^*) - f(S_i) \leq \frac{k}{1-\varepsilon} \Delta(x_{i+1} | S_i)$

def $\delta_i = f(S^*) - f(S_i)$

$\delta_{i+1} \leq \delta_i (1 - \frac{1-\varepsilon}{k})$

-SIGSEGV-

see previous day to get all deltas from 0 to k

then $f(S_k) \geq (1 - (1 - \frac{1-\varepsilon}{k})^k) f(S^*) = (1 - ((1 - \frac{1-\varepsilon}{k})^{\frac{k}{1-\varepsilon}})^{1-\varepsilon}) f(S^*) \geq (1 - (1/e)^{1-\varepsilon}) f(S^*)$

more algebretta, use exponential convexity, get $f(S_k) \geq (1 - e^{\varepsilon - 1}) f(S^*)$, over and out

Application: Kempe-Kleinberg-Tardos

Chapter 7

181105

Recall: densest subgraph (with maximum mean degree)

Technique: linear programming

Linear program: Primal

- Objective:

#Continued onto notes...