CS 229r: Algorithms for Big Data Fall 2013

Lecture 25 — November 26, 2013

Prof. Jelani Nelson *Scribe: Thomas Steinke*

1 Overview

Tody is the last lecture. We will finish our discussion of MapReduce by covering "Solve-and-Sketch" [ANOY14] which is extremely recent work. We will also say a few words about k-means [BMV$^+$12] and other theoretical "big data" activity that we didn't cover.

2 Solve-and-Sketch

We will discuss the Solve-and-Sketch approach towards approximation algorithms for low-dimensional geometric graph problems in the MapReduce framework [ANOY14]. In particular, this approach yields efficient parallel algorithms for

(a) approximate minimum spanning tree,

(b) approximate minimum cost bipartite matching, and

(c) earthmover distance.

Our input is a set of points T in $\mathbb{R}^d$. We associate these with the complete graph where the edges are weighted by the Euclidean distance (or any ℓ_p distance). Today we will focus on (a).

2.1 Earthmover Distance

We will not say anything about earthmover distance beyond defining it.

Consider a set A of points. Each point $x \in A$ has an initial mass $\mu(x) \in \mathbb{R}_+$ associated with it. Each point also has a final mass $\nu(x)$ associated with it. The cost of moving η units of mass from point x to point y is $\eta \cdot ||x - y||$. The *earthmover distance between μ and ν* is the minimum total cost of a series of operations to move the mass from the initial configuration to the final configuration.

Earthmover distance defines a metric on distributions. It is used in computer graphics. Here the points are the pixels and the initial and final configurations are the brightness values of two images. The earthmover distance is empirically a good measure of similarity between images.

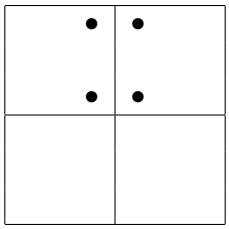
2.2 Solve-and-Sketch Approach

The approach is to hierarchically partition the input set T. The root node contains all points T and the leaves have one point each. There are L levels and the branching factor is c.

For example, a quadtree gives such a hierarchical partition by partitioning $\mathbb{R}^d$ into $c = 2^d$ orthants.

Our goal is to recursively compute a minimum spanning tree going up the hierarchical partition. In particular, we want to be able to contract edges and thereby reduce the number of points we need to keep track of at higher levels. However, we don't want to "regret" choosing an edge later.

Problem A problematic input in the pland and quadtree partition is the following.



The optimal solution joins the upper and lower pair of points with short edges and uses one long edge to join both pairs together. The hierarchical solution uses two long edges to join the points within quadrants. Thus we see that using a plain quadtree gives poor results.

Solution Instead we sill use a randomly shifted and rotated quadtree. Intuitively, the above problem is very brittle. If the partition is shifted slightly, the problem disappears. A randomized quadtree ensures that nearby points are likely to end up in the same subtree.

The overall Solve-and-Sketch algorithm locally computes a partial solution as well as some extra information, based soley on what information is passed to it from its c children. This is called the "unit step": on input of size n_u it produces output of size $p_u(n_u)$ and runs in time $t_u(n_u)$ using space $s_u(n_u)$.

Theorem 1. *Fix $s = (\log n)^{\Omega(d)}$. Suppose there is a unit step with $s_u(p_u(s)) \leq s^{1/3}$ and $p_u(s) \leq s^{1/3}$. Then, for $c = s^a$ ($a < 1$) and $L = O(d \log_s n)$ we can implement Solve-and-Sketch in MapReduce in poly$(\log_s n)$ rounds where each node's runtime is $s \cdot t_u(s)$poly$(\log n)$.*

2.3 Randomized Hierarchical Partitions

Definition 2 ([Tal04])**.** *A randomized hierarchical partition B is (a, b)*-distance preserving with approximation $\gamma > 1$ *($0 < a < 1$) if, for $\Delta_\ell = \gamma \cdot a^{L\ell} \cdot diam(T)$ and every partition $P = (P_0 \cdots P_L)$ in the support of the distribution we have*

- $\forall \ell \geq 0 \quad diam(P_\ell) \leq \Delta_\ell$ *and*
- $\forall x, y \in T, \ell \geq 0 \quad \mathbb{P}[x$ *and* y *are separated at level* $\ell] \leq b||x - y||_2/\Delta_\ell$.

Theorem 3 ([AKS13, §6])**.** *A randomly shifted and rotated c-ary quadtree is $(c^{-1/d}, O(d))$-distance preserving with approximation $O(1)$.*

2.4 Unit Step for Minimum Spanning Tree

A node is responsible for the set C of points in its subtree. It computes a 'spanning forest'.

Input. $V(C) \subset C$ and a partition $\{Q_1 \cdots Q_k\}$ of $V(C)$ corresponding to the connected components of $V(C)$ based on edges that were contracted at lower levels.

1. While $\exists i, j \in [k]$ such that $\ell_2(Q_i, Q_j) \leq \varepsilon \Delta_\ell$:

1a. Pick $u \in Q_i$ and $v \in Q_j$ such that $||u - v||_2$ is minimal.

1b. Output edge (u, v) as an edge in the final spanning tree.

1c. Merge Q_i and Q_j.

2. Output $V' \subset V(C)$ that is an $\varepsilon^2 \Delta_\ell$-cover of V and the induced partition Q' of V'.

Recall that, if (X, d) is a metric space, $X' \subset X$ is an ε-cover of X if, for all $x \in X$, there exists $x' \in X'$ such that $d(x, x') \leq \varepsilon$.

For input points u and v, we define $\ell_{\text{cut}}(u, v)$ to be the highest level in the hierarchy where u and v are not in the same node.

Each node does the following.

It gets $V'_1 \cdots V'_c$ and partitions $Q'_1 \cdots Q'_c$ from its children.

It sets $V = \bigcup_i V'_i$ and $Q = \bigcup_i Q'_i$. (At leaves V is all the points and Q a partition into singletons.)

It runs the unit step with input V and Q to obtain output V' and Q'.

It passes V' and Q' to its parent.

2.5 Analysis Sketch

(i) It's clear that the output is a forest. It turns out that we get a spanning tree.

(ii) Show that there exist edge weights such that the algorithm outputs a minimum spanning tree with respect to those weights (rather than the original distance weights).

(iii) There is a way to define these weights in terms of the tree such that

$$\forall u, v \quad ||u - v||_2 \leq w(u, v) \leq (1 + \beta)||u - v||_2 + \alpha \Delta_{\ell_{\text{cut}}(u,v)},$$

where $\beta = O(\varepsilon)$ and $w(u, v)$ is the weight of edge (u, v). Thus $\mathbb{E}[w(u, v)] \leq (1 + O(\varepsilon))||u - v||_2 + \alpha \, \mathbb{E}[\Delta_{\ell_{\text{cut}}(u,v)}]$.

(iv) Because we used a randomized hierarchical partition (Definition 2), we have

$$\mathbb{E}[\Delta_{\ell_{\text{cut}}(u,v)}] = \sum_{\ell=0}^{L} \mathbb{P}[\ell_{\text{cut}}(u, v) = \ell] \cdot \Delta_\ell \leq \sum_{\ell=0}^{L} b||u - v||_2 / \Delta_\ell \cdot \Delta_\ell = b||u - v||_2 L.$$

3 Other MapReduce Topics

We were not able to cover all the work on MapReduce. Here is an incomplete list of topics.

3.1 Sorting

We can perform sorting in MapReduce with $O(\log_M N)$ rounds and $O(N \log_M N)$ total communication, where M is the Input/Output bound on individual mappers/reducers; see [GSZ11]. This sorting algorithm is used as a subroutine in Solve-and-Sketch.

3.2 k-means Clustering

Given $x_1 \cdots x_n$ we wish to find k centers $c_1 \cdots c_k$ that minimize $\sum_{i \in [n]} \min_{j \in [k]} ||x_i - c_j||_2^2$.

A popular method is Lloyd's algorithm:

1. Start with initial centers $c_1 \cdots c_k$.
2. Partition the points into clusters.
3. Move each center to the average of the points in its cluster.
4. Repeat.

Lloyd's algorithm can only improve the objective function (see problem set 4 problem 2a). But it remains to choose good initial centers.

k-means++ The initial centers can be chosen as follows [AV07].

1. Let C be one random data point.
2. While $|C| < k$:

2a. Sample one data point, where x is chosen with probability proportional to $\min_{c \in C} ||x - c||_2^2$.

2b. Add the sample to C.

3. Output C.

It can be shown that this choice of C is within a $O(\log k)$ factor of optimal with high probability. Unfortunately, this method is inherently sequential.

k-means|| An alternative approach which is more parallelizable is as follows [BMV+12].

1. Let C be one random data point.
2. Let $\psi = \text{cost}(C) = \sum_{i \in [n]} \min_{c \in C} ||x - c||_2^2$.

3. For $O(\log \psi)$ iterations:

3a. Sample each data point x independently with probability

$$\Theta\left(\frac{k \cdot \min_{c \in C} ||x - c||_2^2}{\text{cost}(C)}\right).$$

3b. Add this sample to C.

4. Do some cleanup to reduce C to the appropriate size.

References

[AKS13] Dror Aiger, Haim Kaplan, and Micha Sharir. Reporting neighbors in high-dimensional euclidean spaces. In *SODA*, pages 784–803, 2013.

[ANOY14] Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. 2014.

[AV07] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[BMV+12] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7):622–633, 2012.

[GSZ11] Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the mapreduce framework. In *Algorithms and Computation*, pages 374–383. Springer, 2011.

[Tal04] Kunal Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 281–290. ACM, 2004.