CME 323: Distributed Algorithms and Optimization

Instructor: Reza Zadeh (rezab@stanford.edu)

HW#3

1. Consider solving connected components via the Pregel framework. In general the naive algorithm works well because the diameter of the graph is low. However for arbitrary graphs the diameter can be $O(n)$. In this question we consider some attempts to reduce the worst-case runtime of the naive algorithm.

   Recall that at step s of the naive algorithm the state of node i is the maximum ID of all nodes within distance s of node i.

   (a) Suppose we run the naive algorithm, but on the kth iteration we add an edge between each node i and the node j, with the maximum ID within distance k of i, if and only if the shortest path between i and j is of length exactly k. Once these edges have been added the naive algorithm completes running as normal. Provide an example of a graph with high diameter and give a description of the node IDs for which this algorithm would add $O(n-k)$ edges (given n is large and the graph is sparse, this is undesirable as it significantly increases the shuffle size). Analyze the number of iterations required by this algorithm.

   (b) Given the same algorithm in (a) and the same graph, identify the distribution of IDs that achieves the same number of iterations but with the fewest number of edges added. Analyze the number of iterations and identify the optimal k.

   (c) Now suppose we run the naive algorithm, but on the kth iteration each node i selects a node j at random within distance k from itself. If the distance from i to j is greater than αk then we add an edge to the graph; otherwise we do nothing. Provide an algorithm that would implement this efficiently (assume k and α are global parameters known to all nodes).

   (d) Analyze the expected performance of the algorithm in (c) on the chain graph. What values of α and k would you suggest? What kind of speed up is possible?

2. For The Perceptron, we proved in class that if all points are scaled to have *unit norm*, and the two classes can be separated by margin at least γ, then the perceptron will make at most $1/\gamma^2$ mistakes. Notice that the margin changes depending on the scaling of the points. Prove that if all the points have norm at most R (i.e. $|x_i| < R$ for all i - all points live within radius R of the origin), then the perceptron makes at most R^2/γ^2 mistakes.

3. 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.

4. Assume we are optimizing a separable function of n separable parts, where the function has p parameters to be tuned.

   (a) Recall that Parallel Stochastic Gradient Descent for strongly convex functions needs one shuffle of the data and one Reduce to compute an average. What is the total communication cost for this, and how much time does it take when k processors are used?

   (b) Similarly, ADMM requires T iterations, where the communication cost of each iteration consists of an AllReduce. With T iterations, what is the total communication cost of ADMM, and how much time does it take?

   Finally, notice that the amount of communication performed by ADMM is independent of n, whereas Parallel SGD is not.