Approximate Privacy-Preserving Neighbourhood Estimations

ALVARO GARCIA-RECUERO
alvaro.garcia@imdea.org

IMDEA NETWORKS - DTG

July 1, 2021

Table of contents

Motivation

1. General Data Protection Regulation (GDPR) in Europe with the Data Protection Act 2018 for UK implemented, California Consumer Privacy Act 2018 (CCPA) in USA, and all sort of regulations that limit privacy footprint of users on big data platforms.
2. The systems research community (and industry, e.g., Google) is moving to federated systems (secure?) for semi-decentralised privacy-preserving learning and AI.
3. Can we try to approximate the manner in which we compute statistics and metrics over complex networks or graphs with a limited privacy impact and data minimisation?
4. Applications: Data Provenance, Approximate Federated Learning, etc.

Hypothesis and Use Cases

Re-purpose the use of the HyperANF a.k.a HyperBall algorithm,intended for approximate diameter estimation, to the task of **privacy-preserving** "'community detection"' or "'friend recommending systems"' that learn from an approximated, minimised and anonymous fingerprint of the social network graph structure with limited privacy impact.

For example, a user may want to compute locally some network metric estimation without global knowledge of the graph.

Background

- **HyperLoLog**: Flajolet invented this algorithm afaik, he passed away but the algorithm is great.

- **HyperBall**: Uses HyperLogLog, written by Boldi, Paolo and Rosa, Marco and Vigna, Sebastiano in the "'Hyperanf: Approximating the neighbourhood function of very large graphs on a budget. In Proceedings of the 20th international conference on World wide web(2011), ACM, pp. 625–634"'

- **Degrees of Separation**: Performing data analysis over large graphs with billions of edges is an important yet challenging task. Previous work has estimated the average distance in the graph between any two users of large social networking sites such as Facebook, resulting in 4.75 hops on average by Boldi in 2012.

Theory HLL

Table: Parameters of HLL

$h : D \rightarrow 2^b$	a hash function from the domain of items
M	an array of $m = 2^t$ counters each initialised to $-\infty$
α_m	a constant that depends on the number of counters

HyperBall Algo

$c[-]$: ▷ an array of n HLL counters.

function UNION(M: counter, N: counter)
    for i ¡ p **do**
        $\{M[i] \leftarrow \max M[i], N[i]\ \}$

$r \leftarrow 0$

function GETBALL(c: counter)
    repeat
        for v $\in$ V **do** $a \leftarrow c[v]$
            for w $\in$ N(v) **do**
                $a \leftarrow UNION(c[w], a)$

▷ write tuple (v, a) to disk, which estimates $|B_r + 1(v)|$.

▷ Update the array $c[-]$ with the new (v, a) pairs.

        $r \leftarrow r + 1$
    until no counter changes its value
    return GetCount(c)

for v $\in$ V **do** ▷ Initialisation
    function ADDITEM$(\{\}c[v], v\}$

$|\hat{B}_r|_{(r>1)} =$ GETBALL(c: counter) ▷ Use it

Algo for average path length using HyperBall

```
nr˙paths:                    ▷ number of paths of length per node.
max_t: ▷ max. distance of HyperBall computations is equal to b.

function HyperBall(G:graph, b:radius of ball, p:hll_c_prec)
    return HB
function NumNodesDistFrom(v, t)
    if t = 0 then
        return 1
    else if t >= get_max_t then
        return 0
    else
        return balls[v][t].size() - self.balls[v][t - 1].size()
function average_path_length(G)
    for v ∈ G do
        for t ∈ 1..max_t do
            nr˙paths[v] = HB.NumNodesDistFrom(v, t)
```

Results

Table: HyperBall computation times in (hh:mins:secs)

			Bfs	HyperBall	
	Nodes	**Edges**	**Sequential**	**Sequential**	**Parallel**
Twitter	81306	2420766	>1:00:00.00	0:02:54.874483	0:02:46.414789
Facebook	4039	88234	0:00:56.965906	0:00:08.249678	0:00:08.533745
Mastodon	566520	6493563	>1:00:00.00	0:11:16.773455	**0:05:29.926316**

Results

Table: Time difference for Path Length with NetworkX vs HyperBall computing vs approximating three network metrics.

			NetworkX	HyperBall			
	Nodes	**Edges**	**hh:mins:secs**	**(hh:mins:secs)**	**Avg. Shortest Path Length (R/G)**	**Clustering Coeff.(G/L)**	**Small World Coeff.(l - c)**
Twitter	81306	2420766	0:03:21.698368	0:03:34.580287	0.9072928958767731	0.565311468612065	0.3415490228344126
Facebook	4039	88234	0:00:12.997504	**0:00:09.321404**	0.8024902838225895	1.0171284634760704	-0.3015614725029204
Mastodon	566520	6493563	0:15:28.511020	0:19:39.377784	1.11599944828774786	≈ 0.0	1.1159994828774786

Approximating intersections of HyperBalls

- Existing approaches can obtain intersections by computing the product of the *HLL* with an approximation of the Jaccard similarity using MinHashes.
- Because MinHashes is approximately the intersection of the Jaccard among two sets divided by a fixed parameter k (see equation 1), we are left with just an approximated value of the intersection among any two sets. So Looking at equation 2., can we do the same for two HyperBalls (ball cardinality of each node)? hint: look at the array pairs $< v, a >$ in HyperBall algo.

$$\frac{|h_k(A_i) \cap h_k(B_i)|}{k} \tag{1}$$

$$\left|\bigcap A_i\right| = J(A_1, \ldots, A_n) \cdot \left|\bigcup A_i\right| \approx \mathrm{MinHash} \cdot \mathrm{HLL} \tag{2}$$

Future Work

- Partition graph in N connected sub-graphs of size $\leq$ k randomly, which resembles local neighbourhood of users in the graph.
- Run the Average path length with HyperBall to compare the result in each of the sub-graphs of size $\mathbf{N_k}$, e.g., 500, 1000, 10000.
- Re-sample varying size of hyperball to compare if higher values increase similarity of average path length among sub-graphs (expected) with the trade-off of higher performance overhead. Normally, calculating over smaller sub-graphs of hashed items should be less accurate due to HLL size needs, but faster due to being embarrassingly parallel.
- Result should be so that, higher values of radius of the hyperball should show a higher cardinality of hashes to be computed in each hyperball step, thus approximating better the resulting value through our approach of Jaccard together with Min-Hashes.

Discussion and Ideas

Paper preprint, in progress: https://arxiv.org/abs/2102.12610
Code: https://github.com/algarecu/ppanf