Estimating Network Parameters

Anirban Dasgupta
Joint work with Ravi Kumar and Tamas Sarlos

Google

Google
Setting

• Given a large network, compute its basic parameters
  – Number of nodes
  – Number of edges
  – Fraction of nodes of a particular type
  – Local/global clustering coefficient
  – Triangle count
  – Number of K(3,3)…
Applications

- **Business intelligence**
  - What is the “average size” of ego network in population X?
  - Is social network X growing faster than network Y?

- **Algorithmic reasons**
  - Is the triangle density unusually small in certain portions of the graphs?
  - How does the graph densify over time?
Why is this not easy?

Access to entire graph impossible/expensive
Graph may be constantly evolving
Queries to graph can be noisy
Why is this not easy?

Access to entire graph impossible/expensive
Graph may be constantly evolving
Queries to graph can be noisy

Sampling is a viable option
  • Must settle for an approximation
  • Number of samples
  • Graph access model
Estimation by sampling

• German tank problem
  – Frequentist, Bayesian estimates
  – Military applications

• Mark and recapture
  – Peterson-Lincoln-Chapman indices
  – Used in ecology
Sampling parameters

• For approximating statistic $f$
  – Estimation error $\varepsilon$
  – Confidence level $\delta$

• relative approximation
  – #samples needed to compute $\Pr[ f' \text{ in } (1\pm\varepsilon)f ] > 1 - \delta$

• additive approximation
  – #samples to compute $\Pr[ f' \text{ in } (f \pm\varepsilon) ] > 1 - \delta$
Complexity of sampling

**Theorem** (folklore): $\frac{1}{\varepsilon^2} \log(\frac{1}{\delta})$ iid samples are enough to output an additive $\varepsilon$ estimate

**Proof** (by Chernoff bound): given iid Bernoulli samples $x_1, x_2, \ldots$

$$\Pr[ | \mathbb{E}[X] - \frac{\sum x_i}{k} | > \varepsilon ] < 2 \exp(-k \varepsilon^2)$$

**Theorem** (Canetti, Even, Goldreich, Bar-Yossef, Kumar, Sivakumar): Optimal even when adaptive methods

Relative estimate needs $\frac{\log(\frac{1}{\delta})}{(\mathbb{E}[X] \varepsilon^2)}$ samples
Graph access model

- **“Ideal” model**: nodes can be sampled according to some distribution
  - Uniform random nodes
  - Proportion to (a function of) degree

- **Query oracle**: node level information available when queried with node-id
  - Degree of the node
  - A random neighbor, or all the neighbors

- Given the above query oracle, can implement sampling distributions using random walks
  - Cost of implementing a distribution depends on the corresponding mixing time
Estimating $n = \#\text{nodes}$

- **Birthday paradox:** expected #collisions in $k$ uniform random samples is roughly $k^2/(2n)$
Estimating \#nodes = n

- **Birthday paradox**: expected \#collisions in k uniform random samples is roughly $k^2/(2n)$

- **Collision counting (Katzir, Liberty, Somekh)**:
  - Sample nodes proportional to degree
  - Compute $(\sum d_u)$ and $(\sum 1/d_u)$ over entire sample
  - Output $(\sum d_u)(\sum 1/d_u) / (2 \#collisions)$

- **Degree biased sampling** is random-walk friendly for undirected graphs
Collision Counting

\[ E [\# \text{ collisions } ] = k(k - 1)/2 \sum (d_u/ 2m)^2 \]

**Theorem (KLS):** To get a relative estimate, the number of samples can be written as (certain) norms of the degrees

- If graph is regular then \( O(\sqrt{n} ) \) samples suffice
- If graph has Zipfian degrees with parameter 2, then \( O( n^{1/4} ) \) samples suffice
Average degree

How to estimate average degree $2m/n$
Average degree

How to estimate average degree $2m/n$

- Known $n$, estimate $m$ using edge collisions
  - $O(\sqrt{m})$ samples
Average degree

How to estimate average degree $2m/n$

- Known $n$, estimate $m$ using edge collisions
  - $O(\sqrt{m})$ samples

- Estimate $m$ using node collisions
  - $k^2 / (2n \sum X_{ij}(u) / d(u))$
  - $O(\sqrt{1/n \times d_{avg} / d_{min}})$ samples

- Estimate $m$ and $n$ using collision counting
  - Needs $O(\sqrt{m} + \sqrt{n})$ samples
A natural algorithm

• **Algorithm**
  - Sample nodes *uniformly at random*
  - Return the average of the degrees

• **Theorem (Feige):** If \#samples is at least $O(\sqrt{(n/L)/\epsilon})$ where $L < d_{\text{avg}}$ then we get a $(2 + \epsilon)$ approximation
Limitations

• Naive bound will involve maximum degree
Limitations

- Naive bound will involve maximum degree

- Cannot get better than a 2-approximation
  - This bound is tight with $\Omega(n/d)$ queries

$d(u) = d$ for all $u$

$d(u) = d$ for all $u$ in $[1, n-d]$

$d(v) = n-1$ for other $u$
(1 + ε) - estimator

Goldreich, Ron

- Bucket uniformly sampled nodes by degree
- Discard small buckets (as represented in sample)
  - Reduces variance
  - Estimator is not unbiased

Choose a random neighbor for a node – query it for its degree

Theorem: If #samples is at least \( O(\sqrt{n/L} \log(n)/\text{poly}(\varepsilon)) \) where \( L < d_{\text{avg}} \), then we get a \((1 + \varepsilon)\) approximation
Yet another estimator

- If we have weights $w_1, \ldots, w_n$ and want to get their sum/average
  - How many samples for relative estimate?

**Theorem** (Motwani, Panigrahy, Xu): Need #samples $O(n^{1/3} \text{poly}(\log n, 1/\varepsilon))$

- Combination of weighted and uniform sampling
- Estimate the contribution of heavier weights separately
- Shows this is tight under certain conditions
- Assumes knowledge of $n$
Can we do better?

- Sampling lower bound of $\Omega(\sqrt{n})$
  - Uniform sampling
- Uniform samples are also hard to get
  - Corresponding random walk mixes slowly
- What about non-uniform distributions?
  - e.g. degree biased?
Balancing low vs high degrees?

- **Uniform**: harsh for high degrees
- **Degree biased**: harsh for low degrees
  - How to boost them?
- If we sample nodes with probability proportional to degree + smoothing constant
  - How to choose the smoothing constant?
  - How to implement using random walks?
Overview of algorithm

**Refined estimator:** Arbitrary approximation given a smoothing const

**Coarse estimator:** Constant approximation

**Combined estimator:**

- Sample and run the coarse estimator
- Use the coarse estimator as the smoothing constant and run the refined estimator
Refined Estimator

- Given a coarse estimate $c$, sample $k$ nodes $x_1, \ldots, x_k$ with probability proportional to degree + $c$ and output the estimate

$$\frac{\sum d_u / (d_u + c)}{\sum 1 / (d_u + c)}$$

$$d_{\text{avg}} = \frac{E[A]}{E[B]}$$
Theorem. If $c = \alpha d_{\text{avg}}$ and $k = \max(\alpha, 1/\alpha)1/\varepsilon^2 \log(1/\delta)$ then refined estimator outputs a $(\varepsilon, \delta)$ estimate.

Proof Sketch. Each of A and B are concentrated. Analyze via Bernstein's inequality

Setting $\alpha = 1/d_{\text{avg}}$, $c = 1$ and the #samples $= d_{\text{avg}}/\varepsilon^2 \log(1/\delta)$

[D., Kumar, Sarlos, WWW 2014]
Other Properties

• Bias and variance are bounded
  – Bias at most \((\alpha + 1/\alpha)d_{avg}/k\)
  – Similar bound on variance also

• Random walk version
  – Easy to implement in practice
  – Sample complexity in terms of original mixing time

\[
c/2 \text{ self loops}
\]
\[
\text{degree}(u)
\]
Coarse Estimator

Guess and Verify

For $c$ in $\{1, 2, 4, 8, \ldots\}$

- Sample nodes with probability proportional to $\text{degree} + c$
- If the fraction of nodes with degree below $c$ is more than $5/12$, then return $c$ as a good approximation
Proof of coarse estimator

If $c = \alpha d_{\text{avg}}$, then

$$\frac{(\alpha - 1)}{(\alpha + 1)} < \Pr[d_u < c] < \frac{2\alpha}{(\alpha + 1)}$$

Using this, one can show that

- If $c < d_{\text{avg}}/3$, then the fraction of low degree nodes is $< 5/12$
- If $c > 3d_{\text{avg}}$, then the fraction of low degree nodes is $> 5/12$
**Final Bound**

**Theorem.** Can \((\varepsilon, \delta)\) approximate the average degree by using

\[
\#\text{samples} = (\log U \log \log U + 1/\varepsilon^2) \log(1/\delta)
\]

degree biased node samples, where \(U\) is an upper bound on the maximum degree.
Experiments

- Skitter, DBLP, LiveJournal, Orkut
Collision Counting
Average degree: recap

- Simple estimator for average degree
- No need for purely uniform samples
- Random walk friendly
- Fewer samples than by existing algorithms or using most obvious algos
- Works well in practice
Further questions

- Applications to estimating other properties?
- Directed graphs?
- Bound on oracle queries when using random walk?

- More general question about effectiveness of random walks

- In general, how can we create and maintain a “representation” of large networks in memory such that queries can be answered approximately
Thanks!
Random walks

- Random walk based methods have proven to be extremely useful in social network setting
  - 'small' mixing time → small #samples
  - low memory overhead
  - natural

- **Jerrum-Sinclair**: 
  \[ \tau_{\text{mix}} \propto \frac{1}{\Phi^2} \]
  \[ \Phi = \min_{S, d(S) \leq d(G)/2} \frac{e(S, V \setminus S)}{d(S)} \]
  \( \Phi \) is the minimum conductance of G.
Large Social and Information Networks

Leskovec, Lang, Dasgupta, and Mahoney (WWW 2008 & arXiv 2008)

Focus on the red curves (local spectral algorithm) - blue (Metis+Flow), green (Bag of whiskers), and black (randomly rewired network) for consistency and cross-validation.