Stat260/CS294: Randomized Algorithms for Matrices and Data

Lecture 5 - 09/18/2013

Lecture 5: Matrix Multiplication, Cont.; and Random Projections

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.

5 Matrix Multiplication, Cont.; and Random Projections

Today, we will use the concentration results of the last few classes to go back and make statements about approximating the product of two matrices; and we will also describe an important topic we will spend a great deal more time on, i.e., random projections and Johnson-Lindenstrauss lemmas. Here is the reading for today.

- Dasgupta and Gupta, "An elementary proof of a theorem of Johnson and Lindenstrauss"
- Appendix of: Drineas, Mahoney, Muthukrishnan, and Sarlos, "Faster Least Squares Approximation"
- Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins"

5.1 Spectral norm bounds for matrix multiplication

Here, we will provide a spectral norm bound for the error of the approximation constructed by the `BasicMatrixMultiplication` algorithm. Recall that, given as input a $m \times n$ matrix A and an $n \times p$ matrix B, this algorithm randomly samples c columns of A and the corresponding rows of B to construct a $m \times c$ matrix C and a $c \times p$ matrix R such that $CR \approx AB$, in the sense that some matrix norm $||AB - CR||$ is small. The Frobenius norm bound we established before immediately implies a bound for the spectral norm, but in some cases we will need a better bound than can be obtained in this manner. Since, in this semester, we will only need a spectral norm bound for the spectial case that $B = A^T$, that is all that we will consider here.

Theorem 1 *Let $A \in \mathbb{R}^{m \times n}$, and consider approximating AA^T. Construct a matrix $C \in \mathbb{R}^{m \times c}$, consisting of c sampled and rescaled columns of A, with the `BasicMatrixMultiplication` algorithm, where the sampling probabilities $\{p_i\}_{i=1}^n$ satisfy*

$$p_i \geq \beta \frac{\left\|A^{(i)}\right\|_2^2}{\|A\|_F^2} \tag{1}$$

for all $i \in [n]$, and for some constant $\beta \in (0,1]$. Assume, for simplicity, that $\|A\|_2 \leq 1$ and $\|A\|_F^2 \geq 1/24$, let $\epsilon \in (0,1)$ be an accuracy parameter, and let

$$c \geq \frac{96\,\|A\|_F^2}{\beta\epsilon^2} \log\left(\frac{96\,\|A\|_F^2}{\beta\epsilon^2\sqrt{\delta}}\right). \tag{2}$$

Then, with probability $\geq 1-\delta$, we have that

$$\left\|AA^T - CC^T\right\|_2 \leq \epsilon.$$

In addition, if we set $\delta = 1$ in Eqn. (2), then

$$\mathbf{E}\left[\left\|AA^T - CC^T\right\|_2\right] \leq \epsilon.$$

XXX. CHECK DETAILS OF THAT LAST CLAIM. XXX. IS THIS THEOREM NOW ROBUST TO EXACTLY c VERSUS APPROXIMATELY c SAMPLING.

Before proving this theorem, here are a few things to note about it.

- The assumptions on the spectral and Frobenius norms of A are not necessary, but instead are only to simplify the expressions.
- The assumption on c is important. That is, whereas the Frobenius norm bound we discussed previously holds for any value of c (perhaps yielding weak results if c is too small), here we will need to set c to be at least the value of Eqn. (2) for the theorem to hold.
- We can have $\epsilon\,\|A\|_F$ on RHS by modifying the sampling comlexity. XXX. CHECK DETAILS AND MAKE SURE RHS NORM IS RIGHT. In particular, this can give the sampling complexity in terms of the stable rank. If we define the *stable rank* of a matrix A as $\mathrm{sr}(S) = \frac{\|A\|_F^2}{\|A\|_2^2}$, then $\mathrm{sr}(A) \leq \mathrm{rank}(A)$, and the stable rank is a more robust notion than the usual rank, and bounds parameterized in this way are sometimes of interest.
- We can generalize this to the product of two different matrices, and we get slightly weaker results for $\|AB - CR\|_2$.
- We formulate it this way since we will only need spectral norm bounds for approximating matrix products of the form AA^T and since we will use this theorem by setting, e.g., $\epsilon = 1/2$.

Proof: For the proof of this spectral norm bound, we will need a matrix concentration result. For convenience here within the proof, we will use a slightly different version of matrix concentration than we proved last time—in particular, one due to Oliviera, rather than Recht, which we established last time. Here, we will simply state that version—it's proof is similar to the version we proved last time and so will be omitted. For completeness, though, here is a brief history of matrix concentration bounds and how they are used in randomized numerical linear algebra.

- Alshwede-Winter: the original result related to bounding the matrix m.g.f. that started this recent flurry of work in this area.
- Christofides-Markstron: introduced a matrix version of Heoffding-Azuma

- Rudelson and Vershynin: had the original bounds for operator-valued random variables that were originally used to bound the spectral norm error. They bounds had a similar form, but they depended on heavier-duty arguments from convex analysis, and they sometimes didn't provide constants, which made it awkward for numerical implementation.
- Gross, Recht, and Oliviera: several different versions of matrix Chernoff bounds.
- Tropp: provides a nice review of this line of work.

In this proof, we will use the version due to Oliviera (in "Sums of random Hermitian matrices and an inequality by Rudelson"), which we will state but not prove.

Lemma 1 *Let $X_1, \ldots, X_n$ be i.i.d. random column vectors in $\mathbb{R}^d$ such that $\|X_i\|_2 \leq M$ a.s. and $\|\mathbf{E}[X_i X_i^*]\|_2 \leq 1$. Then, $\forall t \geq 0$,*

$$\mathbf{Pr}\left[\left\|\frac{1}{n}\sum_{i=1}^{n} X_i X_i^* - \mathbf{E}[X_1 X_1^*]\right\|_2 \geq t\right] \leq (2n)^2 \exp\left(\frac{-nt^2}{16M^2 + 8M^2 t}\right)$$

To use this in the proof of our spectral norm matrix multiplication bound, consider the random sampling algorithm, and note that

$$AA^T = \sum_{i=1}^{n} A^{(i)} A^{(i)^T}.$$

We shall view the matrix AA^T as the true mean of a bounded operator valued random variable, whereas $CC^T = AS(AS)^T = ASS^T A^T$ will be its empirical mean. Then, we will apply Lemma 1 of [**?**]. To this end, define a random vector $y \in \mathbb{R}^m$ as

$$\mathbf{Pr}\left[y = \frac{1}{\sqrt{p_i}} A^{(i)}\right] = p_i$$

for $i \in [n]$. The matrix $C = AS$ has columns $\frac{1}{\sqrt{c}} y^1, \frac{1}{\sqrt{c}} y^2, \ldots, \frac{1}{\sqrt{c}} y^c$, where $y^1, y^2, \ldots, y^c$ are c independent copies of y. Using this notation, it follows that

$$\begin{aligned} \mathbf{E}\left[yy^T\right] &= \sum_{i=1}^{n} p_i \frac{1}{\sqrt{p_i}} A^{(i)} \frac{1}{\sqrt{p_i}} A^{(i)^T} \\ &= \sum_{i=1}^{n} A^{(i)} A^{(i)^T} \\ &= AA^T \end{aligned} \tag{3}$$

and

$$CC^T = ASS^T A^T = \frac{1}{c}\sum_{t=1}^{c} y^t y^{t^T}.$$

Finally, let

$$M = \|y\|_2 = \frac{1}{\sqrt{p_i}} \left\|A^{(i)}\right\|_2 \leq \frac{1}{\sqrt{\beta}} \|A\|_F, \tag{4}$$

where the inequality follows by the form of Eqn. (1). We can now apply Lemma 1, p. 3 of [**?**]. Notice that from Eqn. (3) and our assumption on the spectral norm of A, we immediately get that

$$\left\|\mathbf{E}\left[yy^T\right]\right\|_2 = \left\|AA^T\right\|_2 \leq \|A\|_2 \left\|A^T\right\|_2 \leq 1.$$

Then, Lemma 1 of [**?**] implies that

$$\left\|CC^T - AA^T\right\|_2 < \epsilon, \tag{5}$$

with probability at least $1 - (2c)^2 \exp\left(-\frac{c\epsilon^2}{16M^2+8M^2\epsilon}\right)$. Let δ be the failure probability of Theorem 1. We seek an appropriate value of c in order to guarantee that $(2c)^2 \exp\left(-\frac{c\epsilon^2}{16M^2+8M^2\epsilon}\right) \leq \delta$. Equivalently, we need to satisfy

$$\frac{c}{\ln\left(2c/\sqrt{\delta}\right)} \geq \frac{2}{\epsilon^2}\left(16M^2 + 8M^2\epsilon\right).$$

Recall that $\epsilon < 1$, and by combining Eqn. (1) with the above equation, it suffices to choose a value of c such that

$$\frac{c}{\ln\left(2c/\sqrt{\delta}\right)} \geq \frac{48}{\beta\epsilon^2}\|A\|_F^2,$$

or, equivalently,

$$\frac{2c/\sqrt{\delta}}{\ln\left(2c/\sqrt{\delta}\right)} \geq \frac{96}{\beta\epsilon^2\sqrt{\delta}}\|A\|_F^2.$$

We now use the fact that for any $\eta \geq 4$, if $x \geq 2\eta\ln\eta$ then $\frac{x}{\ln x} \geq \eta$. Let $x = 2c/\sqrt{\delta}$, let $\eta = 96\|A\|_F^2 / \left(\beta\epsilon^2\sqrt{\delta}\right)$, and note that $\eta \geq 4$ if $\|A\|_F^2 \geq 1/24$, since β, ϵ, and δ are at most one. Thus, it suffices to set

$$\frac{2c}{\sqrt{\delta}} \geq 2\frac{96\|A\|_F^2}{\beta\epsilon^2\sqrt{\delta}} \ln\left(\frac{96\|A\|_F^2}{\beta\epsilon^2\sqrt{\delta}}\right),$$

which concludes the proof of the theorem.

◇

5.2 Random projections and Johnson-Lindenstrauss lemmas

Here, we will discuss a related way to perform dimensionality reduction on matrices known as random projections, which has strong connections with an important result known as the Johnson-Lindenstrauss lemma. Random projections, and in particular the results provided by the JL lemma are very powerful, in the sense that the points can be in more general metric spaces, etc. Thus, they have strong conenctions with random sampling and matrix multiplication, which we will make explicit. The reason we have started with the latter is that since they make more explicit the Euclidean vector space structure, which in turn allows us to get finer bounds that are more useful for numerical implementations, machine learning and data analysis applications, etc.

The general question is one of so-called *dimensionality reduction*, i.e., mapping a high-dimensional data set (i.e., a set of data points modeled as vectors in some high-dimensional, typically but

not always, Euclidean vector space) to a much lower-dimensional space (again, typically, a low-dimensional Euclidean vector space) in such a way that important structural properties of the data are preserved. The most common way to do this, e.g., in statistics and machine learning and many other related areas, involves choosing a small number of directions in the original vector space in which the data have high variance. That is, if one looks at the data and one asks "What are the directions in the high-dimensional space that capture the most amount of variance?" then one is interested in finding those directions.

The most common way to do that is with the SVD, or relatedly PCA. The SVD, and as a practical matter partial SVDs, i.e., computing a small number of singular valeus and singular vectors, as opposed to the full SVD, is moderately byt not extremely expensive to compute, and it is useful in many situations where it is not obviously-appropriate, e.g., where Gaussian-like assumptions underlying its or truncated PCAs are violated. A lot of what we will be interested in later in the course will be computing partial SVDs more quickly than off-the-shelf methods, and we will use random sampling and random projections to do this. Although there are strong connections between random projections and the properties of the SVD that involve capturing maximum variance directions, it is actually useful to take a step back and consider other types of dimensionality reduction methods. Random projections are most-easily viewed this way, and so we will start with that, and we will make the connections with SVD later. To that end, consider a different type of dimensionality reduction where, rather than finding the directions that capture the most variance, the goal is to construct some sort of mapping that preserves all $\binom{n}{2}$ pairwise distances between pairs of data points.

(As we said, there will be connections between preserving pair-wise distances and capturing variance, but at this point just note that they are two different metrics of interest. For example, one can easily imagine that maximizing variance preserves all the pairwise distances "on average," but that a few pairwise distances are violated a lot; and conversely that if we force ourselves to preserve every pairwise distance, then we might "overfit" the data at hand and fail to preserve a large-scale measure like overall variance. We will return to this later, but you should think of the two different persepctives that we mentioned in the first class: preserving every pairwise distance is more natural from the algorithmic perspective, where we view the data in front of us as all there is, in which case we want to preserve the properties on it, and we get worst-case quality of approximation bounds that depend on the worst-case distortion; while preserving overall variance might involve sacrificing a few pairwise distances and might be more robust in the presence of a bit of noise, and so this might be associated with better inferential properties.)

Today, we will consider a very simple but remarkably powerful method to do dimensionality reduction that is of the latter flavor, and we will describe the results one can prove and the analysis, which go by the name the Johnson-Lindenstrauss lemma. We won't use Chernoff bounds directly, and so we won't call our previous results as a black box, but we will draw connections later, and we will see that the proof will use ideas that are very similar to the proofs of Chernoff bounds.

Say that we have n points $\{u_i\}_{i=1}^n$, each of which is in $\mathbb{R}^d$, e.g., the rows of an $n \times d$ matrix A, and we want to find n points $\{v_i\}_{i=1}^n$, each of which is in $\mathbb{R}^k$ such that

- $k \ll d$
-
  - $||v_i|| \approx ||u_i||$, for all i
  - $||v_i - v_{i'}|| \approx ||u_i - u_{i'}||$, for all i, i'

Here, $||\cdot||$ refers to the Euclidean norm, i.e., $\|\cdot\|_2$ in the above notation. The first thing to note is that it isn't immediately obvious that such a mapping even exists, nevermind that it can be computed efficiently and exploited algorithmically. We will show that such a mapping does exist for the Euclidean norm, but it is known that such a mapping does not exist for other norms, e.g., the ℓ_1 norm. Since the proof of this result is now sufficiently simple to be presented in a class, it is worth paying attention to what steps are standard and which steps are peculiar to the Euclidean norm and fail to hold for other norms.

We will construct a "random projection," and prove a version of the Johnson-Lindenstrauss lemma. Here is a brief history of these methods.

- Johnson-Lindenstrauss: proved the result for a random subspace as part of a more general result they were interested in proving.
- Frankl and Meahara: project onto k random orthogonal vectors.
- IM, DG: project onto a matrix whose entries consist of i.i.d. Gaussian random variables.
- Ach: project onto a matrix consisting of $\{\pm 1\}$ random variables.
- Ailon and Chazelle: construct a "fast" Hadamard-based version of JL, in which case the projection matrix can be multiplied more quickly than vanilla matrix multiplication using fast-Fourier methods.
- Sarlos: made explicit the "subspace JL" result, basically by putting an ϵ-net on a unit ball, to show that JL-like bounds hold for infinitely many vectors drawn from a low-dimensional subspace, thereby yielding a distortion bound of the form we saw with approximate matrix multiplication.
- Clarkson and Woodruff: the random projection matrix can be *extremely* sparse and one can get JL-like results, assuming that the input vectors are from a low-dimensional subspace

Here, we will prove the version for Gaussian random variables: although it is easier than some of the other versions, similar ideas hold for the other versions, and we will revisit some of the other versions later.

Before doing that, here is a word about terminology. In linear algebra and functional analysis, a projection is a linear transformation P from a vector space to itself such that $P^2 = P$. In particular, it is idempotent and its eigenvalues are in $\{0, 1\}$, i.e., equal to either 0 or 1. For example, given a matrix A, let $A = U\Sigma V^T$ be its truncated SVD, and let $A = QR$ be its QR decomposition. Then, the projection onto the column space of A is $P_A = UU^T = QQ^T$, and it is easy to verify that $P_A = Q(Q^TQ)Q^T$, since Q^TQ is a low-dimensional identity.

The JL "projection" is more general and is typically not a projection in that linear algebraic sense of the word. (Although it is ϵ-close to a projection in that traditional linear algebraic sense of the word, and quantifying this observation is at the heart of the analysis of many of the random sampling and random projection algorithms in RandNLA.) Why is it not a projection in that sense of the word?

- First, it can be applicd to arbitrary points in arbitrary metric spaces. This makes it applicable more generally than Euclidean vector spaces, but it makes its use slightly overkill form many data analysis and machine learning problems that involve matrices in $\mathbb{R}^n$.

- Second, in spite of that, it "looks like" an orthogonal matrix in many ways. For example, if P is a matrix of i.i.d. Gaussians, then range(P^TP) is a uniformly distributed subspace, but the eigenvalues are *not* in $\{0,1\}$.
- Third, if the random vectors were exactly orthogonal (as they actually were in the original JL constructions), then we would have that the JL projection was an orthogonal projection and the eigenvalues would be in $\{0,1\}$; but although this is false for Gaussians, $\{\pm 1\}$ random varialbes, and most other constructions, one can prove that the resulting vectors are approximately unit length and approximately orthogonal, and for most applications of the JL lemma in RandNLA (as well as more generally), this is "good enough."
- Fourth, for $\{\pm 1\}$, Hadamard, etc. constructions, they are *not* even spherically symmetric, so the analysis is messier, but we will be able to show that they lead to JL projections that are almost orthognal matrices and thus projections in the linear algebraic sense of the word. But the analysis is messier.

With those comments in place, here is the version of the JL lemma that we will prove in detail.

Lemma 2 (JL lemma) *Given n points $\{u_i\}_{i=1}^n$, each of which is in $\mathbb{R}^d$, $P \in \mathbb{R}^{d\times k}$ be such that $P_{ij} = \frac{1}{\sqrt{k}}N(0,1)$, and let $\{v_i\}_{i=1}^n$ be points in $\mathbb{R}^k$ defined as $v_i = u_iP$. Then, if $k \geq \frac{9\log(n)}{\epsilon^2-\epsilon^3}$, for some $\epsilon \in (0,1/2)$, then with probability at least $1/2$, all pairwise distances are preserved, i.e., for all i, i', we have*

$$(1-\epsilon)\left\|u_i - u_i'\right\|_2^2 \leq \left\|v_i - v_i'\right\|_2^2 \leq (1+\epsilon)\left\|u_i - u_i'\right\|_2^2.$$

Proof: Let u be any fixed vector in $\mathbb{R}^n$, and consider $v = uP$. We will establish results for the expectation of the norm of v as well as results that state with high probability the norm does not deviate much above or below the expectation. Then, we will set parameters to get that a union bound argument means that the result holds for $\binom{n}{2}$ pairs of points with probability at least $1/2$. First, let's get the expectation.

$$\begin{aligned}
\mathbf{E}\left[\|v\|_2^2\right] &= \mathbf{E}\left[\sum_{j=1}^k\left(\sum_{i=1}^d \frac{1}{\sqrt{k}}u_iP_{ij}\right)^2\right] \\
&= \sum_{j=1}^k \frac{1}{k}\mathbf{E}\left[\left(\sum_{i=1}^d u_iP_{ij}\right)^2\right] \\
&= \sum_{j=1}^k \frac{1}{k}\sum_{i=1}^d u_i^2\mathbf{E}\left[P_{ij}^2\right] \\
&= \sum_{j=1}^k \frac{1}{k}\sum_{i=1}^d u_i^2 \\
&= \|u\|_2^2
\end{aligned}$$

Most of the equalities are fairly straightforward, but note that the third equality follows since $P_{ij} \sim N(0,1)$; for other JL constructions, establishing this is more complicated, but it can be done.

(Note that this derivation basically says that if you take an arbitrary unit-length vector in $\mathbb{R}^d$ and "project" it down to $\mathbb{R}^k$ by taking random linear combinations, weighted by $N(0,1)$ random

variables, then the squared length of the resulting vector is $\frac{k}{d}$, and thus to preserve the length, we have to rescale by $\frac{d}{k}$.)

Next, let's bound the probability that the projected vector stretched by more than a small amount from the expectation. To do so, let's define $x_j = \frac{1}{\|u\|_2} u^T P^{(j)}$ and $x = k\frac{\|v\|_2^2}{\|u\|_2^2} = \frac{1}{\|u\|_2^2}\sum_{j=1}^k (u^T P^{(j)})^2 = \sum_{j=1}^k x_j^2$. (Note that this notation is inconsistent with the usual way we use subscripts, but we will use it only here in a self-contained way.) With these definitions, we have that

$$\begin{aligned}
\mathbf{Pr}\left[\|v\|_2^2 \geq (1+\epsilon)\|u\|_2^2\right] &= \mathbf{Pr}\left[x \geq (1+\epsilon)k\right] \\
&= \mathbf{Pr}\left[e^{\lambda x} \geq e^{\lambda(1+\epsilon)k}\right] \\
&\leq \frac{\mathbf{E}\left[e^{\lambda x}\right]}{e^{\lambda(1+\epsilon)k}} \\
&= \frac{\mathbf{E}\left[e^{\lambda \sum_{j=1}^k x_j^2}\right]}{e^{\lambda(1+\epsilon)k}} \\
&= \frac{\Pi_{j=1}^k \mathbf{E}\left[e^{\lambda x_j^2}\right]}{e^{\lambda(1+\epsilon)k}} \\
&= \left(\frac{\mathbf{E}\left[e^{\lambda x_1^2}\right]}{e^{\lambda(1+\epsilon)}}\right)^k
\end{aligned}$$

where the second equality follows for all $\lambda > 0$ to be chosen later, and the rest of the steps should be clear, based on how we derived the Chernoff bounds. To calculate $\mathbf{E}\left[e^{\lambda x_1^2}\right]$, recall that $x_1 \sim N(0,1)$, and thus

$$\begin{aligned}
\mathbf{E}\left[e^{\lambda x_i^2}\right] &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} e^{-\lambda t^2} dt \\
&= \frac{1}{\sqrt{1-2\lambda}} \int_{-\infty}^{\infty} \frac{\sqrt{1-2\lambda}}{\sqrt{2\pi}} e^{\frac{t^2}{2}(1-2\lambda)} dt \\
&= \frac{1}{\sqrt{1-2\lambda}},
\end{aligned}$$

for $\lambda < \frac{1}{2}$. Plugging this into the above, we have that

$$\begin{aligned}
\mathbf{Pr}\left[\|v\|_2^2 \geq (1+\epsilon)\|u\|_2^2\right] &= \left(\frac{e^{-2\lambda(1+\epsilon)}}{1-2\lambda}\right)^{k/2} \\
&= \left((1+\epsilon)e^{-\epsilon}\right)^{k/2} \\
&\leq \exp\left(-\left(\epsilon^2 - \epsilon^3\right)k/4\right)
\end{aligned}$$

where the second equality follows if we choose $\lambda = \frac{\epsilon}{2(1+\epsilon)}$ and the inequality follows since $1+\epsilon < \exp\left(\epsilon - \left(\epsilon^2 - \epsilon^3\right)/2\right)$.

Third, let's bound the probability that the projected vector is shrinked by more than a small amount from the expectation. The derivation is similar to above derivation, and so we just state

the main steps of it.

$$\begin{aligned}
\mathbf{Pr}\left[\|v\|_2^2 \leq (1-\epsilon)\|u\|_2^2\right] &= \mathbf{Pr}\left[x \leq (1-\epsilon)k\right] \\
&= \mathbf{Pr}\left[e^{-\lambda x} \geq e^{-\lambda(1-\epsilon)k}\right] \\
&\leq \left(\frac{\mathbf{E}\left[e^{-\lambda x_1^2}\right]}{e^{-\lambda(1-\epsilon)}}\right)^k \\
&= \left(\frac{e^{2\lambda(1-\epsilon)}}{1+2\lambda}\right)^{k/2} \quad \text{for } \lambda = \frac{\epsilon}{2(1-\epsilon)} \\
&= \left((1-\epsilon)\,e^{\epsilon}\right)^{k/2} \\
&\leq \exp\left(-\left(\epsilon^2-\epsilon^3\right)k/4\right)
\end{aligned}$$

Finally, let's put it all together. By combining the above two results, we have that for any one fixed point u that is mapped to v, we have that

$$\mathbf{Pr}\left[(1-\epsilon)\|u\|_2^2 \leq \|v\|_2^2 \leq (1+\epsilon)\|u\|_2^2\right] \geq 1 - 2\exp\left(-\left(\epsilon^2-\epsilon^3\right)k/4\right)$$

Since we have n points, we want this type of bound to hold for $\binom{n}{2} = \Theta(n^2)$ pairs of points, i.e., for the vectors defining the distances between each of the pairs of points. If we want the failure probability to be $\leq \frac{1}{2}$, then by the union bound, we need that

$$2n^2 \exp\left(-\left(\epsilon^2-\epsilon^3\right)k/4\right) < \frac{1}{2}.$$

That is, we need that

$$\left(\epsilon^2-\epsilon^3\right)k/4 > 2\log(2n),$$

and for this it suffices that $k > \frac{9\log(n)}{\epsilon^2-\epsilon^3}$, under the simplifying assumption that $n > 16$.

$\diamond$

5.3 Additional comments and thoughts on random projections

Before proceeding, here are a few remarks on the proof of the JL lemma.

- One can view the n points $\{u_i\}_{i=1}^n$ as the rows of an $n \times d$ matrix A. In this case, if $u_i = A_{(i)}$ is the i^{th} row of A, then $v_i = (AP)_{(i)}$ is the i^{th} row of the product matrix AP.
- The failure probability of $1/2$ is for convenience; it can be made to be less that δ, for any fixed $\delta > 0$ by adjusting the dimension k to be slightly larger.
- All proof of JL lemmas have the same basic structure that we followed above. Basically, one proves that the JL mapping doesn't distort any one fixed vector too much away from its expectation, and then one performs a union bound to show that the result holds for $\binom{n}{2}$ pairs of vectors—that can be chosen to be the differences between all $\binom{n}{2}$ difference vectors $u_i - u_{i'}$. (This is basically just the "random projection theorem" mentioned below.

- The dependence on k in this lemma is simplified for convenience, e.g., the 9 is suboptimal. But the dependence on n is optimal, and the dependence on ϵ is optimal up to $\log(1/\epsilon)$ factors.

- One of the main points is that the dependence on k is logarithmic in n, the number of data points, but it is independent of d, the formal dimensionality of those data points. In some of what follows, we will be projecting on the left, and sometimes we will be projecting on the right. In addition, if we have a "rectangular" matrix, by which we mean that one dimension is much larger than the other, we will usually be projecting on the high dimension to make it smaller, but sometimes we will project on the low dimension to make it even lower. Plus, in some cases, we will project a subspace, and so we will actually preserve distances among an uncountably infinite number of pairs of points, basically by using an ϵ net argument. For these reasons, as well as the fact that we will use different letters to denote the dimensionality of different matrices, one might to loose sight of the simple and important point that random projections will allow us to project some number of points to a dimension that depends logarithmically on the number of original points and is independent of thier formal dimensionality.

Next we will spend a bit of time describing, more informally, what is going on with the JL lemma and what makes it work, with an eye toward topics that we will return to throughout the semester.

If we consider n points in $\mathbb{R}^d$, as in the JL setup, and we want to put them in $\mathbb{R}^k$, for some $k \ll d$, the obvious naive way to do that is to choose k coordiantes u.a.r. and evaluate pairwise distances based on those k coordinates and hope for the best. Although naive, that procedure actually works for certain classes of input vectors that are "spread out" in ways that we will quantify later. The question is: which are properties of input vectors that will cause problems for this naive procedure? Basically, the answer is that two points can be very far apart in Euclidean distance, while differing on only a small number of coordinates, or even a single coordiante. In this case, if we sample uniformly, then with high probability we will not select those coodrinates that are important for maintaining pairwise distances. On the other hand, if for all pairs of points, it is the case that all coordinates contribute roughly the same amount then this simple procedure works.

(Note that essentially this same issue arose in the motivation of the random sampling algorithm—there might be a small number of rank-one components that wre particularly important, and if we don't find them, then we will get a very poor approximation to the matrix product. Before the solution was to sample nonuniformly. Here we will find a different solution. And later we will discuss connections between these two approaches, illustrating how the complement one another.)

Motivated by this, the basic idea underlying random projection algorithms is that, given n points in $\mathbb{R}^d$, rather than sampling in the canonical basis, if we instead apply some sort of "random rotation" to the original point set, then we will get a new "random basis" in $\mathbb{R}^d$, and in that basis things will be "spread out" in nice ways, so we can sample uniformly. To see this, note that choosing the first k coordinates applying a random rotation is exactly the same as projecting onto a spherically-symmetric k-dimensional hyperplane. So, essentially, randomization in the JL projection protects against problems with axis-alignment in the canonical basis—which is exactly what the nonuniform sampling probabilities ensured against. If, on the other hand, the input are well-spread-out, then: either we can have very sparse random projection matrices; or we can perform uniform sampling.

As we mentioned above, JL actually projected onto a random k-dimensional hyperplane. FM did something similar, but viewed it as projecting onto k orthonormal vectors. If P or Π is a proejction

matrix that is a $d \times k$ real-valued matrix, then we can construct such a matrix as follows: for the first row, choose a random uniform vector w.r.t. the Haar measure on the sphere S^{d-1}, the $(d-1)$-dimensional unit sphere in $\mathbb{R}^d$ (to choose a random unit vector in $\mathbb{R}^d$, choose d i.i.d. $N(0,1)$ random variables, and normalize the resulting vector to have Euclidean norm 1); then the second row is a random unit vector from the space orthogonal to the first vector; and so on. The resulting matrix is a projection onto a random k-dimensional subspace of $\mathbb{R}^d$ with the following properties: spherical symmetry (i.e., for all orthodonal matrices U, P and PU have the same distribution); orthogonality (i.e., the columns of P are orthogonal to each other); normality (i.e., the columns of P are unit length).

And what are random subspaces? A random subspace of dimension equal to 1 is just a random line through the origin. A random subspace of dimension equal to k is specified by a random line through the origin, a second random line through the origin orthogonal to the first, etc., and their span is the random subspace of interest.

How long are vectors when you project them onto random subspaces? Let's say we project a fixed unit-length vector u in $\mathbb{R}^d$ onto a random k-dimensional subspace, constructed as above. Then, by the Pythagorean Theorem, the length squared of the new vector is the sum of the lengths squared of the components. Intuitively, in a random direction the squared length of an arbitrary unit length vector should be $\sim \frac{1}{d}$ (meaning exactly or approximately that, e.g., that in expectation and with high probability very near that). In this case, the squared length of a projection onto a random k-dimensional subspace should be $\sim \frac{k}{d}$. So, the length or norm of the projected vector should be $\sim \sqrt{\frac{k}{d}}$. The following theorem says that this is the case—in particular, it says that the length of the projected vector is very close to this expectation, with a failure probability that is exponentially small in k.

Theorem 2 (Random Projection Theorem) *Let u be a fixed unit-length vector in $\mathbb{R}^d$, let V be a random k-dimensional subspace, and let v be the projection of u onto V. Then, for all $\epsilon \in (0,1)$, we have that*

$$\mathbf{Pr}\left[\left|\,\|v\|_2 - \frac{k}{d}\right| \geq \epsilon\sqrt{\frac{k}{d}}\right] \leq 3\exp\left(-k\epsilon^2/64\right).$$

Clearly, given this random projection theorem, the simplest form of the JL lemma follows immediately by a union bound argument. As for other forms of the JL lemma—and in particular those that are algorithmically more appealing like when P consists of i.i.d. Gausasians, $\{\pm 1\}$ random variabales, or structured Hadamard-based transforms—similar ideas also hold. Basically, things are not as smooth or nice as with random subsapces but similar concentration results can be shown to hold.

IM and DG, in particular, dropped the explicit requirement of normality and orthogonality. But, if we choose every element of P i.i.d. $\sim N(0,1)$, as we did in the above construction, then we get normality and orthogonality in expectation, i.e., the squared length of all columns is 1 in expectation, and the inner product between all pairs of columns is 0 in expectation. This p is still symmetric, but the independence of entries makes it easier to generate.

(That last statement is actually true in theory, but not necessarily in practice. That is, if you try to implement a random projection by multiplying an input matrix with i.i.d. Gaussians, then the most expensive step can be generating the random variables, motivating either more sophisticated lower-level methods to generate random variables from Gaussian distributions or random projections

with strucutrally simpler entries. Both have been used in RandNLA, and here we will focus on the latter.)

Ach considered entried that were basically $\{\pm 1\}$ random variables, thus dropping the spherical symmetry constraint. In particaular, choosing $\{\pm 1\}$ random variables, up to rescaling, means working with one of the following two distributions. First, let

$$P_{ij} = \begin{cases} 1/\sqrt{k} & \text{w.p.} = 1/2 \\ -1/\sqrt{k} & \text{w.p.} = 1/2 \end{cases}$$

and second, let

$$P_{ij} = \begin{cases} 3/\sqrt{k} & \text{w.p.} = 1/6 \\ 0 & \text{w.p.} = 2/3 \\ -3/\sqrt{k} & \text{w.p.} = 1/6 \end{cases}$$

Note the scaling in these two means that things work out in expectation, and the analysis shows that they work out with high probability, in a manner similar to what we did above. Working with $\{\pm 1\}$ random varialbes means that you can construct the entries of P with one random bit per entry, which is easier to generate since flipping coins is easier than generating Gaussians, and it has other advantages that we won't get into. In addition, the second distribution above (and the analysis, which we won't get into, which says that it satisfies JL-like properties) shows that we can work with projection matrices that are somewhat sparse. In this case, 2/3 of the entries can be zero-ed out, and in worst case not much more can be zero-ed out, at least in this very simple way, but it raises the question of whether and how one can sparsity even more.

Two other extensions that we will get to in a few classes are the following.

- Fast JL: proposed by Ailon and Chazelle, this makes these random projection ideas useful for even relatively-quick computations like least-squares regression and many related matrix problems.
- Subspace JL: originally made explicit by Sarlos, this allows us to make statements about an entire subspace of vectors, which will allow us to make statements analogous to that the approximate matrix multiplication bounds say.

5.4 A random projection algorithm for approximating matrix multiplication

Recall the `BasicMatrixMultiplication` algorithm and that, using the sampling matrix formalism, we can write the output of it and what it is doing as $C = ASD$, $R = (SD)^T B$, and

$$CR = ASD(SD)^T B = A\mathcal{S}\mathcal{S}B \approx AB,$$

where $\mathcal{S} = SD$ is just a way to absorb the diagonal rescaling matrix into the sampling matrix.

With this suggestive notation, here is a random projection algorithm for approximating matrix multiplication. Given as input an $m \times n$ matrix A, an $n \times p$ matrix B, a positive integer c, do the following.

1. Let Π be an $n \times c$ random projection matrix, as defined above.
2. Let $C = A\Pi$ and $R = \Pi^T B$ be sketches of the columns of A and rows of B.

3. Compute and return $CR = A\Pi\Pi^T B$.

Depending on how we choose parameters, we can establish similar quality-of-approximation results with this procedure as we saw before. That is, the matrix multiplication primitive can be done in one to two ways.

- In a data-aware manner, in which we perform random sampling with sampling probabilities that depend on the input matrices.
- In a data-agnostic manner, in which we perform random projections without looking at the input data.

Both of these approaches will be useful later.