Uniroma1(2018-19) - Social and Behavioural Networks

Andrea Proietto

October 2018

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 Basic ideas

Note: "pick" and "sample" are used interchangeably.

About $E(X)$: It is called the "expected value" because it means the midpoint where all outcomes, compounded with their weight, contribute to the equilibrium.

Question: 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* $\mathbb{N}$*?*

2.1.1 Sampling

Sampling is a simple algorithmic approach used for abstracting and estimating the percentage of elements of a certain type in a large set.

Example: 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.

2.1.2 Union bound

$$\Pr\left(\bigcup_{i=0}^{n} E_i\right) \leq \sum_{i=0}^{n}(\Pr(E_i)) \tag{2.1}$$

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

2.1.3 Markov inequality

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

$$\Pr(X \geq a) \leq \frac{E[X]}{a} \tag{2.2}$$

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.

2.1.4 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

$$M_X(t) = e^{tX}, t \in \mathbb{R} \tag{2.3}$$

To be continued...

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

2.2 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: $X \in \mathcal{B}inom(2^n, p)$, 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 define the Chernoff bound as:

$$\Pr(|X - E[X]| \geq t) \leq e^{-2t^2/n} \tag{2.4}$$

where $t := \varepsilon n$.

In the binomial case, [2.4] 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

$$\Pr\left(\left|\frac{1}{n}\sum_{i=1}^{n}(X_i) - p\right| > \varepsilon\right) \leq 2e^{-2n^3\varepsilon^2} \tag{2.5}$$

Example: 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:

$$\Pr(|X - E[X]| \geq \varepsilon \cdot n) \leq 2e^{-\frac{\varepsilon^2}{3}n} \tag{2.6}$$

Example: 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 3

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:

- **Jaccard similarity**: Given two sets of objects A and B, their Jaccard similarity is defined as follows:

$$\mathcal{J}acc(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

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

$$\mathcal{H}amm(A,B) = \frac{|A \cap B| + |\complement_U(A \cup B)|}{|U|} \tag{3.2}$$

3.1 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.

An observation to make is the following: 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:

$$\mathcal{J}acc(distinct(T_1), distinct(T_2)) = \frac{3}{5} \tag{3.3}$$

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 3.1: 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 defined the hashing function as:

$$h_\pi \in \mathcal{P}(U) \to U : h_\pi(A) = min_\pi(A) \tag{3.4}$$

In other words, we take the "minimum" in A according to the ordering specified by π. A simple but useful observation would be:

$$\forall A \subseteq U \Rightarrow h_\pi(A) \in A \tag{3.5}$$

Example:

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

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 $X_{1\ldots n} \in \mathcal{B}er(p)$ 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:

$$\Pr\left(|avg_{i=1}^{n}(X_i - p)| \geq \varepsilon\right) \leq 2e^{-n\varepsilon^2} \tag{3.6}$$

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 [3.1] if efficient: instead of comparing two entire pages, it only compares n integers.

3.2 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: 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.

3.3 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:

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

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

Example: We can apply the min-hash scheme [3.1] 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: Similarly, we can apply the function based on the Hamming Similarity we saw in [3.2] to the same set $U = [3]$ and see we get new partitions. We chhose $i = 2$ UAR, the function maps each subset of U to a hash as follows:

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

thus defining 2 partitions over U.

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

Observation: 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 $\Pr(H(a) = H(b)) = S(a,b) \forall a, bin U$

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

Observation: In the previous webpage example [3.1], 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

$$\forall A, B \in U \Rightarrow \Pr_p(A \sim_p B) = S(A,B) = \Pr_h(h(A) = h(B)) \tag{3.7}$$

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:

$$X \in \mathcal{R}and(\mathcal{P}(U)) : \forall A, B \in U \implies \mathcal{E}(A \sim_X B) = S(A,B) \tag{3.8}$$

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

Example: 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.

3.4 More distances

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

Dice similarity:

$$D(A, B) = \frac{|A \cap B|}{|A \cap B| + \frac{1}{2}|A \triangle B|} \tag{3.9}$$

Anderberg similarity:

$$An(A, B) = \frac{|A \cap B|}{|A \cap B| + 2|A \triangle B|} \tag{3.10}$$

Generalizing Jaccard, Dice and Anderberg:

$$S_\gamma(A, B) = \frac{|A \cap B|}{|A \cap B| + \gamma|A \triangle B|} \tag{3.11}$$

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: (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 [3.3]):

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

Corollary: 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: 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.

3.5 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:

$\mathcal{G}en_X(\alpha) = \sum_{x=0}^{|\Omega|} \Pr(X = x)\alpha^x$ (note that all outcomes appear by their probability)

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

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

Afterthought: Are we "applying" a probability distribution over a similarity (which in turn, since it is lshable, means it has a probdist over a subset of hashfunctions)?

Proof: see below

Consequence: Applying the theorem to the Jaccard similarity:

Start with $f_\gamma(\alpha) = \sum_{x=1}^{\infty} \frac{(1-\frac{1}{\gamma})^x}{\gamma - 1}\alpha^x$

We have to demonstrate that the coefficients represent a probability distribution: $\sum_{i=1}^{\infty} \frac{(1-\frac{1}{\gamma})^i}{\gamma - 1} = 1 \Rightarrow \sum_{i=1}^{\infty}(1 - \frac{1}{\gamma})^i = \gamma - 1$

Now: $\sum_{i=1}^{\infty}(1-\frac{1}{\gamma})^i = (1-\frac{1}{\gamma})\sum_{i=0}^{\infty}(1-\frac{1}{\gamma})^i = (1-\frac{1}{\gamma})\frac{1}{1-(1-\frac{1}{\gamma})} = \frac{\gamma-1}{\gamma}\frac{1}{1/\gamma} = \gamma - 1$

Apply PGF to Jaccard: $f_\gamma(J(A,B)) = ...\text{algebretta insiemistica}... = \frac{\cap}{\cap + \gamma\ \triangle} = S_\gamma(A,B)$, which in turn is LSH-able

3.6 181008

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

$$\forall X, Y \in U \quad \Pr_h(h(X) = h(Y)) = S(X,Y) \tag{3.12}$$

Reprise: If a similarity S is LSHable and f is a PGF, then f(S) is LSHable

Equivalent statement:

$$f(S) := T \in U^2 \to [0,1] : \forall A, B \in U \quad T(A,B) = f(S(A,B)) \tag{3.13}$$

Lemma (1): The similarity $O \in U^2 \to [0,1] : (A,B) \mapsto 1$ admits a LSH Proof: Give prob. 1 to a constant function (duh!): $h \in U \to \mathbb{1}A \mapsto 0 \forall A in U$

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

Lemma(2): 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

first, sample h_S for S; then sample h_T independently for T return the function $h : A \mapsto (h_S(A), h_T(A))$

$\Pr_h(h(A) = h(B)) = \Pr_{h_S}(h_S(A) = h_S(B)) \cdot \Pr_{h_T}(h_T(A) = h_T(B)) = S(A,B) \cdot T(A,B)(\forall \{A,B\} \in \mathcal{P}_2(U))$by independency

Lemma(3): if S is LSHable then $\forall i \in \mathbb{N}$ S^i is lshable

proof by induction:

base (Lemma1): i=0 and S^0=O OK

ind: Use lemma 2 on S^i and S to obtain S^{i+1}; S has a scheme by def, S^i has a scheme by induction hypothesis

lemma(4): if $p_0, ..., p_i, ... : \sum_{i=0}^{\infty} p_i = 1, and p_i \geq 0 \forall i$, and $S_0, ..., S_i, ...$ are lshable similarities, then $\sum_{i=0}^{\infty} p_i S_i$ is lshable

scheme: first pick(sample, they are synonyms) i* at random from $\mathbb{N}$ with probability $p_0, ..., p_i, ...$ then, sample h from the hash functions of S_{i*}

$\Pr(h(A) = h(B)) = \sum_{i=0}^{\infty} (p_i S_i(A,B))$

$\Pr(h(A) = h(B)) = \sum_{i=0}^{\infty} (\Pr(i = i*) \Pr_h(h(A) = h(B) | i = i*))$, $\Pr(i = i*) = p_i$, $hahb|i = i* \rightarrow S_i(A,B)$

Final proof: sum of $p_i S^i$ has a scheme

L3: S^i has a scheme

L4: the sum is lshable, proven

Example of a pgf: $sum(2^-1 times x^i)$

---??sorensen dice??cosine similarity?? inner product??johnson-lindenstrauss??

a sketch is n instance of a pgf which can be used to implement other similarities NONONONONOO

PGF is an approach for making schemes for similarities from other schemes

3.7 181010

Let f a PGF, $\alpha \in [0,1]$, ...?

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

$T \in U^2 \to [0,1] : \forall t, t' \in \mathcal{P}_2(U) T(t,t') = 0$

for the 1 case we wanted a nbanal 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

GOTO

Approximation examples

Consider S_γ: $\gamma \geq 1 \Leftrightarrow S_\gamma$ is LSH-able

Focus on $\gamma < 1$

Definition: "Distortion of a similarity"

Let S be a similarity, then its distortion is "the minimum*(meaning inferior extremum) delta geq 1 : exists LSHABLE S' (forall A, B in P2U, $1/\delta \cdot S(A,B) \leq S'(A,B) \leq S(A,B)$)"

delta tends to 1 -> S is lshable

using jaccard to approximate sgamma we wold obtain 1/gamma

delta is the approximation factor

Centerlemma: Let S be a LSHable similarity : $\exists \chi \subseteq U : \forall \{x, x'\} \in \mathcal{P}_2(\chi) S(x,x') = 0$, then

$\forall y \in U avg_{x \in \chi}(S(X,Y) \leq) \frac{1}{|\chi|}$; (it trivially follows that $\exists x* \in \chi : S(x*, Y \leq \frac{1}{|\chi|})$)

Proof: (Fix $y \in U$) If the hash function h has positive probability in the (chosen) lsh for S, then $\forall x, x' \in \mathcal{P}_2(\chi)(h(x) \neq h(x'))$

[χ is actually a (possibly incomplete) section of the partition of U induced

by h]

thus, forall hash functions with positive probability, there can exist at most one x in calx st hx=hy (transitivity of equality)

$$\sum_{x\in\chi} S(x,y) = \sum_{x\in\chi} \Pr_h(h(x)=h(y)) = \sum_{x\in\chi}\sum_h \Pr(h\ is\ chosen)[h(x)=h(y)]$$

IMPORTANT: The brackets here are a boolean evaluation operator

$$= \sum prhischosen \sum evalhx = hy \leq \sum_h \Pr(h\ is\ chosen) \leq 1$$

Thus, $\sum_{x\in\chi} S(x,y) \leq 1 \Rightarrow avg(S(X,Y)) \leq \frac{1}{|\chi|}$, proven

$S := S_\gamma, 0 < \gamma < 1, U = 2^{[n]} = S|S \subseteq [n]$

$\chi := \mathcal{P}_1([n])\ y = [n]$

- let us assume that T finitely distorts sgamma, and T is lshable then $T(,) = 0 foralli, jinP_2(mathbbm(n))$

- $\exists\{i\}in\chi : T(\{i\},[n]) \leq \frac{1}{|\chi|} = \frac{1}{n}$

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

... zenterlemma application ...

$U =$

Chapter 4

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

4.0.1 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: $\hat{\delta} = \frac{Y}{|X|}$

Chernoff bound is applicable, the Ys are IID

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

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

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 $|X| \geq \frac{3}{\varepsilon^2\delta}\ln 2/\eta$, 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: $\frac{Y'}{\cancel{\delta}}(\leq \frac{(1+\varepsilon)\delta n}{(1-\varepsilon)\delta} = (1 + O(\varepsilon))n)$

Lower bound inverts the sign of bigOepsilon

Insight: $\frac{Y'}{\cancel{\delta}} = \frac{Y'}{Y}|X|$

"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 $\Pr(|Z| \geq 1/2(1-1/e)\min(K^{-1}, \delta)(n-1)) \geq 1 - e^{-\Theta(n \min(1/K, \delta))}$

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!!

4.1 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: $cases, t = \{1\} \Rightarrow unionanintersection f(\cdot) = 11 in T, 1 not in S then 1 in union, 1 not in intersection d \emptyset$

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 $(1-1/k)^k$

4.2 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 (polynomial) 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 5

181105

Recall: densest subgraph (with maximum mean degree)

Technique: linear programming

Linear program: Primal

- Objective:

#Continued onto notes...