Stat260/CS294: Randomized Algorithms for Matrices and Data

Lecture 8 - 09/30/2013

Lecture 8: Fast Random Projections and FJLT

Lecturer: Michael Mahoney *Scribe: Michael Mahoney*

Warning: these notes are still very rough. They provide more details on what we discussed in class, but there may still be some errors, incomplete/imprecise statements, etc. in them.

8 Fast Random Projections and FJLT

Today, we will discuss a particular form of random projections known as structured random projections or the FJLT that are "fast" in that one can use fast Fourier methods to apply them quickly to arbitrary or worst case input. We will be able to use this to speed up both random projection as well as random sampling RandNLA algorithms for a wide variety of problems. Here is the reading for today.

- Ailon and Chazelle, "The fast JohnsonLindenstrauss transform and approximate nearest neighbors"
- Matousek, "On variants of the Johnson-Lindenstrauss lemma"
- Drineas, Magdon-Ismail, Mahoney, and Woodruff, "Fast approximation of matrix coherence and statistical leverage"

Today we will focus on two things.

- An introduction to fast Fourier/Hadamard-base random projection methods.
- An introduction to how to use these methods for LS approximation.

8.1 Background on "fast" random projections methods

Let's start with the basic AC Hadamard-based FJLT. To set the context, recall that applying a random projection matrix consisting of i.i.d Gaussians or i.i.d. $\{\pm 1\}$ random variables to an arbitrary input vector take "matrix multiplication time," since we have to actually implement the random projection. (By this, we mean the time to perform an in general dense matrix-vector multiplication—which is *not* via Strassen-like algorithms, except for purely theoretical considerations.) For an $n \times d$ matrix, since this involves projecting with an $r \times n$ matrix, where $r \gtrsim d$, this is $\Theta(ndr)$ time, with the usual matrix-vector product methods. Unfortunately, this is at least as expensive as solving the original LS problem exactly, since this takes $\Theta(nd^2)$ time.

But while working with vanilla Gaussian-based random projections it might not be necessary. In particular, we saw that the point of preprocessing the input with a random projection is to make

the input data "nice," in that the eigenvector or singular vector mass is spread out among all the coordinates, in which case uniform sampling, sparse projections, etc. perform well. This leads to the question:

- Can we preprocess (or "precondition") the input data, so that the data are "nice" in the same or in a similar sense, but do it faster?

The answer is "Yes." There are a range of tradeoffs and details here, depending on what exactly is one's goal, e.g., best theory, best implementations, assumptions on the input, etc., but this opens the door to improved randomized matrix algorithms for a wide variety of problems. We will now turn to this topic.

To provide an overview of fast random projection methods, let's start with the following definition of a Johnson Lindenstrauss Transform, which is of much more general interest.

Definition 1 *Given an $\epsilon > 0$ and n points $\{x_i\}_{i=n}^n$, where $x_i \in \mathbb{R}^d$, an ϵ-JLT (an ϵ-Johnson Lindenstrauss Transform), denoted $\Pi \in \mathbb{R}^{r \times d}$ is a projection of the points into $\mathbb{R}^n$ such that*

$$(1-\epsilon)\ \|x_i\|_2^2 \leq \|\Pi x_i\|_2^2 \leq (1+\epsilon)\ \|x_i\|_2^2 .$$

JLTs are of interest in a wide range of algorithmic applications, basically since they provide a way to embed input data a discrete set of n data points into lower dimension without sacrificing too much in terms of distance information between pairs of those points.

In RandNLA, the notion of a JL Transform is usually not *directly* useful, since we are typically more interested in subspaces than in discrete sets of points. Fortunately, this definition can be generalized to all points in the subspace, and this is sometimes called SubspaceJL. The basic idea is to show that the usual JL result holds with exponentially high probability for each pair of points and then put an ϵ net on the unit ball.

Here is an important point. One can view this (and related) SubspaceJL result in one of two complementary ways.

- As an approximate matrix multiplication result applied to special input, where both matrices are an orthogonal matrix spanning the same space.
- As a generalization of the usual JL lemma from a finite set of vectors to a specially-structured infinite set of vectors.

One perspective of the other is more useful, depending on the situation.

Here, i.e., for the "fast" random projection methods we will discuss today and in the next few classes, we will be interested in the stronger requirement that we get similar JLT bounds, but in addition that we can compute the JLT quickly.

Definition 2 (Fast Subspace JL) *Given an $\epsilon > 0$ and an orthogonal matrix $U \in \mathbb{R}^{n \times d}$, viewed ad d vectors in $\mathbb{R}^n$. A FJLT projects vectors from $\mathbb{R}^n \to \mathbb{R}^r$ s.t.the orthogonality of U is preserved, and it does it quickly. I.e., $\Pi \in \mathbb{R}^{r \times n}$ is an ϵ-FJLT if*

- $\left\| I_d - U^T \Pi^T \Pi U \right\|_2 \leq \epsilon$

- $\forall x \in \mathbb{R}^{n \times d}$, *we can compute* Πx *in* $O(nd \log(r))$ *time.*

The original construction in this area is due to AC (although fast Fourier ideas have certainly existed much longer), but there are many others. Theoretically, they are all to a first approximation the same; but practically, there can be a big difference between them. Here is the original AC construction. (BTW, the notation is inconsistent with what we use above and below, as it is taken from the AC journal paper.) Let $\Pi = PHD$, where

- P is a sparse JL matrix or a uniform sampling matrix with a few extra dimensions. To make things specific, let $P \in \mathbb{R}^{n \times n}$ have elements

$$\Pi_{ij} = \begin{cases} 0 & \text{with probability } 1 - q \\ N(0, q^{-1}) & \text{with probability } q \end{cases},$$

  where $q = \min\{1, \Theta\left(\frac{\log^2(n)}{d}\right)\}$.

- H is structured, so that we can apply Fast Fourier methods to compute it quickly. Basically, it spreads out "spikey" vectors.

- D is a random $\{\pm 1\}$ matrix that basically is used to put bad cases, i.e., localizing delocalized vectors, into the failure probability.

Here are some notes on this AC construction.

- P is a very sparse matrix—in expectation, only a fraction of the elements are nonzero.

- One might hope to use P directly on x, but $\|Px\|_2$ is too large (to be a usual JL) for certain "bad" inputs, basically when x is a very sparse matrix, since then we don't get sufficient concentration.

- If x is "smooth," i.e., if the mass is roughly uniformly spread out, then $\|Px\|_2$ does get good concentration.

- The mapping HD ensures that HDx is smooth and not too "spikey."

- HD is an orthogonal matrix (exactly), meaning in particular that the Euclidean norm of vectors to which it is applied doesn't change.

- So, basically, HD preconditions x before we apply P.

Here, we will make precise the sense in which HD "spreads out" input vectors. (We basically take this particular form from the AC journal paper.)

Lemma 1 *Fix a set* X *of* n *vectors in* $\mathbb{R}^d$. *Then, with probability* $\geq 1 - \frac{1}{20}$, *we have that*

$$\max_{x \in X} \|HDX\|_\infty = O\left(\sqrt{\frac{\log(n)}{d}}\right)$$

Proof: Assume w.l.o.g. that $\|x\|_2 = 1$. Fix some $x \in X$, and define the random variable

$$u = HDx = (u_1, \ldots, u_d)^T .$$

Note that u_i is of the form $\sum_{i=1}^{d} a_i x_i$, where each $a_i = \frac{\pm 1}{\sqrt{d}}$ is chosen uniformly and independently. Then, we can apply a Chernoff argument in the usual way.

$$\begin{aligned} \mathbf{E}\left[e^{tdu_i}\right] &= \Pi_i \mathbf{E}\left[e^{tda_i x_i}\right] \\ &= \Pi_i \mathbf{E}\left[cosh(t\sqrt{d_i} x_i)\right] \\ &\leq \exp\left(t^2 d \|x\|_2^2 / 2\right) \end{aligned} \tag{1}$$

Hence, $\forall s > 0$, by applying Markov's Inequality and plugging $t = sd$ into (1), we have that

$$\begin{aligned} \mathbf{Pr}\left[|u_1|\right] &= 2\mathbf{Pr}\left[e^{2du_1} \geq e^{s^2 d}\right] \\ &\leq 2\mathbf{E}\left[e^{sdu_1}\right] / e^{s^2 d} \\ &\leq 2e^{s^2 d \|x\|_2^2 / 2 - s^2 d} \\ &= 2e^{-s^2 d/2} \\ &\leq \frac{1}{20nd}, \end{aligned}$$

for $s = \Theta\left(\sqrt{\frac{\log(n)}{d}}\right)$. By performing a union bound over all $nd < n^2$ coordinates of the vectors

$$\{HDx : x \in X\},$$

it follows that

$$\max_{x \in X} \|HDx\|_\infty = O\left(\sqrt{\frac{\log(n)}{d}}\right),$$

which establishes the lemma.

$\diamond$

Since the pre-processed or preconditioned vector is flat, in the sense made precise by that lemma, we can now do one of two things.

- Sample uniformly, and oversample a little bit since the uniform sampling probabilities are not exactly optimal.
- Apply very sparse random projections, which is sufficient to get concentration since the input vectors are flat.

The first of these is a sampling procedure, i.e., involves choosing only a single row, and the second of these involves taking a linear combination of a small number of rows. Nevertheless, when coupled with the HD preprocessing, both of these procedures achieve JL-type results and thus can be meaningfully interpret as performing random projections.

8.2 Applying these ideas to LS

Before describing these algorithms, let's start with a lemma that quantifies the manner in which HD uniformizes the information in the left singular subspace of A. Note that this is very similar to Lemma 1—basically, it applies Lemma 1 to orthogonal vectors that define the singular subspace of a given tall input matrix A.

Lemma 2 *Let $U \in \mathbb{R}^{n \times d}$ be an orthogonal matrix (spanning the column space of an $n \times d$ matrix A), and let HD be the $n \times n$ Randomized Hadamard Transform. Then, with probability ≥ 0.95, we have that*

$$\max_{i \in [n]} \left\| (HDU)_{(i)} \right\|_2^2 \leq \frac{2d \log(40nd)}{n}.$$

Proof: Following the above lemma, which states that, for a fixed $j \in [d]$ and a fixed $i \in [n]$,

$$\mathbf{Pr}\left[\left| \left(HDU^{(j)} \right)_i \right| > s \right] \leq 2e^{-s^2 n/2}$$

(Note that we have n and d reversed; ugh.) Let $s = \sqrt{2n^{-1} \log(40nd)}$. So, then we have that

$$\mathbf{Pr}\left[\left| \left(HDU^{(j)} \right)_i \right| \geq \sqrt{2n^{-1} \log(40nd)} \right] \leq \frac{1}{20nd}$$

From the standard union bound, this implies that with probability $\geq 1 - \frac{1}{20}$ we have that

$$\left| \left(HDU^{(j)} \right)_i \right| \geq \sqrt{2n^{-1} \log(40nd)}$$

$\forall i \in [n], j \in [d]$. Since

$$\left\| (HDU)_{(i)} \right\|_2^2 = \sum_{j=1}^{d} \left(HDU^{(j)} \right)_i^2 \leq \frac{2d \log(40nd)}{n}$$

the lemma then follows.

$\diamond$

Now, let's now apply these ideas in the context of LS. Here are three three related fast LS algorithms.

- Random projection.
  - Multiply by HD to approximately uniformize the leverage scores, and then sample uniformly.
  - Multiply by HD to approximately uniformze things, and then use sparse projection matrix like above.
- Random sampling.
  - Use FJLT (either of those two projection-based procedures) to compute approximations to the leverage scores and sample w.r.t. those approximations.

We will describe these results in more detail in the next two classes.