Rensselaer Polytechnic Institute

Machine Learning and Optimization
CSCI 4961/6961 Section: 01

Triangle Counting Problem Set
Answer Key

Antonia Calia-Bogan
Richie Massimilla
Gabriel Orlanski

taught by
Prof. Alex Gittens

December 2, 2020

Question 1. (*Asymptotic bound on spectral counting*) Let A be an adjacency matrix for an undirected graph G. Show that counting triangles in G using $\triangle(G) = \frac{1}{6}\mathrm{tr}(A^3)$ has asymptotically bound by $O(|E||V|)$ where E and V are the sets of edges and nodes in G respectively.

It's recommended that you first express the cost of this process using $\deg(V_i)$, the degree of the i^{th} node. The degree of a node is defined as the number of edges incident to that node. Then, you can make use of the equation $\mathbb{E}[\deg(V_i)] = |E|/|V|$ where i is sampled uniformly from the integers between 1 and $|V|$ inclusive.

Question 2. (*Random trace estimation*) Let A be an $n \times n$ symmetric matrix. Let j be a uniform random integer between 1 and n, and let $\mathbf{z} = \mathbf{e}_j$ be the j^{th} standard basis vector. Show that $n\mathbf{z}^\top A\mathbf{z}$ is an unbiased estimator of $\mathrm{tr}(A)$ by showing that the following are true

$$\mathbb{E}[n\mathbf{z}^\top A\mathbf{z}] = \mathrm{tr}(A)$$

$$\mathrm{Var}(n\mathbf{z}^\top A\mathbf{z}) = n\mathrm{tr}(A^2) - \mathrm{tr}^2(A)$$

Then argue why the TraceTriangle algorithms have lower variance than this.

Question 3. (*Implementing TriangleTrace$_N$*) You will implement the TriangleTrace$_{\mathrm{N}}$ algorithm. The algorithm is as follows

> **Algorithm:** $\mathtt{TraceTriangle_N}$
> **Input:** $\gamma \longleftarrow$ a scalar
> **Output:** $\triangle = \mathtt{TraceTriangle_N}(G$, undirected graph with n nodes)
> Form the adjacency matrix $A \in \mathbb{R}^{n\times n}$
> $M = \lceil \gamma \ln^2 n \rceil$
> **for** $i \in 1, \ldots, M$ **do**
>   Form the vector $\mathbf{x} = [x_0, \ldots, x_n]$,
>   where $x_k \sim \mathcal{N}(0, 1)$ are i.i.d
>   and $k \in 1, \ldots, n$
> $y \longleftarrow A\mathbf{x}$
> $T_i \longleftarrow (y^T Ay)/6$
> **end**
> $\triangle \longleftarrow \frac{1}{M}\sum_{i=1}^{M} T_i$

This will require writing a function `tracetriangle(A,gamma)` which takes as input `A`, an adjacency matrix, and `gamma`, a hyperparameter that scales the number of iterations. You must implement this using a sparse matrix representation such as `scipy.sparse.csr_matrix` objects. Use any undirected graphs from the Stanford SNAP dataset (https://snap.stanford.edu/data/).

Experiment with different values of gamma and produce a graph plotting `gamma` values against mean absolute error which is computed by

$$\delta = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{v_A - v_E}{v_E}\right|$$

where N is the number of datasets that a particular value of `gamma` was evaluated over, v_A is the ground truth triangle count, and v_E is the triangle count given by $\mathtt{TraceTriangle_N}$. On the following page a function to load graphs as sparse adjacency matrices and a function for computing the spectral count, which will be used as the ground truth counts, are given.

```python
# IMPORT NX
import networkx as nx

# GRPAH LOADING FUNCTION
def load_graph(path, data=True, delim=None):
    """
    Given the path to a csv file containing a row for every edge,
    parse the data into an adjacency matrix. Each row should have two
    elements, one for each node in the edge.

    Parameters
    ----------------------------------------------------------------
    path : string
    A path to the csv file for the graph.
    data : list of pairs
    Tuples specifying dictionary key names and types for edge
    data.
    delim : string
    Delimiter string for graph read.

    Returns
    ----------------------------------------------------------------
    out : csr_matrix
    The graph's adjacency matrix.
    """
    with open(path, 'rb') as f:
        G = nx.read_edgelist(f, data=data, delimiter=delim)
    # The adjacency list is returned as a csr_matrix as a computational
    # time improvement since most real graphs will be extremely sparse.
    # This turns |V|^2 operations into |E| operations which is a huge
    # improvement.
    A = nx.to_scipy_sparse_matrix(G)
    return A

# SPECTRAL COUNT
def gt_count(A):
    """
    Uses spectral counting to calculate the exact total number of
    triangles in a graph from its adjaceny matrix.

    Parameters
    ----------------------------------------------------------------
    A : csr_matrix
    Adjacency matrix of the graph.

    Returns
    ----------------------------------------------------------------
    out : int
    Exact total count of triangles in the graph.
    """
    cubed = A ** 3
    trace = cubed.diagonal().sum()
    return trace // 6  # This will be an integer regardless
```