CME 323: Distributed Algorithms and Optimization

Instructor: Reza Zadeh (rezab@stanford.edu)

HW#4 - Due 6/6

1. **Shallow Graphs** For an undirected graph $G = (V, E)$ with n vertices and m edges ($m \geq n$), we say that G is shallow if for every pair of vertices $u, v \in V$, there is a path from u to v of length at most 2 (i.e. using at most two edges).

   (a) Give an algorithm that can decide whether G is shallow in $O(n^{2.376})$ time.

   (b) Given an $n \times r$ matrix A and an $r \times n$ matrix B where $r \leq n$, show that we can multiply A and B in $O\left((n/r)^2 r^{2.376}\right)$ time. Hint: use the fact that we can multiply two $r \times r$ matrices in $O(r^{2.376})$ time.

   (c) Give an algorithm that can decide whether G is shallow in $O(m^{0.55} n^{1.45})$ time. Hint: consider length-2 paths that go from low-degree vertices and length-2 paths that go through high-degree vertices separately. Use result from part (b).

   Solution:

   (a) Consider the adjacency matrix A for G. A_{ij} contains the number of paths of length 1 from node i to j. Similarly, A^2_{ij} contains the number of paths of length 2 from node i to j. Thus, $(A^2 + A)_{ij}$ contains the number of paths of length at most 2 from node i to j. Our algorithm will compute $A^2 + A$ and return true if and only if all non-diagonal entries of $A^2 + A$ are non-zero. A^2 can be computed in $O(n^{2.376})$ using Strassen's algorithm. A can be computed in $O(n^2)$ time, for a total running time of $O(n^{2.376} + n^2) = O(n^{2.376})$.

   (b) We simply split up the $n \times r$ matrix into n/r $r \times r$ matrices, and use block matrix multiplication. In the case that r does not divide n exactly, we can simply add rows of zeros to the left-hand multiplicand matrix, and add columns of zeros to the right-hand multiplicand matrix and then remove extraneous rows and columns from the result.

   We perform $\lceil n/r \rceil \times \lceil n/r \rceil$ block matrix multiplications, each taking $O(r^{2.376})$ time.

   The runtime will be $O(\lceil n/r \rceil^2 r^{2.376}) = O((n/r + 1)^2 r^{2.376}) = O((n/r)^2 r^{2.376})$.

   (c)
   ```
   for edge (v, w) ∈ E do
       if v is low-degree then
           for each neighbor u of v do
               M[u, w] = 1
               M[w, u] = 1
           end for
       end if
       if w is low-degree then
           for each neighbor u of w do
               M[u, v] = 1
               M[v, u] = 1
   ```

```
        end for
    end if
end for
```

We will maintain a boolean matrix M that will have $M_{ij} = 1$ if and only if there is a path of length at most 2 between node i and j. We initialize $M = A$, the adjacency matrix for G, leaving only paths of length 2 to be considered. At the end, we check each entry of M and claim the graph is shallow if and only if all non-diagonal entries of M are positive. Since M is initialized to A, it already contains paths of length 1. We will continuously update M to take into account paths of length 2. To do that, we look at all possible ordered triples (u, v, w). Each triple defines a path of length 2 going from u to w, through v.

We split the vertex set into two sets:

$$V_H = \{v \in V \mid \deg(v) > d\}, \quad V_L = \{v \in V \mid \deg(v) \leq d\}$$

Consider each ordered triple (u, v, w) defining a path from u to v to w. Either $v \in V_L$ or $v \in V_H$.

Case: $v \in V_L$, i.e., the middle vertex is low degree

This step takes at most $O(md)$ time since for each edge we check at most d neighbors.

Case: $v \in V_H$, i.e., the middle vertex is high-degree

We construct a matrix B with dimensions $n \times r$ where $r = |V_H|$. Each row corresponds to a node in V and each column corresponds to a node in V_H. $B_{ij} = 1$ if and only if there is an edge between arbitrary node i and V_H-member j. Thus BB^T gives us the number of paths of length 2 from arbitrary node i to arbitrary node j that go through some high-degree node as the middle node. We can do the BB^T computation in $O((n/r)^2 r^{2.376})$ time. We then update M to $M = M + BB^T$. Since $2m =$ sum of all degrees $\geq |V_H|d = rd$. Thus $r \leq 2m/d$. So the computation takes $O((n/r)^2 r^{2.376}) = O(n^2 r^{0.376}) = O(n^2 (m/d)^{0.376})$.

So now we've covered all cases, M accounts for all possible paths of length 2 going through high-degree or low-degree vertices.

Finally we traverse M and claim the graph is shallow if and only if all non-diagonal entries of M are non-zero. This $O(n^2)$ will be dominated by $O(n^{1.45} m^{0.55})$, since $m \geq n$.

Thus total running time is $O(md + n^2 (m/d)^{0.376})$. We now minimize this bound with respect to d. Setting $md = n^2 (m/d)^{0.376}$ gives $d^* = n^{1.45} m^{-0.45}$. Substituting back in gives a bound of $O(md^* + n^2 (m/d^*)^{0.376}) = O(n^{1.45} m^{0.55})$.

2. Write a Spark program to compute the Singular Value Decomposition of the following 10×3 matrix:

```
-0.5529181 -0.5465480 0.009519836
```

```
-0.5428579 -1.5623879 0.982464609
-1.3038629  0.5715549 0.499441144
 0.6564096  1.1806877 0.495705999
-1.2061171  1.3430651 0.153477135
 0.2938439 -1.7966043 0.914381381
-0.2578953  0.2596407 0.815623895
 0.9659582  2.3697927 0.320880634
-0.4038109  0.9846071 0.488856619
 0.6029003 -0.3202214 0.380347546
```

Assume the matrix is tall and skinny, so the rows should be split up and inserted into an RDD. Each row can fit in memory on a single machine. Report all singular vectors and values and submit your Spark program.

3. Given a matrix M in row format as an RDD[ARRAY[DOUBLE]] and a local vector x given as an ARRAY[DOUBLE], give Spark code to compute the matrix vector multiply Mx.

   Solution:

```
x_bc = sc.broadcast(x)
output = M.map(lambda row: np.dot(row, x_bc.value)).collect()
```

4. In class we saw how to compute highly similar pairs of m-dimensional vectors x, y via sampling in the mappers, where the similarity was defined by cosine similarity: $\frac{x^T y}{|x|_2 |y|_2}$. Show how to modify the sampling scheme to work with overlap similarity, defined as

$$\text{overlap}(x, y) = \frac{x^T y}{\min(|x|_2^2, |y|_2^2)}$$

   (a) Prove shuffle size is still independent of m, the dimension of x and y.

   (b) Assuming combiners are used with B mapper machines, analyze the shuffle size.

   Solution:

   (a) We modify the DIMSUM mapper as follows:

Algorithm 1 DIMSUMOverlapMapper(r_i)

for all pairs (a_{ij}, a_{ik}) in r_i **do**
  With probability $\min\left(1, \gamma \frac{1}{\min(\|c_i\|_2^2, \|c_j\|_2^2)}\right)$
  emit $((j, k) \to a_{ij} a_{ik})$
end for

The shuffle size of this scheme is $O(nL\gamma/H^2)$ where H is the smallest nonzero element of A in magnitude. To show this we start with the expected contribution from each pair of columns.

$$
\begin{aligned}
&= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=1}^{\#(c_i,c_j)} P(\text{DIMSUMOverlapEmit}(c_i, c_j)) \\
&= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \#(c_i, c_j) P(\text{DIMSUMOverlapEmit}(c_i, c_j)) \\
&\leq \sum_{i=1}^{n} \sum_{j=i+1}^{n} \gamma \frac{\#(c_i, c_j)}{\min(\|c_i\|_2^2, \|c_j\|_2^2)} \\
&= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \gamma \frac{\#(c_i, c_j)}{c_i^T c_i} \\
&\leq \gamma \sum_{i=1}^{n} \frac{1}{c_i^T c_i} \sum_{j=1}^{n} \#(c_i, c_j) \\
&\leq \gamma \sum_{i=1}^{n} \frac{1}{\#(c_i) H^2} L \#(c_i) \\
&= \gamma L n / H^2
\end{aligned}
$$

The fourth equality comes from assuming WLOG $\|c_i\|_2^2 \leq \|c_j\|_2^2$.

(b) In the naive case with combiners, each of the B machines will emit at most n^2 pairs — one for each element in $A^T A$. However, without combiners we know that DIMSUM will have a shuffle size of at most $nL\gamma/H^2$. Thus the shuffle size is at most $O(\min(Bn^2, nL\gamma/H^2))$.