Estimating Completeness in Streaming Graphs

Sanghamitra Bandyopadhyay
(Joint work with M. Bhattacharyya and S. Bhattacharya)

Professor, Machine Intelligence Unit
Indian Statistical Institute, Kolkata

Presented at EDBT/ICDT Workshop on MSDM
March 28, 2014

Introduction

Related works

- On completeness and clique numbers
- On streaming algorithms

Theoretical results

- Estimating completeness of bipartite graphs
- Estimating completeness of arbitrary graphs

Proposed algorithms

Empirical results

- Study on Synthetic Networks
- Study on Social Networks

Discussion

Streaming algorithms

In streaming algorithms, the data is available as a stream and we would like –

- the per-item processing time
- storage and
- overall computing time

to be simultaneously $O(N, t)$, preferably $polylog(N, t)$, at any time instant t in the data stream.

Data stream models

Definition (Data stream model)

A *data stream model* defines an input stream $\mathcal{A} =< a_1, a_2, \cdots >$ arriving sequentially, item by item, and describes an underlying signal A, where $A : [1 \ldots N] \rightarrow R$ is a one-dimensional function.

Data stream models

Definition (Data stream model)

A *data stream model* defines an input stream $\mathcal{A} =< a_1, a_2, \cdots >$ arriving sequentially, item by item, and describes an underlying signal A, where $A : [1 \ldots N] \rightarrow R$ is a one-dimensional function.

The data stream models can be of the following three types [Muthukrishnan, 2005]:

- Time Series Model,
- Cash Register Model,
- Turnstile Model.

Data stream models

The models vary based on the information about how the input data elements stream in.

1. Time Series Model
   - Each a_i equals $A[i]$ and they appear in increasing order of i.
   - Observing the traffic at an IP link for each 5 min, or volume estimation of share trading in every 10 min, etc.
2. Cash Register Model
   - Perhaps the most popular data model.
   - Here a_i's are increments to $A[i]$'s.
   - Monitors IP addresses that access a web server.
3. Turnstile Model
   - This is the most general model.
   - Here a_i's are updates to $A[j]$'s.

We will be studying strict turnstile models where updates produce strictly non-negative numbers.

Streaming graph

Definition (Streaming graph)

A *streaming graph* is a simple graph on n vertices $V = \{v_1, v_2, \ldots, v_n\}$ with edges $E = \{(v_i, v_j) : s_k = (i, j)$ for some $k \in [m]\}$, where the data items $s_k \in [n] \times [n]$ are available as an input stream $\mathcal{S} =< s_1, s_2, \ldots, s_m >$.

Streaming graph

Definition (Streaming graph)

A *streaming graph* is a simple graph on n vertices $V = \{v_1, v_2, \ldots, v_n\}$ with edges $E = \{(v_i, v_j) : s_k = (i, j)$ for some $k \in [m]\}$, where the data items $s_k \in [n] \times [n]$ are available as an input stream $\mathcal{S} =< s_1, s_2, \ldots, s_m >$.

Input stream	**Interpretation in terms of a graph**
(1, 2)	Two new vertices and a new edge are included

Streaming graph

Definition (Streaming graph)

A *streaming graph* is a simple graph on n vertices $V = \{v_1, v_2, \ldots, v_n\}$ with edges $E = \{(v_i, v_j) : s_k = (i, j)$ for some $k \in [m]\}$, where the data items $s_k \in [n] \times [n]$ are available as an input stream $\mathcal{S} =< s_1, s_2, \ldots, s_m >$.

Input stream	**Interpretation in terms of a graph**
(1, 2)	Two new vertices and a new edge are included
(1, 3)	A new vertex and a new edge are included

Streaming graph

Definition (Streaming graph)

A *streaming graph* is a simple graph on n vertices $V = \{v_1, v_2, \ldots, v_n\}$ with edges $E = \{(v_i, v_j) : s_k = (i, j)$ for some $k \in [m]\}$, where the data items $s_k \in [n] \times [n]$ are available as an input stream $\mathcal{S} =< s_1, s_2, \ldots, s_m >$.

Input stream	**Interpretation in terms of a graph**
(1, 2)	Two new vertices and a new edge are included
(1, 3)	A new vertex and a new edge are included
(2, 3)	A new edge is included

Streaming graph

Definition (Streaming graph)

A *streaming graph* is a simple graph on n vertices $V = \{v_1, v_2, \ldots, v_n\}$ with edges $E = \{(v_i, v_j) : s_k = (i, j)$ for some $k \in [m]\}$, where the data items $s_k \in [n] \times [n]$ are available as an input stream $\mathcal{S} =< s_1, s_2, \ldots, s_m >$.

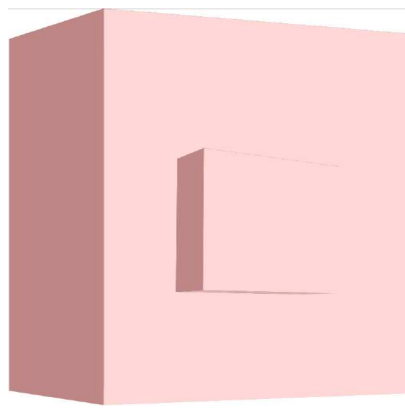
Input stream	**Interpretation in terms of a graph**
(1, 2)	Two new vertices and a new edge are included
(1, 3)	A new vertex and a new edge are included
(2, 3)	A new edge is included
...	...

Sketch

A sketch is often necessary to map the original space to a reduced space, retaining the necessary properties, to achieve this. We formally define a sketch as follows.

Definition (Sketch)

A *sketch* ψ of a data set x, with respect to some function f, is a projection of $x \rightarrow \psi$ from which one can compute $f(x)$.



Heavy hitters

Definition (Heavy hitters)

The ϕ-*heavy hitters* of a set A are the values that are at least ϕ-fraction of the total number of elements in A, i.e. with frequency $\phi|A|$.

Heavy hitters

Definition (Heavy hitters)

The ϕ-*heavy hitters* of a set A are the values that are at least ϕ-fraction of the total number of elements in A, i.e. with frequency $\phi|A|$.

Definition (ϕ-heavy eigen-hitters)

The ϕ-*heavy eigen-hitters* of a graph G are the eigen values that are at least ϕ-fraction of the total mass of all the eigen values of the matrix A_G.

ℓ_p norm

A norm is a function that assigns a strictly positive length (or size) to each vector in a vector space, other than the zero vectors (having a length zero). In general, we define the ℓ_p norm as follows.

Definition (ℓ_p norm)

For any non-zero vector x, the ℓ_p *norm* is defined as

$$||x||_p = (\sum_{i}^{n} |x|^p)^{1/p}, \tag{1}$$

where $p \geq 1$ denotes a real constant.

ℓ_1 *heavy eigen-hitters*

Definition (ℓ_1 heavy eigen-hitters)

The ℓ_1 *heavy eigen-hitters* are the heavy eigen-hitter values based on the total mass in ℓ_1 norm.

On completeness and clique numbers

- Finding the maximum order clique in a graph is known to be an NP-hard problem.

On completeness and clique numbers

- Finding the maximum order clique in a graph is known to be an NP-hard problem.
- The approximation of a maximal clique in polynomial time is hard within a factor of $n^{1-\varepsilon}$ (for any $\varepsilon > 0$), unless NP = ZPP (probabilistic polynomial time algorithm), where n denotes the order of the graph [Hastad, 1999].

On completeness and clique numbers

- Finding the maximum order clique in a graph is known to be an NP-hard problem.
- The approximation of a maximal clique in polynomial time is hard within a factor of $n^{1-\varepsilon}$ (for any $\varepsilon > 0$), unless NP $=$ ZPP (probabilistic polynomial time algorithm), where n denotes the order of the graph [Hastad, 1999].
- A few attempts were made earlier to estimate the chromatic number of a graph using eigen values [Wilf, 1967], which can be further related with the clique number of a graph.

On completeness and clique numbers

- Finding the maximum order clique in a graph is known to be an NP-hard problem.
- The approximation of a maximal clique in polynomial time is hard within a factor of $n^{1-\varepsilon}$ (for any $\varepsilon > 0$), unless NP = ZPP (probabilistic polynomial time algorithm), where n denotes the order of the graph [Hastad, 1999].
- A few attempts were made earlier to estimate the chromatic number of a graph using eigen values [Wilf, 1967], which can be further related with the clique number of a graph.
- Based on eigen value computations, several upper bounds on the clique number were derived previously [Amin, 1972].

On completeness and clique numbers

- Finding the maximum order clique in a graph is known to be an NP-hard problem.
- The approximation of a maximal clique in polynomial time is hard within a factor of $n^{1-\varepsilon}$ (for any $\varepsilon > 0$), unless NP = ZPP (probabilistic polynomial time algorithm), where n denotes the order of the graph [Hastad, 1999].
- A few attempts were made earlier to estimate the chromatic number of a graph using eigen values [Wilf, 1967], which can be further related with the clique number of a graph.
- Based on eigen value computations, several upper bounds on the clique number were derived previously [Amin, 1972].
- These bounds (and also lower bounds) were tightened further [Budinich, 2003].

On completeness and clique numbers

- Finding the maximum order clique in a graph is known to be an NP-hard problem.
- The approximation of a maximal clique in polynomial time is hard within a factor of $n^{1-\varepsilon}$ (for any $\varepsilon > 0$), unless NP = ZPP (probabilistic polynomial time algorithm), where n denotes the order of the graph [Hastad, 1999].
- A few attempts were made earlier to estimate the chromatic number of a graph using eigen values [Wilf, 1967], which can be further related with the clique number of a graph.
- Based on eigen value computations, several upper bounds on the clique number were derived previously [Amin, 1972].
- These bounds (and also lower bounds) were tightened further [Budinich, 2003].
- Current studies indicate that a relation with the spectral radius (largest absolute Eigen value) with the clique might help us to estimate the upper bound of the clique in a streaming model [Kannan, 2009].

On streaming algorithms

- The limited earlier contributions before 2005 have been well reviewed in [Muthukrishnan, 2005].

On streaming algorithms

- The limited earlier contributions before 2005 have been well reviewed in [Muthukrishnan, 2005].
- Diverse efforts were made to revisit and solve a number of problems in a streaming setting. There were studies on matrix approximation, matrix decomposition, low rank approximation, ℓ_p regression, etc.
  [Halko, 2010, Kannan, 2009, Mahoney, 2011].

On streaming algorithms

- The limited earlier contributions before 2005 have been well reviewed in [Muthukrishnan, 2005].
- Diverse efforts were made to revisit and solve a number of problems in a streaming setting. There were studies on matrix approximation, matrix decomposition, low rank approximation, ℓ_p regression, etc.
  [Halko, 2010, Kannan, 2009, Mahoney, 2011].
- There has been an influential line of work on computing a low-rank approximation of a given matrix, starting with the works of [Frieze, 2004, Papadimitriou, 1998].

On streaming algorithms

- The limited earlier contributions before 2005 have been well reviewed in [Muthukrishnan, 2005].
- Diverse efforts were made to revisit and solve a number of problems in a streaming setting. There were studies on matrix approximation, matrix decomposition, low rank approximation, ℓ_p regression, etc.
  [Halko, 2010, Kannan, 2009, Mahoney, 2011].
- There has been an influential line of work on computing a low-rank approximation of a given matrix, starting with the works of [Frieze, 2004, Papadimitriou, 1998].
- Very recently, the ℓ_1 and ℓ_2 heavy eigen-hitter problems have been estimated in the streaming model in a lower dimension [Andoni, 2013]. Andoni and Huy achieved a success probability of $\frac{5}{9}$ [Andoni, 2013]. They also estimated the residual error with the same probabilistic accuracy.

Earlier results

Let A_G be a real symmetric $n \times n$ $(n \geq 1)$ matrix denoting the adjacency relations in a graph G. Further assume $\lambda_i(A_G)$ be the i^{th} largest eigen value of A_G in absolute value. Now, if ψ represents a sketch of the matrix A_G where $\psi = PA_GP^T$, then, we have the following important result from a recent study [Andoni, 2013].

Theorem

There is a linear sketch of the real symmetric matrix A_G, of dimension $n \times n$, using space $O(k^2\epsilon^{-4})(\epsilon > 0,\ k \in \{1, 2, \ldots, n\})$, from which one can produce values $\tilde{\lambda}_i$, for $i \in [k]$, satisfying the following with at least $\frac{5}{9}$ success probability

$$|\lambda_i(A_G) - \tilde{\lambda}_i| \leq \epsilon|\lambda_i(A_G)| + \frac{1}{k}S_1^{k+1},$$

where $S_1^{k+1} = \sum_{i>k} |\lambda_i(A_G)|$ denotes the residual "ℓ_1 error".

Estimating completeness of bipartite graphs

Theorem

On fixing a value of $\epsilon > 0$, *one can ensure whether* G *is a complete bipartite graph by deriving a linear sketch* ψ *from* A_G *whose top two heavy eigen-hitters in absolute value should be the same satisfying*

$$\lambda_1(\psi) = (1 \pm \epsilon)\lambda_1(A_G) \pm S_1^2,$$

and

$$\lambda_2(\psi) = (1 \pm \epsilon)\lambda_2(A_G) \pm 0.5S_1^3,$$

and the third largest eigen value satisfies

$$\lambda_3(\psi) = \pm 0.\dot{3}S_1^4.$$

Estimating completeness of bipartite graphs

Outline of the proof:

The eigen values of a complete bipartite graph G can be ordered as $\{\lambda_1(A_G), 0, \ldots, 0, \lambda_n(A_G)\}$, where $\lambda_1(A_G) = -\lambda_n(A_G) = \lambda$ (say) [Amin, 1972].

Estimating completeness of bipartite graphs

Outline of the proof:

The eigen values of a complete bipartite graph G can be ordered as $\{\lambda_1(A_G), 0, \ldots, 0, \lambda_n(A_G)\}$, where $\lambda_1(A_G) = -\lambda_n(A_G) = \lambda$ (say) [Amin, 1972].

Therefore, if we obtain a decreasing order of the eigen values of A_G in absolute value, we would get $\{\lambda, \lambda, 0, \ldots, 0\}$. Since G does not contain any self-loops, the trace of A_G should be zero, i.e. $\sum_{i=1}^{n} \lambda_i(A_G) = 0$.

Estimating completeness of bipartite graphs

Outline of the proof:

The eigen values of a complete bipartite graph G can be ordered as $\{\lambda_1(A_G), 0, \ldots, 0, \lambda_n(A_G)\}$, where $\lambda_1(A_G) = -\lambda_n(A_G) = \lambda$ (say) [Amin, 1972].

Therefore, if we obtain a decreasing order of the eigen values of A_G in absolute value, we would get $\{\lambda, \lambda, 0, \ldots, 0\}$. Since G does not contain any self-loops, the trace of A_G should be zero, i.e. $\sum_{i=1}^{n} \lambda_i(A_G) = 0$.

It is understandable that if the eigen values are ordered in absolute value, say $\{\lambda'_1(A_G), \lambda'_2(A_G), \ldots, \lambda'_n(A_G)\}$, and if $\lambda'_1(A_G) = \lambda'_2(A_G)$ and $\lambda'_3(A_G) = 0$, then rest of the eigen values of A_G will be certainly zero. So, it is sufficient for A_G, to have the first two largest eigen values same in absolute value and the third one zero, for claiming that the corresponding graph G is complete bipartite.

Estimating completeness of arbitrary graphs

Theorem

On fixing a value of $\epsilon > 0$, *one can ensure whether* G *is a complete graph by deriving a linear sketch* ψ *from* A_G *whose top two heavy eigen-hitters in absolute value satisfy the following*

$$\lambda_1(\psi) = (1 \pm \epsilon)(n-1) \pm S_1^2.$$

and

$$\lambda_2(\psi) = (\epsilon \pm 1) \pm 0.5 S_1^3.$$

Estimating completeness of arbitrary graphs

Outline of the proof:

The eigen values of a complete graph G can be ordered as $\{n-1, -1, ..., -1\}$ [Amin, 1972].

Estimating completeness of arbitrary graphs

Outline of the proof:

The eigen values of a complete graph G can be ordered as $\{n-1, -1, ..., -1\}$ [Amin, 1972].

Therefore, if we obtain a decreasing order of the eigen values of A_G in absolute value, we would get $\{n-1, 1, \ldots, 1\}$. Since G does not contain any self-loops, the trace of A_G should be zero, i.e. $\sum_{i=1}^{n} \lambda_i(A_G) = 0$.

Estimating completeness of arbitrary graphs

Outline of the proof:

The eigen values of a complete graph G can be ordered as $\{n-1, -1, ..., -1\}$ [Amin, 1972].

Therefore, if we obtain a decreasing order of the eigen values of A_G in absolute value, we would get $\{n-1, 1, \ldots, 1\}$. Since G does not contain any self-loops, the trace of A_G should be zero, i.e. $\sum_{i=1}^{n} \lambda_i(A_G) = 0$.

It is understandable that if the eigen values are ordered in absolute value, say $\{\lambda'_1(A_G), \lambda'_2(A_G), \ldots, \lambda'_n(A_G)\}$, and if $\lambda'_1(A_G) = n-1$ and $\lambda'_2(A_G) = 1$, then certainly the rest of the eigen values of A_G should also be one. So, it is sufficient for A_G, to have the first two largest eigen values as $n-1$ and one, respectively, for claiming that the corresponding graph G is complete.

An algorithm for estimating completeness of bipartite graphs

Input: The adjacency matrix A_G of the bipartite graph G.
Output: The decision about the completeness of G.
Algorithmic Steps:

1: Obtain a sketch $\psi = PA_GP^T$, where P is a $t \times n$ matrix with $\Theta(\frac{\log^2 n}{\epsilon^2})$-wise independent entries identically distributed as $N(0, \frac{1}{t})$.
2: Compute the top three largest eigen values of ψ in the decreasing order $\lambda_1(\psi)$, $\lambda_2(\psi)$ and $\lambda_3(\psi)$, respectively.
3: **if** $\lambda_1(\psi) = \lambda_2(\psi)$ and $\lambda_3(\psi) = \pm 0.\dot{3}S_1^4$ **then**
4: G is a complete bipartite graph.
5: **end if**

An algorithm for estimating completeness of any arbitrary graph

Input: The adjacency matrix A_G of the graph G.
Output: The decision about the completeness of G.
Algorithmic Steps:

1: Obtain a sketch $\psi = PA_GP^T$, where P is a $t \times n$ matrix with $\Theta(\frac{\log^2 n}{\epsilon^2})$-wise independent entries identically distributed as $N(0, \frac{1}{t})$.
2: Compute the top two largest eigen values of ψ in the decreasing order $\lambda_1(\psi)$ and $\lambda_2(\psi)$, respectively.
3: **if** $\lambda_1(\psi) = (1 \pm \epsilon)(n-1) \pm S_1^2$ and $\lambda_2(\psi) = (\epsilon \pm 1) \pm 0.5S_1^3$ **then**
4: G is a complete graph.
5: **end if**

Empirical results

We have studied two synthetic networks and a real-life network to verify the performances of the proposed algorithms.

The algorithms were implemented in MATLAB and the simulations were performed on an HP Laptop with Intel(R) Core(TM) i5-2410M processor running at 2.30 GHz speed and having 4 GB primary memory.

Study on Synthetic Networks

We have constructed two synthetic networks, one complete bipartite and another complete network, both having orders 40 for performance analysis of the proposed approaches. The complete bipartite network has equal number of partitions. In both these cases, dimension of the sketch matrix becomes $t \times 40$.

We have varied t from 10 to 25 and several arbitrary matrices are generated by employing random selection method on a normal distribution (identically) with parameters $(0, 0.01)$.

Finally, the eigen values are estimated (using Algorithm 3 and Algorithm 4) and compared with the original values.

Proper selection of the dimension of the sketch vector is very much important.

Study on Synthetic Networks

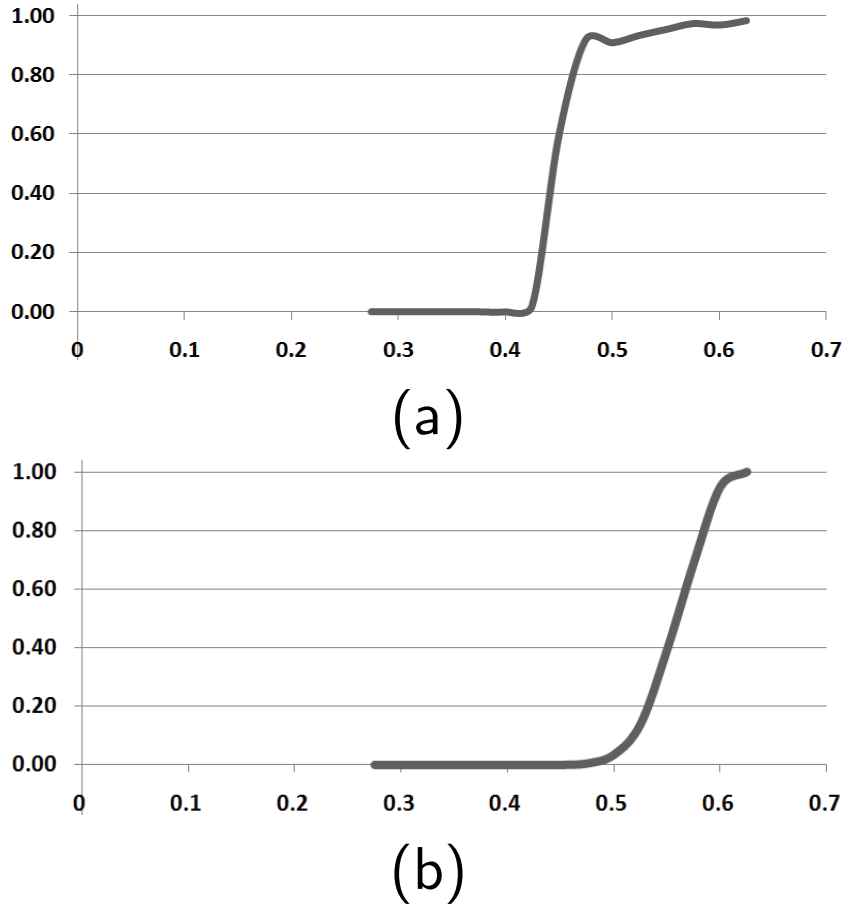


Figure: The average accuracy obtained against the sketch difference of the synthetic (a) complete bipartite network and (b) complete network. x-axis: t/n, y-axis: fraction of largest estimated eigen value to actual eigen value.

Study on Social Networks

We have used a large-scale social interaction data of Facebook, consisting of 'circles' (denoting 'friends lists'), from a recent study [McAuley, 2012]. This interaction data is used to construct a large undirected unweighted social network having 4039 vertices and 88234 edges. The average clustering coefficient of the network is found to be 0.61, establishing that it is not complete.

We have analyzed this and computed a sketch of dimension 100×4039 with elements identically distributed in $N(0, 0.01)$. Finally, Algorithm 4 is applied on this. The obtained eigen values are found to be quite far from the values supporting its completeness (as per Theorem 10).

Discussion

- The theoretical results provided might be useful in estimating the clique number of a graph that is known to depend on the number of eigen values no greater than '-1' [Amin, 1972].
- The implementation details might be useful is developing many other algorithms that work in a streaming setting.
- The approaches to standard vector heavy hitters return the elements that are most frequent [Muthukrishnan, 2005]. On the contrary, we do not find the elements that are heavy hitters, saving an additional factor of $O(\log n)$ to our space requirements (ignoring the random seed size).
- The performances of the proposed algorithms are independent of the seed selection for generating random matrices.
- The algorithms, being linear in computational time, are also capable of supporting arbitrary updates to the matrix in the streaming model.

References

- A. T. Amin *et al.* (1972) *SIAM J Appl Math*, 22(4):569-573.
- Andoni *et al.* (2013) *SODA*, pp. 1729-1737, New Orleans, USA.
- M. Budinich (2003) *Discrete Appl Math*, 127:535-543.
- A. Frieze *et al.* (2004) *J ACM*, 51(6):1025-1041.
- Halko *et al.* (2010) *SIAM Review*, 53(2):217-288.
- J. Håstad (1999) *Acta Math*, 182(1):105-142.

References

- R. Kannan *et al.* (2009) *Found Trend Theor Comput Sci*, 4(3-4):157-288.
- M. W. Mahoney (2011) *Found Trend Mach Learn*, 3(2):123-224.
- J. McAuley *et al.* (2012) *NIPS*, pp. 548-556, Nevada, USA.
- S. Muthukrishnan. (2005) *Found Trend Theor Comput Sci*, 1(2).
- C. H. Papadimitriou *et al.* (1998) *J Comput Syst Sci*, 61(2):217-235.
- H. S. Wilf (1967) *J London Math Soc*, 42(1):330-332.

THANK YOU