Estimating Graphlet Statistics via Lifting

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

1 Introduction

Applications, references, problems.

2 Problem statement and frameworks

Consider graph $G = (V, E)$ with set of vertices V and set of edges $E \subseteq \binom{V}{2}$. We assume G to be simple, connected and undirected. (It's easy to translate our results to the case of directed graphs, since we will mostly be focusing on sampling procedure.) For a subset $W \subseteq V$, a subgraph of G induced by W is a graph with vertices in W and edges in $\binom{W}{2} \cap E$. Notation: $G|W$.

If $|W| = k$ and $G|W$ is connected, then we say that $G|W$ is a k-graphlet of G. The set of all k-graphlets of G is denoted by $\mathcal{V}_k(G)$ (or simply $\mathcal{V}_k$). Given a connected graph H on k nodes, we'll be interested in the number of k-graphlets of G isomorphic to H.

We'll make use of Python notation for lists vs sets: if we don't need indices or respective ordering of elements, we'll write it as a set $\{v_1, \ldots, v_k\}$, and if the order of the elements is fixed or indices are used to distinguish the elements, we'll write it as a list $[v_1, \ldots, v_k]$.

Let $Gr_k = [H_1, H_2, \ldots, H_l]$ be the list of all non-isomorphic connected graphs on k vertices. For $T \in \mathcal{V}_k(G)$, we say that "T is graphlet of type m" if T is isomorphic to H_m. Notation: $T \sim m$.

The number of k-graphlets of G of type m is equal to $N_m(G) = \sum_{T \in \mathcal{V}_k(G)} \mathbb{1}(T \sim m)$, where $\mathbb{1}(A)$ is the indicator function of boolean A.

The exact calculation of that sum is quite computationally expensive, since iterating over all connected k-graphlets of G takes $O\left(\binom{|V|}{k}\right)$ operations in the worst-case scenario. (See exact algorithms in [this paper] and [this paper]). Here, we'll be focusing on more modest goal of estimating the total graphlet count by sampling graphlets.

A common approach in that framework is to sample k-graphlets $T_1, T_2, \ldots, T_n$ from known probability distribution $\pi_k(T)$, and to notice that

$$\hat{N}_m(G) := \frac{1}{n} \sum_{i=1}^{n} \frac{\mathbb{1}(T_i \sim m)}{\pi_k(T_i)} \approx \mathbb{E}_{\pi_k} \frac{\mathbb{1}(T \sim m)}{\pi_k(T)} = \sum_{T \in \mathcal{V}_k(G)} \mathbb{1}(T \sim m) = N_m(G).$$

This approach gives an unbiased estimator assuming $\pi_k(T) > 0$ for any $T \in \mathcal{V}_k$. The biggest issue of this method is the variance of $\hat{N}_m(G)$.

$$\mathrm{Var}(\hat{N}_m(G)) = \frac{1}{n}\mathbb{E}_{\pi_k}\left(\frac{\mathbb{1}(T \sim m)}{\pi_k(T)}\right)^2 + \frac{1}{n^2}\sum_{i \neq j}\mathbb{E}_{\pi_k}\left(\frac{\mathbb{1}(T_i \sim m)\mathbb{1}(T_j \sim m)}{\pi_k(T_i)\pi_k(T_j)}\right) - N_m(G)^2. \quad (1)$$

The first term of the sum on the right-hand side is equal to $\sum_{T \in \mathcal{V}_k(G)} \frac{\mathbb{1}(T \sim m)}{\pi_k(T)}$ and can blow up if $\pi_k(T)$ is much smaller then the uniform probability $\rho(T) = \frac{1}{|\mathcal{V}_k|}$. The second term can blow up when graphlets T_i and T_j have been sampled with high correlation, which is the case for Markov Chain methods discussed below.

The previous papers on that topic focused on efficient method of sampling a sequence of k-graphlets, depending on a framework. We'll try to summarize the frameworks and corresponding methods here.

There are two different frameworks, each with its own problem statement.

- **Framework 1.** We have access to the whole graph structure, for example, in the form of adjacency matrix. We can sample the nodes of the graph cheaply, independently and according to an arbitrary probability distribution. The goal is to sample graphlets quickly, uniformly and independently. The state-of-art method for this framework is Color Coding ([reference]), among others ([many references]).
- **Framework 2.** We start with an access to a single node v_0 and its neighborhood. As a query, we can transition to a neighboring node v_1 and get access to the neighborhood of v_1. Each moment of time we can only get access to the neighbors of previously visited nodes. This framework is applied when queries are either expensive or limited and traversing the whole graph is time-prohibitive at best and not feasible at worst. The goal here is to achieve best accuracy with either limited number of queries ($10^3 - 10^4$ queries for graphs with $10^5 - 10^6$ nodes) or limited time, or something in between (costly queries). State-of-art methods use Monte Carlo Markov Chain (MCMC) sampling (see [reference], [reference] and [another reference]).

For each of the two frameworks above we'll present a modification of the base algorithm to suit corresponding goals.

3 Related Algorithms

Subgraph Random Walk (SRW)
Pairwise Subgraph Random Walk (PSRW)
SRWd
Wadling Random Walk (WRW)
Others

4 Algorithm Description

4.1 Lifting procedure

For a subset of vertices $S \subseteq V_G$, denote $\mathcal{N}_v(S)$ (vertex neighborhood of S) to be the set of all vertices adjacent to some vertex in S minus the set S itself. Denote $\mathcal{N}_e(S)$ (edge neighborhood of S) as the set of all edges that connect vertex from S to vertex outside of S. Formally,

$$\mathcal{N}_v(S) = \{u \in V_G \setminus S \mid \exists s \in S : (s,u) \in E_G\}, \quad \mathcal{N}_e(S) = \{(s,u) \in E_G \mid s \in S, u \notin S\}. \quad (2)$$

Denote the inner degree $\deg_{inner}(S)$ to be the number of edges in the induced graph $G|S$ multiplied by 2, and outer degree $\deg_{outer}(S)$ to be the number of edges in $\mathcal{N}_e(S)$. Formally,

$$\deg_{inner}(S) = 2 \cdot |\{(s_1, s_2) \in E_G \mid s_1, s_2 \in S\}|, \quad \deg_{outer}(S) = |\{(s, u) \in E_G \mid s \in S, u \notin S\}|. \quad (3)$$

Note that $\deg_{outer}(S) + \deg_{inner}(S) = \sum_{v \in S} \deg(v)$.

Lastly, for a vertex $u \in V_G \setminus S$ denote the u-degree $\deg_u(S)$ as the number of vertices from S connected to u:

$$\deg_u(S) = |\{s \in S \mid (u, s) \in E_G\}|.$$

First assume we have a node v_1 sampled from the distribution π_v (we'll address this assumption later). We also assume that $\pi_v(v) = \frac{f(\deg(v))}{K}$, where $f(x)$ is some function (usually a polynomial) and K is some global normalizing constant. We'll use $\pi_v(v)$ and $\pi_v(\deg(v))$ interchangeably.

To start our procedure, sample an edge (v_1, v_2) uniformly from $\mathcal{N}_e(\{v_1\})$. The vertex v_2 is then attached to the set $\{v_1\}$, after which we sample another edge, (v_i, v_3) (with $1 \le i \le 2$) uniformly from $\mathcal{N}_e(\{v_1, v_2\})$. Generally, sample an edge (v_i, v_{r+1}) (with $1 \le i \le r$) from $\mathcal{N}_e(\{v_1, \dots, v_r\})$ uniformly at random, and attach the vertex v_{r+1} to the set $\{v_1, \dots, v_r\}$. After $k-1$ operations, we obtain a sampled k-graphlet $G|\{v_1, \dots, v_k\}$.

We'll refer to the procedure above as the **lifting** procedure starting at vertex v_1.

For starters, we can compute the probability $\pi_k(T)$ of getting a particular k-graphlet $T \in \mathcal{V}_k(G)$ in that procedure. This can be done either recursively or directly. Denote the set of vertices of T as $V_T = \{v_1, \dots, v_k\}$.

Recursive formula starts with known initial probabilities $\pi_1(\{v_i\}) = \pi_v(v_i)$. Given an $r+1$-subraphlet S with $S \subseteq T$, the probability of S can be calculated as

$$\pi_{r+1}(S) = \sum_{R \subset S} \pi_r(R) \frac{\deg_{S \setminus R}(R)}{|\mathcal{N}_e(R)|} = \sum_{R \subset S} \pi_r(R) \frac{\deg_{S \setminus R}(R)}{\sum_{u \in R} \deg(u) - \deg_{inner}(R)},$$

where the sum is taken over all r-subgraphlets of S.

To write a direct formula, we need a specific labelling of vertices of T. We'll say list $[v_1, \dots, v_k]$ is *T-compatible labelling* if $T|\{v_1, \dots, v_r\}$ is a connected subgraph for any $1 \le r \le k$. Denote the set of all T-compatible labellings as $cL(T)$. Then

$$\pi_k(T) = \sum_{cL(T)} \pi_v(v_1) \prod_{r=1}^{k-1} \frac{\deg_{v_{r+1}}(\{v_1, \dots, v_r\})}{\sum_{i=1}^{r} \deg(v_i) - \deg_{inner}(\{v_1, \dots, v_r\})}$$

Although calculation of that probability on-the-fly is cost-prohibitive, we can greatly reduce the number of operations by noticing that the probability $\pi_k(T)$ is a function of degrees of the vertices: for the labelled list $V_T = [v_1, \dots, v_k]$ with $d_i = \deg(v_i)$, the probability is $\pi_k(T) = \frac{1}{K} F_T(d_1, \dots, d_k)$ for some function F_T.

Moreover, for two isomorphic graphlets $T_1 \sim T_2$ (with labelling of nodes in T_2 induced by the isomorphism), the functions are identical: $F_{T_1} = F_{T_2}$. Thus, it is enough to find functions F_{H_i} for each isomorphism class with $Gr_k = [H_1, H_2, \dots, H_l]$ being the fixed list of all non-isomorphic connected graphs of size k.

Example. Consider a triangle, which is a 3-graphlet with edges (v_1, v_2), (v_2, v_3) and (v_1, v_3). Given the degrees d_1, d_2, d_3 of the corresponding vertices, the probability function is

$$\pi_3(\text{triangle}) = \left(\frac{\pi_v(d_1)}{d_1} + \frac{\pi_v(d_2)}{d_2} \right) \frac{2}{d_1 + d_2 - 2} + \\ + \left(\frac{\pi_v(d_2)}{d_2} + \frac{\pi_v(d_3)}{d_3} \right) \frac{2}{d_2 + d_3 - 2} + \left(\frac{\pi_v(d_3)}{d_3} + \frac{\pi_v(d_1)}{d_1} \right) \frac{2}{d_3 + d_1 - 2}.$$

Example. Consider a wedge, which is a 3-graphlet with edges (v_1, v_2) and (v_1, v_3). Given the degrees d_1, d_2, d_3 of the corresponding vertices, the probability function is

$$\pi_3(\text{wedge}) = \left(\frac{\pi_v(d_1)}{d_1} + \frac{\pi_v(d_2)}{d_2}\right)\frac{1}{d_1 + d_2 - 2} + \left(\frac{\pi_v(d_1)}{d_1} + \frac{\pi_v(d_3)}{d_3}\right)\frac{1}{d_1 + d_3 - 2}.$$

The bottleneck of the lifting procedure then becomes precomputing the functions $F_{H_i}(d_1, \ldots, d_k)$. Fortunately, we need to only do this once: when a k-graphlet T is sampled via lifting procedure, we find an isomorphism $H_i \sim T$, and use the function F_{H_i} together with the degrees $d_1, \ldots, d_k$ of vertices of T to compute the value of $\pi_k(T) = \frac{1}{K}F_{H_i}(d_1, \ldots, d_k)$.

4.2 Sampling a starting vertex

Framework 1. It is no problem to sample the starting vertex v independently and from an arbitrary distribution π_v in this framework. Thus we sample starting vertices $v_1, \ldots, v_n$ independently, and apply the lifting process to get independent samples $T_1, \ldots, T_n$. The only term that contributes to the variance is $\sum_{T\in\mathcal{V}_k(G)} \frac{\mathbb{1}(T\sim m)}{\pi_k(T)}$, which is determined by $\frac{1}{\pi_k(T)}$, which in turn can be controlled by an appropriate choice of π_v, i.e. the starting distribution.

Example. For $k = 3$, take $\pi_v(u) = \frac{\deg(u)(\deg(u)-1)}{K}$, where $K = \sum_{u\in V_G} \deg(u)(\deg(u) - 1)$. Then

$$\pi_3(\text{triangle}) = \frac{6}{K}, \quad \pi_3(\text{wedge}) = \frac{2}{K}.$$

Calculating K takes $O(n)$ operations (preparation), sampling starting vertex v takes $O(\log(n))$ operations, and lifting takes $O(\Delta)$, where Δ is the maximum vertex degree in G.

Work in progress. For $k = 4$, ...

Framework 2. When we don't have access to the whole graph structure, the best choice of sampling a starting vertex is via a random walk. Biggest advantage of the lifting process in this framework is that it takes exponentially less steps to sample uncorrelated starting vertices (and lift them to uncorrelated graphlets) rather then sample uncorrelated graphlets with SRW.

Definition 1. Given a random walk with transitional probabilities $p(i \to j)$ and stationary distribution $\pi(i)$ on graph G, the mixing time is defined as

$$\tau_G(\varepsilon) = \min\left\{t \mid \frac{\left|p^{(t)}(i \to j) - \pi(j)\right|}{\pi(j)} < \varepsilon \quad \forall i, j \in G\right\}.$$

Here, we'll be using a simple random walk on G with transitional probabilities $p(i \to j) = \frac{1}{\deg(i)}$ whenever j in connected to i with an edge. As long as G is connected and non-bipartite, the stationary distribution of the walk is $\pi_v(u) = \frac{\deg(u)}{2|E_G|}$, where $|E_G|$ is the number of edges in G. The mixing time of that walk on G can be bounded by the following formula.

$$\frac{\mu}{2(1-\mu)\log(\frac{1}{2\varepsilon})} \le \tau_G(\varepsilon) \le \frac{\log(n) + \log(\frac{1}{\varepsilon})}{1-\mu},$$

where μ is the second largest (by absolute value) eigenvalue of the adjacency matrix of G and $n = |V_G|$.

We'll take the worst-case mixing time $\tau_G(\varepsilon) \approx \kappa(\log(n) + \log(\varepsilon^{-1}))$, with $\kappa = (1 - \mu)^{-1}$. Note that κ can vary from 20 to 500 for different social graphs (for more information about mixing time, see [this paper]). On the other hand, it has been shown in [CC paper] that the mixing time of SRW can be as large as $\Omega(n^{k-2})$, even when the graph G has mixing time of constant order.

The lifting procedure allows us to mix samples on the level of vertices instead of the level of subgraphs, making the mixing much faster.

The complete algorithm is thus as follows:

1. Starting from v_0, perform τ_{mix} number of steps of simple Markov Chain on G to get starting vertex v_1.
2. Apply lifting procedure to sample k-graphlet T.
3. Classify T and find the probability $\pi_k(T)$.
4. $v_0 = v_1$
5. Repeat previous steps until time limit or query limit.

There is a choice of mixing steps τ_{mix} that we need to make. This choice would depend on the cost of queries: if queries are cheap, we'll do $\tau_{mix} \approx \kappa \log(n)$, and if the queries are expensive or limited, we choose some constant τ_{mix}. Further discussion on the choice of τ_{mix} when the number of queries is limited will be based on empirical results.