COMS E6998-9: Algorithms for Massive Data (Spring'19) Apr 25, 2019

Lecture 26: Presentations, Class Recap

Instructor: *Alex Andoni* Scribes: *Patrick Kwok*

1 Introduction

This lecture contains two parts. The first part includes three final project presentations, and the second part is a recap of what has been covered throughout this semester starting from the first lecture. In the remaining two lectures, the other groups who have not presented yet will give their presentations of their final projects.

2 Final Project Presentations

2.1 Learning Fast Linear System Solvers

The presentation of this final project is given by Kiran Vodrahalli.

The main focus of this project is solving linear system problems in the form $Ax = b$, where $x \in \mathbb{R}^n$, $b \in \mathbb{R}^n$, and $A \in \mathbb{R}^{n \times n}$. This n-dimensional linear system can be rewritten as a normal equation $A^T Ax = A^T b$.

We first need to define the linear system approximation problem (A, b, ϵ), as follows:

Definition 1. *x^* is an ϵ-approximate solution to $Ax = b$ if $||Ax - \Pi_A b||_2 \leq \epsilon ||\Pi_A b||_2$*

Here Π_A is the projection of b onto the image of A.

The fast structured linear system solvers aim to solve the linear system approximation problem (A, b, ϵ) in "near-linear" time, where "near-linear" time refers to $O((|A|_0 + n) \log n \log \frac{1}{\epsilon})$ time. We may not be able to solve all linear system approximation problems in near-linear time, so we need to put some constraints on the structure of A.

One sub-type of linear system approximation problems where fast (near-linear time) solvers exist is Laplacian Systems. Given a graph G with vertices x_i, edges (x_i, x_j) and weights of edges w_{ij}, the graph Laplacian can be defined as $x^T L_G x = \sum_{i<j} w_{ij}(x_i - x_j)^2$. There is one equation for each edge, in the form $w_r(x_i - x_j) = b_r$, and if we stack all such equations $B^T B$ would be the Laplacian. It turns out that fast solvers for Laplacian Systems exist, by first preconditioning a system to solve smaller systems, and then run conjugate gradient algorithm on the smaller systems.

While certain types of real-life problems, like the max-flow problem, can be solved by casting them

as Laplacian Systems, for some other problems we need a generalized form of the Laplacian Systems. One such type of generation is the Graph-Structured Block Matrices (GSBMs), in which for each edge (x_i, x_j) in the graph G, the weight w_{ij} as in Laplacian Systems is replaced by a pair of matrices (S_r, T_r), and the equation for each edge thus becomes $S_r x_i - T_r x_j = b_r$, where $S_r, T_r \in \mathbb{R}^{r \times d}$, $x_i \in \mathbb{R}^d$, and $b_r \in \mathbb{R}^s$. Note that S_r and T_r need not be equal. Similar to the case in Laplacian Systems, if we stack the equations as rows, then the Graph-Structured Block Matrix (GSBM) is $B^T B$.

A special type of GSBMs is the Multi-Commodity Flow Systems. In this scenario $S_r = T_r$, so the equation for each edge becomes $S_r x_i - S_r x_j = b_r$. Furthermore, the structure of S_r is restricted. In the MC_2 system where there are only two commodities, S_r can only be in the form $w_r(1, 0)$, $w_r(0, 1)$ or $w_r(1, -1)$. While it seems that Multi-Commodity Flow Systems are much simpler to solve than general GSBMs, it turns out that (according to Kyng and Zhang, 2017) "if linear systems in 2-commodity Laplacians can be solved approximately in nearly-linear time, then all linear systems ... can be solved in nearly-linear time."

In order to reduce general integral linear systems to MC_2 systems, we can first reduce them to $G_{z,2}$ systems, which are general integral linear systems with coefficients summing to 0 and positive coefficients summing to powers of 2. Then we reduce $G_{z,2}$ systems to MC_2 systems. For the second reduction, the key idea is to simplify rows by adding new constraints. For example, we can add $C(2x' - (x_i + x_j))$ to the left-hand side of a row equation, where x' is a new variable, then add $C((x_i + x_j) - 2x') = 0$ to the equations as a new constraint.

2.2 Auditing Fairness Through Awareness

The presentation of this final project is given by Darshan Thaker and Roland Maio.

The key idea of this project is to design an evaluation system for individuals, or a mapping from individuals to outcomes, such that similar individuals should receive similar outcomes. We may define the following elements:
V is the universe of individuals;
A is the set of outcomes;
M is a randomized classifier mapping individuals in V to distributions over outcomes in A;
d is a task-specific similarity metric over the universe of individuals;
D is a similarity metric over distributions.
The key idea can then be formalized as the follows: For any $v, v' \in V$, $D(M(v), M(v')) \leq d(v, v')$.

The key idea specified above is consistent with the intuitions and notions of individual fairness, and it implies some notions of group fairness under certain conditions. Fairness Through Awareness (FTA) can be efficiently learned given unlimited access to d, and relaxed FTA is learnable with few queries to d.

FTA can be expressed as a Lipschitz Testing problem:

Definition 2. *f is 1-Lipschitz if and only if for any $x, y \in X$, $||f(x) - f(y)||_2 \leq ||x - y||_2$*

Here f is a function defined over X.
FTA then becomes equivalent to testing whether or not M is 1-Lipschitz.

According to Jha et al. (2013), there is an algorithm for Lipschitz Property Testing (LPT) of a function f that maps $\{0,1\}^d$ to $\mathbb{R}$, as follows:
1. Sample random points and check that diameter of the function is at most d
2. Sample random edges of the hypercube and reject if we find a violation

To prove that this algorithm is correct, we can relate the number of violations on the edges of hypercube, $V(f)$, to how "far" we are from Lipschitz, with the following lemma:

Lemma 3. *If $f : \{0,1\} \to \mathbb{Z}$ is ϵ-far from Lipschitz, then $V(f) \geq \frac{\epsilon \times 2^{d-1}}{ImageDiam(f)}$*

Proof. This lemma can be proved by transforming an ϵ-far function into Lipschitz, by modifying it on violations. We can "repair" one dimension at a time. □

2.3 Sketching Earth Mover Distance

The presentation of this project is given by Dan Mitropolsky and Tim Randolph.

The Earth Mover Distance is defined as follows:

Definition 4. $EMD(A,B) = \min_{\pi:A \xrightarrow{1:1} B} \sum_{a \in A} ||a - \pi(a)||$

Here $A, B \subset [\Delta]^d$, $|A| = |B| = n$.

An approximation of the EMD can be found by embedding. We need to find a "clever" embedding into a different space, and then reconstruct a good approximation of the original quantity (EMD) in this space. For example, if the embeddings of A and B in the new space are $v(A)$ and $v(B)$ respectively, then $R(v(A), v(B)) \approx d(A,B)$, and if $A, B \subset [\Delta]^d$, we have $v(A), v(B) \in \{\mathbb{N}\}^k$, and $||v(A) - v(B)||_1 \approx EMD(A,B)$. Note that embedding in l_1 enables us to use LSH to speed up nearest-neighbor search under EMD.

The project covers two types of approximation. One of them is the log-Δ approximation. For $i = -1, 0, 1, ..., \log \Delta$, let $v_i(A)$ be the histogram vector of the d-dimensional grid with side length 2^i. The embedding is $v(A) = 2^{-1}v_{-1}(A), 2^0 v_0(A), 2^1 v_1(A), ..., 2^{\log \Delta} v_{\log \Delta}(A)$. According to Indyk and Thaper (2003):

Theorem 5. *For some constants c_1 and c_2, we have:*
$c_1 \times EMD(A,B) \leq ||v(A) - v(B)||_1 \leq c_2 \log \Delta \times EMD(A,B)$

Initution: $||v(A) - v(B)||_1$ corresponds to a matching in which we pair points with their cellmates in the tightest possible grid.

The other type of approximation is $\frac{1}{\epsilon}$-approximation. Here we extend the metric to multisets of potentially different sizes, and for $A, B \subset [\Delta]^2$,
$EEMD_\Delta(A,B) = \min_{S \subset A, S' \subset B, |S| = |S'|} [EMD(S,S') + \Delta(|A - S| + |B - S'|)]$
Here "set $A \subset [\Delta]^2$" corresponds to having a vector $x_A \in \mathbb{R}^{\Delta^2}$. $EEMD(A,B)$ is induced by a metric on $\mathbb{R}^{\Delta^2}$, meaning that there exists a metric $||.||_{EEMD}$ such that $EEMD_\Delta(A,B) = ||x_A - x_B||_{EEMD}$.

For $x \in \mathbb{R}^{\Delta^2}$, $||x||_{EEMD}$ can be approximated by a sum of norms $\sum_i ||F_i(x)||_{EEMD}$ where each $F_i(x) \in \mathbb{R}^{\Delta^\epsilon}$.

Theorem 6. *Fix $0 < \epsilon < 1$. Then, for $n = \Delta^{O(1)}$ we can randomly map $x \to F_1(x), ..., F_n(x)$, where each $F_i : \mathbb{R}^{\Delta^2} \to \mathbb{R}^{\Delta^\epsilon}$ such that for any $x \in \mathbb{R}^{\Delta^2}$,*
1. $||x||_{EEMD} \leq \sum_i ||F_i(x)||_{EEMD}$
2. $\mathbb{E}[\sum_i ||F_i(x)||_{EEMD}] \leq O(\frac{1}{\epsilon})||x||_{EEMD}$

This theorem was proved by Indyk (2007) and used to get $N \log^{O(1)}(N)$ time $O(1)$-approximation.

There is also a general technique for approximating sum of norms in any normed space. Let $x = (x_1, ..., x_n)$, each $x_i \in X$, so $x \in X^n$. Denote $SUM(x) = \sum_i ||x_i||_X$.

Theorem 7. *Suppose there is a threshold M and factor γ s.t. you know that $\frac{M}{\gamma} \leq SUM(x) \leq M$. Then, with only $k = (\gamma \log n)^{O(1)}$, we can choose random map $\mu : X^n \to X^k$ s.t. with high probability, $E(\mu(X))$ is $O(1)$-approximation to $SUM(x)$, where E is some non-linear estimator.*

This is a main development of Andoni (2009).

Now we can combine the two techniques,
1. $\sum_{i=1}^n ||F_i(x)||_{EEMD}$ is $O(\frac{1}{\epsilon})$-estimate to $||x||_{EEMD}$, with $n = \Delta^{O(1)}$ and each $F_i(x) \in \mathbb{R}^{\Delta^\epsilon}$
2. From Indyk (2003) we can approximate $||x||_{EEMD}$ up to $\log \Delta$, and we estimate M such that $\frac{M}{\gamma} \leq \sum_{i=1}^n ||F_i(x)||_{EEMD} \leq M$ where $\gamma = \log \Delta$. Using sum of norms approximation, we need $(\gamma \log n)^{O(1)} \leq (\log \Delta)^{O(1)}$-sized sketch.
Overall, the size of the sketch is $(\log \Delta)^{O(1)} \Delta^\epsilon$.

3 Class Recap

Over the course of the semester we have discussed efficient data representation, in which we have a function to map "data unit" to some representation of the original data. This representation can be sketch, samples, etc. This function for compressing the data is usually good for a particular task.

We have discussed streaming models, in which given a stream of data we need to compute some properties of the data stream. For example, we can compute the length of the stream, the number of distinct elements, the l_p-norm of the frequency vector, the heavy hitters, and so on. For the l_2-norm of the frequency vector, F_2, we have discussed a tug-of-war algorithm. We can use a linear sketch for dimension reduction.

Other topics that have been covered include metric embeddings for dimension reduction, L.S. hashing (we care only about cases where the distance is smaller than r or larger than $c \times r$, and we want to estimate the probability of a collision for the hash function), "sketch and solve" (using dimension-reduction methods to reduce the size of a problem), streaming algorithms for graphs (for determining the connectivity and detecting connected components of a graph, etc., using sketches), distribution testing (determining whether the distribution of a dataset is the same as a certain probability distribution or ϵ-far from it), monotonicity testing, I/O models and MPC models.