

RandNLA: Randomization in Numerical Linear Algebra: Theory and Practice

Petros Drineas
RPI

Ilse Ipsen (organizer)
NCSU

Michael W. Mahoney
UC Berkeley

To access our web pages use your favorite search engine.



Why RandNLA?

Randomization and sampling allow us to design provably accurate algorithms for problems that are:

➤ *Massive*

(matrices so large that can not be stored at all, or can only be stored in slow memory devices)

➤ *Computationally expensive* or NP-hard

(combinatorial optimization problems, such as the Column Subset Selection Problem)



RandNLA in a slide

Randomized algorithms

- By (carefully) **sampling rows/columns of a matrix**, we can construct new, smaller matrices that are close to the original matrix (w.r.t. matrix norms) with high probability.

Example:
Randomized
Matrix
Multiplication

$$\begin{pmatrix} A \end{pmatrix} \cdot \begin{pmatrix} B \end{pmatrix} \approx \begin{pmatrix} C \end{pmatrix} \cdot \begin{pmatrix} R \end{pmatrix}$$

- By **preprocessing the matrix using "random projection" matrices**, we can sample rows/columns much less carefully (uniformly at random) and still get nice bounds with high probability.

Matrix perturbation theory

- The resulting smaller matrices behave similarly (e.g., in terms of singular values and singular vectors) to the original matrices thanks to the norm bounds.



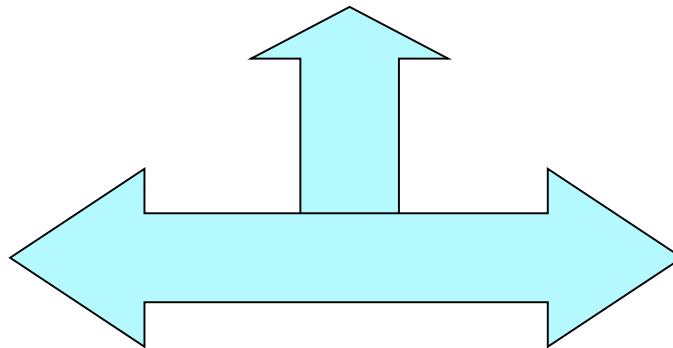
Interplay

Applications in BIG DATA

(Data Mining, Information Retrieval,
Machine Learning, Bioinformatics, etc.)

Theoretical Computer Science

Randomized and approximation
algorithms



Numerical Linear Algebra

Matrix computations and linear
algebra (ie., perturbation theory)



Tutorial roadmap

(**Petros**) discrete probability, randomized matrix multiplication

(**Ilse**) randomized Gram matrix multiplication, least-squares problems

(**Michael**) RandNLA practice: implementations and evaluations

RandNLA work will also be covered in **Haim Avron's plenary talk** on "Sketching-Based Matrix Computations for Large-Scale Data Analysis."

(IP3, Monday October 26, 1:45 pm-2:30 pm)



Intro to discrete probability theory

Let's start with discrete probability.

I will actually skip most of the (fairly introductory) material in the next several slides and will focus on measure concentration inequalities that are particularly useful in the analysis of RandNLA algorithms.

Overview

- Probabilistic experiments (definition, state space, events, independence, probabilities, conditional probabilities, union bound, etc.)
- Random variables (definition, independence, expectation, variance, etc.)
- Measure concentration inequalities (Markov, Chebyshev, Bernstein/Chernoff/Hoeffding, etc.)



Random or probabilistic experiment

A random experiment is any procedure that can be infinitely repeated and has a well-defined set of possible outcomes.

Sample space Ω : the *finite* set of all possible outcomes of the random experiment.

Bernoulli trial: a random experiment with exactly two possible outcomes (success and failure)

Event: any subset of the sample space Ω .

Example: let the random experiment be: "toss a coin three times". The sample space is

$\Omega = \{HHH, HHT, HTH, HTT, THH, THT, TTH, TTT\}$.

The event E described by "the output of the experiment was either all heads or all tails" is the set $\{HHH, TTT\}$.



Probability

Probability: the probability measure or probability function Pr mapping the *finite* sample space to the interval $[0,1]$, e.g., the function Pr mapping every element ω in Ω to $0 \leq \text{Pr}(\omega) \leq 1$.

Additional property (finite Ω): $\sum_{\omega \in \Omega} \text{Pr}(\omega) = 1$

Uniform probability: all elements of the sample space have the same probability.

Let A be an event (a subset of Ω). The probability of A is: $\text{Pr}(A) = \sum_{\omega \in A} \text{Pr}(\omega)$

Obviously, $\text{Pr}(\emptyset) = 0$
 $\text{Pr}(\Omega) = 1$.



Discrete vs. continuous probability

Countably infinite sample spaces Ω must be treated with some care; one can define the discrete probability space to be the set of all possible subsets of Ω , i.e., the powerset of Ω .

In this case, the probability measure or probability function P_r is defined over all possible events and satisfies the so-called positivity, normalization, and additivity properties.

If the set of all possible events is a proper subset of the powerset of Ω or if Ω is uncountably infinite, even more care is necessary; this is not discrete probability any more.



Properties of events

Events are sets, thus set operations (union, intersection, complementation) are applicable.

Examples:

$$\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B)$$
$$\Pr(\bar{A}) = 1 - \Pr(A).$$

The first property follows from the inclusion-exclusion principle, which can be generalized to more than two sets and thus to more than two events.

Additionally, if A is a subset of B , then $\Pr(A) \leq \Pr(B)$.



Union bound

Union bound: for any set of events $A_i, i=1\dots n$,

$$\Pr \left(\bigcup_{i=1}^n A_i \right) \leq \sum_{i=1}^n \Pr (A_i)$$

Simple inductive proof, using the inclusion-exclusion principle for two sets.

The events (sets) A_i do not need to satisfy any special property.



Disjoint and independent events

Disjoint or mutually exclusive events: events A and B are disjoint or mutually exclusive if their intersection is empty.

Independent events: events A and B are independent if the occurrence of one does not affect the probability of the other.

Formally, two events A and B are independent if

$$\Pr(A \cap B) = \Pr(A)\Pr(B)$$



Random variables

Random variables: functions mapping the sample space Ω to real numbers.

(They are called *variables*, even though they really are *functions*.)

Notice that given any real number a , the inverse

$$X^{-1}(a) = \{\omega \in \Omega \mid X(\omega) = a\}$$

is a subset of Ω **and** thus is an event **and** has a probability.



Random variables

Random variables: functions mapping the sample space Ω to real numbers.

(They are called *variables*, even though they really are *functions*.)

Notice that given any real number a , the inverse

$$X^{-1}(a) = \{\omega \in \Omega \mid X(\omega) = a\}$$

is a subset of Ω **and** thus is an event **and** has a probability.

Abusing notation: we use $\Pr(X = a)$ instead of $\Pr(X^{-1}(a))$.

This function of a is of great interest. We can also define $\Pr(X \leq a)$ as follows:

$$\Pr(X \leq a) = \Pr(X^{-1}((-\infty, a])) = \Pr(\{\omega \in \Omega \mid X(\omega) \leq a\})$$



Example

Example: let Ω be the sample space of two dice rolls with 36 elements and assume that both dice are fair.

The sum of the two numbers on the two dice (denoted by - say - S) is a random variable, taking values between 2 and 12.

We can compute the probabilities $\Pr(S=s)$:

| | | | | | | | | | | | |
|--------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| s | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| $\Pr(S = s)$ | $\frac{1}{36}$ | $\frac{2}{36}$ | $\frac{3}{36}$ | $\frac{4}{36}$ | $\frac{5}{36}$ | $\frac{6}{36}$ | $\frac{5}{36}$ | $\frac{4}{36}$ | $\frac{3}{36}$ | $\frac{2}{36}$ | $\frac{1}{36}$ |



PMF and CDF

Probability mass function or pmf:

$$f(a) = \Pr(X = a), \quad a \in \mathbb{R}$$

Cumulative distribution function or cdf:

$$F(a) = \Pr(X \leq a), \quad a \in \mathbb{R}$$

Clearly, $F(a) = \sum_{x \leq a} f(x)$.

If the sample space is countably infinite some more care required; a lot more care is required if the sample space is uncountably infinite.



Independent random variables

Two random variables X and Y are independent if (for all reals a, b)

$$\underbrace{\Pr(X = a \text{ and } Y = b)} = \Pr(X = a)\Pr(Y = b)$$

Joint mass function

$$f_{X,Y}(a,b) = \Pr(\{\omega \in \Omega \mid (X(\omega) = a) \wedge (Y(\omega) = b)\})$$



Expectation

Given a random variable X , its expectation $E(X)$ is defined as

$$E(X) = \sum_{x \in X(\Omega)} x \cdot \Pr(X = x)$$

$X(\Omega)$ is the image of X (recall that X is a function). For finite Ω

$$E(X) = \sum_{\omega \in \Omega} X(\omega) \Pr(\omega).$$

For countably infinite sample spaces care needs to be taken regarding the convergence of the appropriate sum.

The Lebesgue-Stieljes integral can be used for uncountable sample spaces.



Properties of expectation

Linearity of expectation: for any random variables X and Y and real λ

$$E(X + Y) = E(X) + E(Y)$$

$$E(\lambda X) = \lambda E(X).$$

The first property generalizes to any (finite) sum of random variables and holds even if the random variables are not independent.

If two random variables X and Y are independent then

$$E(XY) = E(X)E(Y).$$



Variance

The variance of a random variable X is defined as

$$\text{Var}(X) = E(X - E(X))^2.$$

In words, it measures the average of the (square) of the difference $X - E(X)$.

The standard deviation is the square root of the variance.

Easy to prove:

$$\text{Var}(X) = E(X^2) - (E(X))^2 \leq E(X^2).$$



Properties of variance

If X and Y are independent random variables, then

$$\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y).$$

For any real λ

$$\text{Var}(\lambda X) = \lambda^2 \text{Var}(X).$$



Markov's inequality

Let X be a **non-negative** random variable; for any $a > 0$

$$\Pr(X \geq a) \leq \frac{E(X)}{a}$$

Really simple to apply: only needs a bound for the expectation of X .



Markov's inequality: equivalent formulas

Let X be a **non-negative** random variable; for any $k > 1$

$$\Pr(X \geq k \cdot E(X)) \leq \frac{1}{k}$$

or, equivalently,

$$\Pr(X < k \cdot E(X)) \geq 1 - \frac{1}{k}$$



Markov's inequality: proof hint

Let X be a **non-negative** random variable; for any $a > 0$

$$\Pr(X \geq a) \leq \frac{E(X)}{a}$$

(*Proof hint:* define the random variable I_a that takes the value 1 if $X \geq a$ and 0 otherwise; come up with an obvious upper bound for I_a and then bound its expectation.)



Chebyshev's inequality

Let X be a random variable with expectation $E(X)$ (finite) and variance $\text{Var}(X) = \sigma^2$ (also finite:); Then, for any $a > 0$,

$$\Pr(|X - E(X)| \geq a\sigma) \leq \frac{1}{a^2}$$

We now need bounds for the expectation and the variance of X .

Proof hint: Apply Markov's inequality on $(X - E(X))^2$.



Bernstein-type inequalities

We now focus on inequalities that **measure how much a sum of random variables deviates from its expectation.**

Given **independent** random variables X_i for $i=1\dots n$ we seek bounds for:

$$\left| \sum_{i=1}^n X_i - \sum_{i=1}^n \mathbb{E}(X_i) \right|$$

These bounds should hold with “high probability”.



Bernstein-type inequalities

We now focus on inequalities that **measure how much a sum of random variables deviates from its expectation.**

Given **independent** random variables X_i for $i=1\dots n$ we seek bounds for:

$$\left| \sum_{i=1}^n X_i - \sum_{i=1}^n E(X_i) \right|$$

These bounds should hold with “high probability”.

Remark 1: We already know from the Central Limit Theorem that a sum of n independent, identically distributed random variables with bounded, common, mean and variance converges to a normal distribution as n approaches infinity. The inequalities that we will discuss deal with smaller, finite values of n .

Remark 2: Early versions of such inequalities are often attributed to Bernstein in the 1920's, but have been rediscovered in various forms several times (starting in the 1950s with Chernoff bounds, Hoeffding's inequality, Azuma's inequality, etc.).



Bernstein-type inequalities

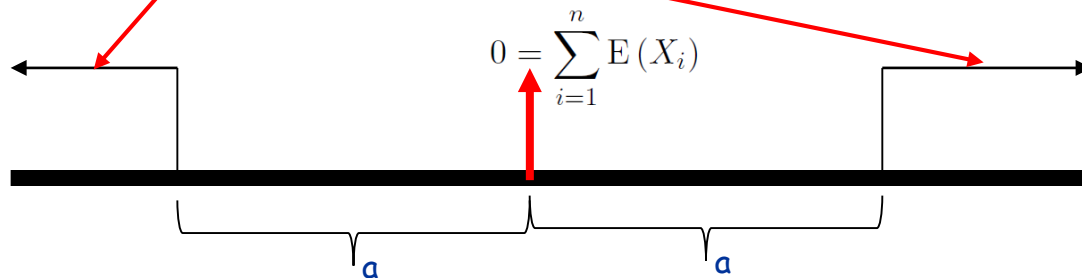
Given **independent** random variables X_i such that $E(X_i) = 0$ and $-M \leq X_i \leq M$ for all $i=1\dots n$,

$$\Pr \left(\left| \sum_{i=1}^n X_i \right| > a \right) \leq 2 \exp \left(- \frac{a^2/2}{\sum_{i=1}^n E(X_j^2) + Ma/3} \right)$$

Bernstein-type inequalities

Given **independent** random variables X_i such that $E(X_i) = 0$ and $-M \leq X_i \leq M$ for all $i=1\dots n$,

$$\Pr \left(\left| \sum_{i=1}^n X_i \right| > a \right) \leq 2 \exp \left(- \frac{a^2/2}{\sum_{i=1}^n E(X_i^2) + Ma/3} \right)$$





Bernstein-type inequalities

Given **independent** random variables X_i such that $E(X_i) = 0$ and $-M \leq X_i \leq M$ for all $i=1\dots n$,

$$\Pr \left(\left| \sum_{i=1}^n X_i \right| > a \right) \leq 2 \exp \left(- \frac{a^2/2}{\sum_{i=1}^n E(X_j^2) + Ma/3} \right)$$

The above inequality holds for any $a > 0$.

- The expectation of the sum of the X_i 's is zero.
- In words, the above inequality states that the probability that the sum of the X_i 's deviates from its expectation by more than a is bounded.
- Many variants can be derived.

Approximating Matrix Multiplication

(Drineas, Kannan, Mahoney SICOMP 2006)

Problem Statement

Given an m -by- n matrix A and an n -by- p matrix B , approximate the product $A \cdot B$,

OR, equivalently,

Approximate the sum of n rank-one matrices.

$$A \cdot B = \sum_{i=1}^n \underbrace{\begin{pmatrix} A_{*i} \\ \end{pmatrix} \cdot \begin{pmatrix} B_{i*} \\ \end{pmatrix}}_{\in \mathbb{R}^{m \times p}}$$

i -th column of A

i -th row of B

Each term in the summation is a rank-one matrix

A sampling approach

(DKM SICOMP 2006)

$$A \cdot B = \sum_{i=1}^n \underbrace{\begin{pmatrix} A_{*i} \end{pmatrix}}_{\substack{\text{i-th column of } A \\ \in \mathbb{R}^{m \times p}}} \cdot \begin{pmatrix} B_{i*} \end{pmatrix}_{\substack{\text{i-th row of } B}}$$

Algorithm

1. Fix a set of probabilities $p_i, i = 1 \dots n$, summing up to one.
2. For $t = 1 \dots c$,
set $j_t = i$, where $\Pr(j_t = i) = p_i$.
(Pick c terms of the sum, with replacement, with respect to the p_i .)
3. Approximate the product AB by summing the c terms, after scaling.

Sampling (cont'd)

(DKM SICOMP 2006)

$$A \cdot B = \sum_{i=1}^n \underbrace{\begin{pmatrix} A_{*i} \end{pmatrix}}_{\substack{\text{i-th column of } A \\ \in \mathbb{R}^{m \times p}}} \cdot \begin{pmatrix} B_{i*} \end{pmatrix}$$

B_{i*} ← i-th row of B

$$\approx \frac{1}{c} \sum_{t=1}^c \frac{1}{p_{j_t}} \underbrace{\begin{pmatrix} A_{*j_t} \end{pmatrix}}_{\in \mathbb{R}^{m \times p}} \cdot \begin{pmatrix} B_{j_t*} \end{pmatrix}$$

Keeping the terms j_1, j_2, \dots, j_c .



The algorithm (matrix notation)

$$\begin{pmatrix} A \\ m \times n \end{pmatrix} \cdot \begin{pmatrix} B \\ n \times p \end{pmatrix} \approx \begin{pmatrix} C \\ m \times c \end{pmatrix} \cdot \begin{pmatrix} R \\ c \times p \\ m \times c \end{pmatrix}$$

Algorithm

1. Pick c columns of A to form an m -by- c matrix C and the corresponding c rows of B to form a c -by- p matrix R .
2. Approximate $A \cdot B$ by $C \cdot R$.

Notes

3. We pick the columns and rows with non-uniform probabilities.
4. We scale the columns (rows) prior to including them in C (R).

The algorithm (matrix notation, cont'd)

$$\begin{pmatrix} A \\ m \times n \end{pmatrix} \cdot \begin{pmatrix} B \\ n \times p \end{pmatrix} \approx \begin{pmatrix} C \\ m \times c \end{pmatrix} \cdot \begin{pmatrix} R \\ c \times p \end{pmatrix}$$

- Create C and R by performing c i.i.d. trials, with replacement.
- For $t = 1 \dots c$, pick a column $A_{(j_t)}$ and a row $B_{(j_t)}$ with probability

$$\Pr(j_t = i) = \frac{\|A_{*i}\|_2 \|B_{i*}\|_2}{\sum_{j=1}^n \|A_{*j}\|_2 \|B_{j*}\|_2}$$

- Include $A_{*j_t} / (cp_{j_t})^{1/2}$ as a column of C , and $B_{j_t*} / (cp_{j_t})^{1/2}$ as a row of R .



The algorithm (matrix notation, cont'd)

We can also use the sampling matrix notation:

Let S be an n -by- c matrix whose t -th column (for $t = 1 \dots c$) has a single non-zero entry, namely

$$S_{jtt} = \frac{1}{\sqrt{cp_{jt}}}$$

Clearly:

$$A \cdot B \approx C \cdot R = (AS) \cdot (S^T B)$$



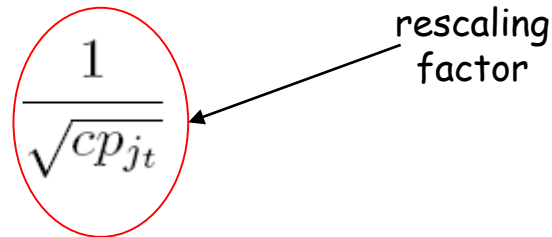
The algorithm (matrix notation, cont'd)

We can also use the sampling matrix notation:

Let S be an n -by- c matrix whose t -th column (for $t = 1 \dots c$) has a single non-zero entry, namely

$$S_{jtt} = \frac{1}{\sqrt{cp_{jt}}}$$

rescaling
factor



Clearly:

$$A \cdot B \approx C \cdot R = (AS) \cdot (S^T B)$$

Remark: S is sparse (has exactly c non-zero elements, one per column).



Simple Lemmas

- It is easy to implement this particular sampling in two passes, when the input matrices are given as a sequence of triplets (i, j, A_{ij}) .
- The expectation of CR (element-wise) is AB (unbiased estimator), regardless of the sampling probabilities, i.e.,

$$\mathbf{E}[(CR)_{ij}] = (AB)_{ij}$$

- We can also bound the variance of the (i, j) -th entry of CR :

$$\mathbf{Var}[(CR)_{ij}] = \frac{1}{c} \sum_{k=1}^n \frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{c} (AB)_{ij}^2$$



Simple Lemmas

$$\mathbf{E} [(CR)_{ij}] = (AB)_{ij}$$

$$\mathbf{Var} [(CR)_{ij}] = \frac{1}{c} \sum_{k=1}^n \frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{c} (AB)_{ij}^2$$

The above two bounds can now be used to bound

$$\mathbf{E} \left[\|AB - CR\|_F^2 \right]$$

Our particular choice of sampling probabilities **minimizes the above expectation.**



A bound for the Frobenius norm

For the above algorithm,

$$\mathbf{E} \|AB - CR\|_F = \mathbf{E} \|AB - ASS^T B\|_F \leq \frac{1}{\sqrt{c}} \|A\|_F \|B\|_F$$

- We can now use Markov's inequality to directly get a “constant probability” bound.
- “High probability” follows from a martingale argument (we will skip this discussion).
- The above bound immediately implies an upper bound for the spectral norm of the error $AB - CR$ (but better bounds can be derived).



Special case: $B = A^T$

If $B = A^T$, then the sampling probabilities are

$$\Pr(j_t = i) = \frac{\|A_{*i}\|_2^2}{\sum_{j=1}^n \|A_{*j}\|_2^2} = \frac{\|A_{*i}\|_2^2}{\|A\|_F^2}$$

Also, $R = C^T$, and the error bounds are:

$$\mathbf{E} \|AA^T - CC^T\|_F = \mathbf{E} \|AA^T - ASS^T A^T\|_F \leq \frac{1}{\sqrt{c}} \|A\|_F^2$$



Special case: $B = A^T$ (cont'd)

(Drineas et al. Num Math 2011, Theorem 4)

A better **spectral norm** bound via matrix Chernoff/Bernstein inequalities:

Assumptions:

- Spectral norm of A is one (not important, just normalization)
- Frobenius norm of A is at least 0.2 (not important, simplifies bounds).

- **Important:** Set

$$c = \Omega \left(\frac{\|A\|_F^2}{\epsilon^2} \ln \left(\frac{\|A\|_F^2}{\epsilon^2 \sqrt{\delta}} \right) \right)$$

Then: for any $0 < \epsilon < 1$, with probability at least $1 - \delta$,

$$\|AA^T - CC^T\|_2 = \|AA^T - ASS^T A^T\|_2 \leq \epsilon$$



An inequality by Oliveira

(Oliveira 2010, Lemma 1)

Let y^1, y^2, \dots, y^c be independent identically distributed copies of the m -dimensional random vector y with

$$\|y\|_2 \leq M \quad \text{and} \quad \|\mathbf{E}(yy^T)\|_2 \leq 1$$

Then, for any $a > 0$,

$$\left\| \frac{1}{c} \sum_{i=1}^c y^i y^{iT} - \mathbf{E}(yy^T) \right\|_2 \leq a$$

holds with probability at least

$$1 - (2c)^2 \exp\left(-\frac{ca^2}{16M^2 + 8M^2a}\right)$$



Proof outline

(Drineas et al. Num Math 2011, Theorem 4)

Define the random vector \mathbf{y} such that

$$\Pr \left(\mathbf{y} = \frac{1}{\sqrt{p_i}} A_{*i} \right) = p_i$$

Now, it is easy to see that the matrix \mathbf{C} has columns

$$\frac{1}{\sqrt{c}} \mathbf{y}^1, \frac{1}{\sqrt{c}} \mathbf{y}^2, \dots, \frac{1}{\sqrt{c}} \mathbf{y}^c$$

where $\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^c$ are independent copies of \mathbf{y} . It is easy to prove that

$$\mathbf{E} [\mathbf{y}\mathbf{y}^T] = \mathbf{A}\mathbf{A}^T \quad \text{and} \quad \mathbf{C}\mathbf{C}^T = \mathbf{A}\mathbf{S}\mathbf{S}^T\mathbf{A}^T = \frac{1}{c} \sum_{t=1}^c \mathbf{y}^t \mathbf{y}^{tT}.$$



Proof outline

(Drineas et al. Num Math 2011, Theorem 4)

We can now upper bound the norm of the vector \mathbf{y} :

$$M = \|\mathbf{y}\|_2 = \frac{1}{\sqrt{p_i}} \|A_{*i}\|_2$$

Also, given our assumption that the spectral norm of \mathbf{A} is at most one, we get:

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

We can now apply a matrix-Bernstein inequality (Lemma 1 of Oliveira 2010) to get the desired bound.



Using a dense S (instead of a sampling matrix...)

We approximated the product AA^T as follows:

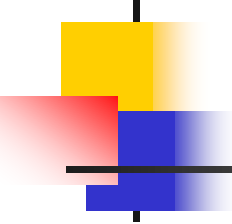
$$A \cdot A^T \approx C \cdot C^T = (AS) \cdot (S^T A^T)$$

Recall that S is an n -by- c sparse matrix (one non-zero entry per column).

Let's replace S by a dense matrix, the random sign matrix:

$$S_{ij} = \begin{cases} +1/\sqrt{c} & ,\text{w.p. } 1/2 \\ -1/\sqrt{c} & ,\text{w.p. } 1/2 \end{cases}$$

Note that S is oblivious to the input matrix A !



Using a dense S (instead of a sampling matrix...) (and assuming A is normalized)

Approximate the product AA^T (assuming that the spectral norm of A is one):

$$A \cdot A^T \approx C \cdot C^T = (AS) \cdot (S^T A^T)$$

Let S by a dense matrix, the random sign matrix:

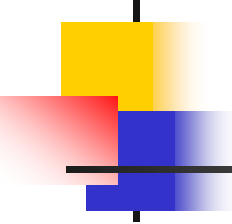
$$S_{ij} = \begin{cases} +1/\sqrt{c} & , \text{w.p. } 1/2 \\ -1/\sqrt{c} & , \text{w.p. } 1/2 \end{cases}$$

If

$$c = \Omega \left(\frac{\|A\|_F^2}{\epsilon^2} \ln m \right)$$

then, with high probability:

$$\|AA^T - CC^T\|_2 = \|AA^T - ASS^T A^T\|_2 \leq \epsilon$$



Using a dense S (instead of a sampling matrix...) (and assuming A is normalized)

Approximate the product AA^T (assuming that the spectral norm of A is one):

$$A \cdot A^T \approx C \cdot C^T = (AS) \cdot (S^T A^T)$$

Let S be a dense matrix, the random sign matrix:

$$S_{ij} = \begin{cases} +1/\sqrt{c} & , \text{w.p. } 1/2 \\ -1/\sqrt{c} & , \text{w.p. } 1/2 \end{cases}$$

If

$$c = \Omega \left(\frac{\|A\|_F^2}{\epsilon^2} \ln m \right)$$

Similar structure with the
sparse S case; some
differences in the \ln factor

then, with high probability:

$$\|AA^T - CC^T\|_2 = \|AA^T - ASS^T A^T\|_2 \leq \epsilon$$



Using a dense S (cont'd)

Other choices for dense matrices S ?

Why bother with a sign matrix?

(Computing the product AS is somewhat slow, taking $O(mnc)$ time.)

Similar bounds are known for better, i.e., computationally more efficient, choices of “random projection” matrices S , most notably:

- When S is the so-called subsampled Hadamard Transform Matrix.

(much faster; avoids full matrix-matrix multiplication; see Sarlos FOCS 2006 and Drineas et al. (2011) Num Math)

- When S is the input sparsity time projection matrix of Clarkson & Woodruff STOC 2013.

(the matrix multiplication result appears in Mahoney & Meng STOC 2013 and was improved by Nelson and Huy FOCS 2013).



Recap: approximating matrix multiplication

We approximated the product AB as follows:

$$A \cdot B \approx C \cdot R = (AS) \cdot (S^T B)$$

Let S be a sampling matrix (actual columns from A and rows from B are selected):

We need to carefully sample columns of A (rows of B) with probabilities that depend on their norms in order to get "good" bounds of the following form:

$$\mathbb{E} [\|AB - CR\|_F] = \mathbb{E} [\|AB - ASS^T B\|_F] \leq \frac{1}{\sqrt{c}} \|A\|_F \|B\|_F$$

$$\|AA^T - CC^T\|_2 = \|AA^T - ASS^T A^T\|_2 \leq \varepsilon$$

Holds with probability at least $1-\delta$ by setting $c = \Omega\left(\frac{\|A\|_F^2}{\varepsilon^2} \ln\left(\frac{\|A\|_F^2}{\varepsilon^2 \sqrt{\delta}}\right)\right)$

Randomized
Gram Matrix Multiplication
and
Least Squares

Outline

- 1 How to sample
- 2 Randomized matrix multiplication $\mathbf{A}^T \mathbf{A}$, again but now with **uniform** probabilities
- 3 **Singular values** of sampled **orthonormal** matrices
- 4 Randomized least squares

Assume: **Exact** arithmetic

How to Sample, Non-Uniformly

Given: Probabilities $0 \leq p_1 \leq \dots \leq p_m$ with $\sum_{i=1}^m p_i = 1$

Want: Sample index $t = j$ from $\{1, \dots, m\}$ with probability p_j

Inversion by sequential search [Devroye 1986]

- 1 Determine partial sums

$$S_k \equiv \sum_{i=1}^k p_i \quad 1 \leq k \leq m$$

- 2 Pick uniform $[0, 1]$ random variable U
- 3 Determine integer j with $S_{j-1} < U \leq S_j$
- 4 Sampled index: $t = j$ with probability $p_j = S_j - S_{j-1}$

How to Sample, Uniformly

Want: Sample c indices from $\{1, \dots, m\}$
independently and identically, each with probability $1/m$

Matlab

```
randi(m, [c, 1])
```

$c \times 1$ vector of **pseudo-random integers** between 1 and m

Randomized Gram Matrix Multiplication

DKM Algorithm

[Drineas, Kannan & Mahoney 2006]

DKM Algorithm (special case)

- Given: $\mathbf{A} = (\mathbf{a}_1 \ \cdots \ \mathbf{a}_m)^T \in \mathbb{R}^{m \times n}$ with rows $\mathbf{a}_j \in \mathbb{R}^n$
- Gram product = sum of outer products

$$\mathbf{A}^T \mathbf{A} = \sum_{j=1}^m \mathbf{a}_j \mathbf{a}_j^T \in \mathbb{R}^{n \times n}$$

DKM Algorithm (special case)

- Given: $\mathbf{A} = (\mathbf{a}_1 \ \cdots \ \mathbf{a}_m)^T \in \mathbb{R}^{m \times n}$ with rows $\mathbf{a}_j \in \mathbb{R}^n$
- Gram product = sum of outer products

$$\mathbf{A}^T \mathbf{A} = \sum_{j=1}^m \mathbf{a}_j \mathbf{a}_j^T \in \mathbb{R}^{n \times n}$$

- DKM algorithm samples c rows of \mathbf{A} , independently and identically, with uniform probabilities $p_j = 1/m$

$$\mathbf{S} = \sqrt{\frac{m}{c}} (\mathbf{e}_{t_1} \ \cdots \ \mathbf{e}_{t_c})^T \in \mathbb{R}^{c \times m}$$

$\mathbf{e}_{t_1}, \dots, \mathbf{e}_{t_c}$ not necessarily distinct

DKM Algorithm (special case)

- Given: $\mathbf{A} = (\mathbf{a}_1 \ \cdots \ \mathbf{a}_m)^T \in \mathbb{R}^{m \times n}$ with rows $\mathbf{a}_j \in \mathbb{R}^n$
- Gram product = sum of outer products

$$\mathbf{A}^T \mathbf{A} = \sum_{j=1}^m \mathbf{a}_j \mathbf{a}_j^T \in \mathbb{R}^{n \times n}$$

- DKM algorithm samples c rows of \mathbf{A} , independently and identically, with uniform probabilities $p_j = 1/m$

$$\mathbf{S} = \sqrt{\frac{m}{c}} (\mathbf{e}_{t_1} \ \cdots \ \mathbf{e}_{t_c})^T \in \mathbb{R}^{c \times m}$$

$\mathbf{e}_{t_1}, \dots, \mathbf{e}_{t_c}$ not necessarily distinct

- Output = sum of c scaled outer products

$$(\mathbf{S}\mathbf{A})^T (\mathbf{S}\mathbf{A}) = \frac{m}{c} \sum_{j=1}^c \mathbf{a}_{t_j} \mathbf{a}_{t_j}^T$$

Expected Value of Output

- Output from DKM algorithm

$$\mathbf{X} = (\mathbf{S}\mathbf{A})^T (\mathbf{S}\mathbf{A}) = \mathbf{X}_1 + \cdots + \mathbf{X}_c$$

- Sum of matrix-valued random variables $\mathbf{X}_j = \frac{m}{c} \mathbf{a}_{t_j} \mathbf{a}_{t_j}^T$
- Expected value

$$\mathbb{E}[\mathbf{X}_j] = \sum_{i=1}^m \frac{1}{m} \frac{m}{c} \mathbf{a}_i \mathbf{a}_i^T = \frac{1}{c} \mathbf{A}^T \mathbf{A}$$

- Output is unbiased estimator

$$\mathbb{E}[\mathbf{X}] = c \mathbb{E}[\mathbf{X}_j] = \mathbf{A}^T \mathbf{A}$$

Idea for Probabilistic Bounds

- Output from DKM algorithm

$$\mathbf{X} = (\mathbf{SA})^T (\mathbf{SA}) = \mathbf{X}_1 + \cdots + \mathbf{X}_c$$

- Independent matrix-valued random variables

$$\mathbf{X}_j = \frac{m}{c} \mathbf{a}_{t_j} \mathbf{a}_{t_j}^T \quad \text{with} \quad \mathbb{E}[\mathbf{X}_j] = \frac{1}{c} \mathbf{A}^T \mathbf{A}$$

- Change of variable to **zero-mean**

$$\mathbf{Y}_j = \mathbf{X}_j - \frac{1}{c} \mathbf{A}^T \mathbf{A} \quad \text{with} \quad \mathbb{E}[\mathbf{Y}_j] = \mathbf{0}$$

- Sum

$$\begin{aligned} \mathbf{Y} &= \mathbf{Y}_1 + \cdots + \mathbf{Y}_c \quad \text{with} \quad \mathbb{E}[\mathbf{Y}] = \mathbf{0} \\ &= \mathbf{X} - \mathbf{A}^T \mathbf{A} \end{aligned}$$

Deviation of **Y** from mean zero \triangleq Absolute error in **X**

Idea for Probabilistic Bounds, ctd

- Apply **matrix Bernstein concentration inequality** to \mathbf{Y}

$$\Pr [\|\mathbf{Y}\|_2 \geq \hat{\epsilon}] \leq \delta(\hat{\epsilon}, \dots)$$

{Deviation of \mathbf{Y} from mean since $\mathbb{E}[\mathbf{Y}] = \mathbf{0}$ }

- Retrieve **original random variables**

$$\mathbf{Y} = \mathbf{X} - \mathbb{E}[\mathbf{X}] = \mathbf{X} - \mathbf{A}^T \mathbf{A}$$

- Relative error due to randomization for \mathbf{X}

$$\Pr \left[\|\mathbf{X} - \mathbf{A}^T \mathbf{A}\|_2 \leq \hat{\epsilon} \right] \geq 1 - \delta(\hat{\epsilon}, \dots)$$

with $\hat{\epsilon} = \|\mathbf{A}^T \mathbf{A}\|_2 \epsilon$

Matrix Bernstein Concentration Inequality [Tropp 2011]

- 1 Independent random symmetric $\mathbf{Y}_j \in \mathbb{R}^{n \times n}$
- 2 $\mathbb{E}[\mathbf{Y}_j] = 0$ {zero mean}
- 3 $\|\mathbf{Y}_j\|_2 \leq \beta$ {bounded}
- 4 $\left\| \sum_j \mathbb{E}[\mathbf{Y}_j^2] \right\|_2 \leq \nu$ {"variance"}

Failure probability: For any $\hat{\epsilon} > 0$

$$\Pr \left[\left\| \sum_j \mathbf{Y}_j \right\|_2 \geq \hat{\epsilon} \right] \leq n \exp \left(- \frac{\hat{\epsilon}^2/2}{\nu + \beta \hat{\epsilon}/3} \right)$$

{Deviation from mean}

Check Assumptions for Concentration Inequality

- 1 Independent symmetric $n \times n$ random variables

$$\mathbf{Y}_j = \mathbf{X}_j - \frac{1}{c} \mathbf{A}^T \mathbf{A} = \frac{1}{c} \left(m \mathbf{a}_{t_j} \mathbf{a}_{t_j}^T - \mathbf{A}^T \mathbf{A} \right)$$

- 2 Zero mean $\mathbb{E}[\mathbf{Y}_j] = \mathbf{0}$

- 3 Bounded

$$\|\mathbf{Y}_j\|_2 \leq \frac{1}{c} \max_{1 \leq i \leq m} \{m \|\mathbf{a}_i\|_2^2, \|\mathbf{A}\|_2^2\} = \underbrace{\frac{m}{c} \max_{1 \leq i \leq m} \|\mathbf{a}_i\|_2^2}_{\beta}$$

- 4 Variance

$$\left\| \sum_{j=1}^c \mathbb{E}[\mathbf{Y}_j^2] \right\|_2 \leq \underbrace{\beta}_{\nu} \|\mathbf{A}\|_2^2$$

Apply Concentration Inequality

- Failure probability

$$\delta = n \exp\left(-\frac{\hat{\epsilon}^2/2}{\nu + \beta \hat{\epsilon}/3}\right)$$

- Insert $\nu = \beta \|\mathbf{A}\|_2^2$ and $\hat{\epsilon} = \epsilon \|\mathbf{A}\|_2^2$

$$\delta = n \exp\left(-\frac{\epsilon^2 \|\mathbf{A}\|_2^2}{2\beta(1 + \epsilon/3)}\right)$$

- Row mass $\mu = \max_{1 \leq i \leq m} \|\mathbf{a}_i\|_2^2 / \|\mathbf{A}\|_2^2$

- Insert $\beta = \frac{m}{c} \|\mathbf{A}\|_2^2 \mu$

$$\delta = n \exp\left(-\frac{c \epsilon^2}{2 m \mu (1 + \epsilon/3)}\right)$$

Relative Error due to Randomization [Holodnak & Ipsen 2015]

- Success probability

$$\Pr \left[\left\| (\mathbf{SA})^T (\mathbf{SA}) - \mathbf{A}^T \mathbf{A} \right\|_2 \leq \epsilon \left\| \mathbf{A}^T \mathbf{A} \right\|_2 \right] \geq 1 - \delta$$

where

$$\delta = n \exp \left(- \frac{c \epsilon^2}{2 m \mu (1 + \epsilon/3)} \right)$$

- Row mass $\mu = \max_{1 \leq i \leq m} \|\mathbf{a}_i\|_2^2 / \|\mathbf{A}\|_2^2$

Relative Error due to Randomization [Holodnak & Ipsen 2015]

- Success probability

$$\Pr \left[\left\| (\mathbf{SA})^T (\mathbf{SA}) - \mathbf{A}^T \mathbf{A} \right\|_2 \leq \epsilon \left\| \mathbf{A}^T \mathbf{A} \right\|_2 \right] \geq 1 - \delta$$

where

$$\delta = n \exp \left(- \frac{c \epsilon^2}{2 m \mu (1 + \epsilon/3)} \right)$$

- Row mass $\mu = \max_{1 \leq i \leq m} \|\mathbf{a}_i\|_2^2 / \|\mathbf{A}\|_2^2$
- Sampling amount: If

$$c \geq \frac{8}{3} m \mu \frac{\ln(n/\delta)}{\epsilon^2}$$

then with probability $\geq 1 - \delta$

$$\left\| (\mathbf{SA})^T (\mathbf{SA}) - \mathbf{A}^T \mathbf{A} \right\|_2 \leq \epsilon \left\| \mathbf{A}^T \mathbf{A} \right\|_2$$

Conclusion for DKM Algorithm

- $\mathbf{A} = (\mathbf{a}_1 \ \cdots \ \mathbf{a}_m)^T \in \mathbb{R}^{m \times n}$ $\mu = \max_{1 \leq i \leq m} \frac{\|\mathbf{a}_i\|_2^2}{\|\mathbf{A}\|_2^2}$
- Failure probability $0 < \delta \leq 1$
- Relative error bound $0 < \epsilon < 1$
- Uniform probabilities $p_j = 1/m, 1 \leq j \leq m$
- Sampling amount

$$c \geq \frac{8}{3} m \mu \frac{\ln(n/\delta)}{\epsilon^2}$$

With probability $\geq 1 - \delta$

DKM algorithm **samples c rows \mathbf{SA}** with

$$\|(\mathbf{SA})^T (\mathbf{SA}) - \mathbf{A}^T \mathbf{A}\|_2 \leq \epsilon \|\mathbf{A}^T \mathbf{A}\|_2$$

Matrices with Orthonormal Columns

- $\mathbf{Q} = (\mathbf{q}_1 \ \cdots \ \mathbf{q}_m)^T \in \mathbb{R}^{m \times n}$ with $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}_n$
- Coherence

$$\mu = \max_{1 \leq i \leq m} \|\mathbf{q}_i\|_2^2 \quad \frac{n}{m} \leq \mu \leq 1$$

Coherence [Donoho & Huo 2001]

$$\mathbf{Q} = (\mathbf{q}_1 \ \cdots \ \mathbf{q}_m)^T \in \mathbb{R}^{m \times n} \text{ with } \mathbf{Q}^T \mathbf{Q} = \mathbf{I}_n$$

$$\text{Coherence } \mu = \max_{1 \leq i \leq m} \|\mathbf{q}_i\|_2^2$$

- $\frac{n}{m} \leq \mu \leq 1$
- **Maximal** coherence: $\mu = 1$
At least one row \mathbf{q}_j^T is a **canonical vector**
- **Minimal** coherence: $\mu = \frac{n}{m}$
Columns of \mathbf{Q} are columns of a $m \times m$ **Hadamard matrix**
- **Property of the column space** $\mathcal{S} = \text{range}(\mathbf{Q})$
Independent of particular orthonormal basis for \mathcal{S}
- **Coherence measures**
 - Correlation with canonical basis
 - Difficulty of **recovering** matrix from sampling

Matrices with Orthonormal Columns

- $\mathbf{Q} = (\mathbf{q}_1 \ \cdots \ \mathbf{q}_m)^T \in \mathbb{R}^{m \times n}$ with $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}_n$
- Coherence

$$\mu = \max_{1 \leq i \leq m} \|\mathbf{q}_i\|_2^2 \quad \frac{n}{m} \leq \mu \leq 1$$

Matrices with Orthonormal Columns

- $\mathbf{Q} = (\mathbf{q}_1 \ \cdots \ \mathbf{q}_m)^T \in \mathbb{R}^{m \times n}$ with $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}_n$

- Coherence

$$\mu = \max_{1 \leq i \leq m} \|\mathbf{q}_i\|_2^2 \quad \frac{n}{m} \leq \mu \leq 1$$

- Singular values of sampled matrix

$$\sigma_1(\mathbf{S}\mathbf{Q}) \geq \cdots \geq \sigma_n(\mathbf{S}\mathbf{Q})$$

- Well conditioning of singular values

$$\|(\mathbf{S}\mathbf{Q})^T (\mathbf{S}\mathbf{Q}) - \mathbf{I}_n\|_2 \leq \epsilon \quad \iff$$

$$\sqrt{1 - \epsilon} \leq \sigma_j(\mathbf{S}\mathbf{Q}) \leq \sqrt{1 + \epsilon} \quad 1 \leq j \leq n$$

Singular Values of Sampled Orthonormal Matrices

[Holodnak & Ipsen 2015]

- Failure probability $0 < \delta \leq 1$
- Relative error bound $0 < \epsilon < 1$
- Sampling amount $c \geq \frac{8}{3} m \mu \frac{\ln(n/\delta)}{\epsilon^2}$

With probability $\geq 1 - \delta$

- Simultaneous bound for **all** singular values

$$\sqrt{1 - \epsilon} \leq \sigma_j(\mathbf{SQ}) \leq \sqrt{1 + \epsilon} \quad 1 \leq j \leq n$$

Singular Values of Sampled Orthonormal Matrices

[Holodnak & Ipsen 2015]

- Failure probability $0 < \delta \leq 1$
- Relative error bound $0 < \epsilon < 1$
- Sampling amount $c \geq \frac{8}{3} m \mu \frac{\ln(n/\delta)}{\epsilon^2}$

With probability $\geq 1 - \delta$

- Simultaneous bound for **all** singular values

$$\sqrt{1 - \epsilon} \leq \sigma_j(\mathbf{S}\mathbf{Q}) \leq \sqrt{1 + \epsilon} \quad 1 \leq j \leq n$$

- Extreme singular values

$$\begin{aligned} 0 < \sqrt{1 - \epsilon} &\leq \sigma_n(\mathbf{S}\mathbf{Q}) = 1 / \|(\mathbf{S}\mathbf{Q})^\dagger\|_2 \\ \|\mathbf{S}\mathbf{Q}\|_2 &= \sigma_1(\mathbf{S}\mathbf{Q}) \leq \sqrt{1 + \epsilon} \end{aligned}$$

Singular Values of Sampled Orthonormal Matrices

[Holodnak & Ipsen 2015]

- Failure probability $0 < \delta \leq 1$
- Relative error bound $0 < \epsilon < 1$
- Sampling amount $c \geq \frac{8}{3} m \mu \frac{\ln(n/\delta)}{\epsilon^2}$

With probability $\geq 1 - \delta$

- Simultaneous bound for **all** singular values

$$\sqrt{1 - \epsilon} \leq \sigma_j(\mathbf{S}\mathbf{Q}) \leq \sqrt{1 + \epsilon} \quad 1 \leq j \leq n$$

- Extreme singular values

$$\begin{aligned} 0 < \sqrt{1 - \epsilon} &\leq \sigma_n(\mathbf{S}\mathbf{Q}) = 1 / \|(\mathbf{S}\mathbf{Q})^\dagger\|_2 \\ \|\mathbf{S}\mathbf{Q}\|_2 &= \sigma_1(\mathbf{S}\mathbf{Q}) \leq \sqrt{1 + \epsilon} \end{aligned}$$

- **Full rank:** $\text{rank}(\mathbf{S}\mathbf{Q}) = n$

Conditioning of Sampled Matrices

- $\mathbf{Q} = (\mathbf{q}_1 \ \cdots \ \mathbf{q}_m)^T \in \mathbb{R}^{m \times n}$ $\mu = \max_{1 \leq i \leq m} \|\mathbf{q}_i\|_2^2$
- Failure probability $0 < \delta \leq 1$
- Relative error bound $0 < \epsilon < 1$
- Sampling amount $c \geq \frac{8}{3} m \mu \frac{\ln(n/\delta)}{\epsilon^2}$
- DKM algorithm samples c rows $\mathbf{S}\mathbf{Q}$ with uniform probabilities $p_j = 1/m$

With probability $\geq 1 - \delta$

$$\text{rank}(\mathbf{S}\mathbf{Q}) = n \quad \text{and} \quad \kappa(\mathbf{S}\mathbf{Q}) = \|\mathbf{S}\mathbf{Q}\|_2 \|(\mathbf{S}\mathbf{Q})^\dagger\|_2 \leq \sqrt{\frac{1+\epsilon}{1-\epsilon}}$$

Strategy for DKM Algorithm

- 1 Choose sampling matrix \mathbf{S} to produce unbiased estimator

$$\mathbb{E}[(\mathbf{SA})^T (\mathbf{SA})] = \mathbf{A}^T \mathbf{A}$$

- 2 Absolute error equals deviation from the mean

$$(\mathbf{SA})^T (\mathbf{SA}) - \mathbf{A}^T \mathbf{A} = (\mathbf{SA})^T (\mathbf{SA}) - \mathbb{E}[(\mathbf{SA})^T (\mathbf{SA})]$$

- 3 Express deviation from mean as
sum of matrix-valued (zero-mean) random variables

$$(\mathbf{SA})^T (\mathbf{SA}) - \mathbb{E}[(\mathbf{SA})^T (\mathbf{SA})] = \mathbf{Y}_1 + \cdots + \mathbf{Y}_c$$

- 4 Apply matrix concentration inequality to $\mathbf{Y}_1 + \cdots + \mathbf{Y}_c$

For relative error use $\hat{\epsilon} = \|\mathbf{A}^T \mathbf{A}\|_2 \epsilon$

Analysis of DKM Algorithm

- Relative error bound

$$\|(\mathbf{SA})^T (\mathbf{SA}) - \mathbf{A}^T \mathbf{A}\|_2 \leq \epsilon \|\mathbf{A}^T \mathbf{A}\|_2$$

- Sampling with uniform probabilities $p_j = 1/m$
- Number of samples $c = \Omega\left(m \mu \frac{\ln n}{\epsilon^2}\right)$ depends on

Large dimension m

Row mass $\mu = \max_j \|\mathbf{a}_j\|_2^2 / \|\mathbf{A}\|_2^2$

- Orthonormal matrices \mathbf{Q} with $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}_n$

$$\|(\mathbf{SQ})^T (\mathbf{SQ}) - \mathbf{I}_n\|_2 \leq \epsilon \iff \sqrt{1-\epsilon} \leq \sigma_j(\mathbf{SQ}) \leq \sqrt{1+\epsilon}$$

- Probability that \mathbf{SQ} has full rank, bounds on condition number

Randomized Least Squares DMMS Algorithm

[Drineas, Mahoney, Muthukrishnan & Sarlós 2011]

DMMS Algorithm (most basic version)

- Given: $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $\text{rank}(\mathbf{A}) = n$ and $\mathbf{b} \in \mathbb{R}^m$

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2$$

- Unique solution $\mathbf{x}_o \equiv \mathbf{A}^\dagger \mathbf{b}$ with residual $\mathbf{r} \equiv \mathbf{Ax}_o - \mathbf{b}$

DMMS Algorithm (most basic version)

- Given: $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $\text{rank}(\mathbf{A}) = n$ and $\mathbf{b} \in \mathbb{R}^m$

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2$$

- Unique solution $\mathbf{x}_o \equiv \mathbf{A}^\dagger \mathbf{b}$ with residual $\mathbf{r} \equiv \mathbf{Ax}_o - \mathbf{b}$
- DMMS algorithm samples c rows of $(\mathbf{A} \ \mathbf{b})$ independently and identically, with uniform probabilities $p_j = 1/m$

$$\mathbf{S} = \sqrt{\frac{m}{c}} (\mathbf{e}_{t_1} \ \cdots \ \mathbf{e}_{t_c})^T \in \mathbb{R}^{c \times m}$$

$\mathbf{e}_{t_1}, \dots, \mathbf{e}_{t_c}$ not necessarily distinct

- Output = minimal norm solution of

$$\min_{\tilde{\mathbf{x}}} \|\mathbf{S}(\mathbf{A}\tilde{\mathbf{x}} - \mathbf{b})\|_2$$

Bounding the Error with regard to Randomization

- Perturbed solution $\tilde{\mathbf{x}}_o = (\mathbf{SA})^\dagger (\mathbf{Sb})$
- Residual $\tilde{\mathbf{r}} = \mathbf{A}\tilde{\mathbf{x}}_o - \mathbf{b}$ with regard to *exact* problem
- Want: Bounds for relative errors

$$\|\tilde{\mathbf{x}}_o - \mathbf{x}_o\|_2 / \|\mathbf{x}_o\|_2 \quad \text{and} \quad \|\tilde{\mathbf{r}} - \mathbf{r}\|_2 / \|\mathbf{r}\|_2$$

Bounding the Error with regard to Randomization

- Perturbed solution $\tilde{\mathbf{x}}_o = (\mathbf{SA})^\dagger (\mathbf{Sb})$
- Residual $\tilde{\mathbf{r}} = \mathbf{A}\tilde{\mathbf{x}}_o - \mathbf{b}$ with regard to **exact** problem
- Want: Bounds for relative errors

$$\|\tilde{\mathbf{x}}_o - \mathbf{x}_o\|_2 / \|\mathbf{x}_o\|_2 \quad \text{and} \quad \|\tilde{\mathbf{r}} - \mathbf{r}\|_2 / \|\mathbf{r}\|_2$$

Two-part analysis

- 1 **Structural part**
Multiplicative perturbation bounds for least squares
{Push deterministic perturbation analysis as far as possible}
- 2 **Probabilistic part**
Apply concentration inequality

Preparing for the Structural Part

- Full and thin QR decompositions

$$\mathbf{A} = (\mathbf{Q}_1 \quad \mathbf{Q}_2) \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} = \mathbf{Q}_1 \mathbf{R}$$

where

$$\mathbf{Q}_1^T \mathbf{Q}_1 = \mathbf{I}_n \quad \mathbf{Q}_2^T \mathbf{Q}_2 = \mathbf{I}_{m-n} \quad \mathbf{Q}_1^T \mathbf{Q}_2 = \mathbf{0}$$

Preparing for the Structural Part

- Full and thin QR decompositions

$$\mathbf{A} = (\mathbf{Q}_1 \quad \mathbf{Q}_2) \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} = \mathbf{Q}_1 \mathbf{R}$$

where

$$\mathbf{Q}_1^T \mathbf{Q}_1 = \mathbf{I}_n \quad \mathbf{Q}_2^T \mathbf{Q}_2 = \mathbf{I}_{m-n} \quad \mathbf{Q}_1^T \mathbf{Q}_2 = \mathbf{0}$$

- Exact solution and residual

$$\mathbf{x}_o = \mathbf{R}^{-1} \mathbf{Q}_1^T \mathbf{b} \quad \mathbf{r} = -\mathbf{Q}_2 \mathbf{Q}_2^T \mathbf{b}$$

Preparing for the Structural Part

- Full and thin QR decompositions

$$\mathbf{A} = (\mathbf{Q}_1 \quad \mathbf{Q}_2) \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} = \mathbf{Q}_1 \mathbf{R}$$

where

$$\mathbf{Q}_1^T \mathbf{Q}_1 = \mathbf{I}_n \quad \mathbf{Q}_2^T \mathbf{Q}_2 = \mathbf{I}_{m-n} \quad \mathbf{Q}_1^T \mathbf{Q}_2 = \mathbf{0}$$

- Exact solution and residual

$$\mathbf{x}_o = \mathbf{R}^{-1} \mathbf{Q}_1^T \mathbf{b} \quad \mathbf{r} = -\mathbf{Q}_2 \mathbf{Q}_2^T \mathbf{b}$$

- Angle θ between \mathbf{b} and $\text{range}(\mathbf{A})$

$$\tan \theta = \frac{\|\mathbf{Q}_2^T \mathbf{b}\|_2}{\|\mathbf{Q}_1^T \mathbf{b}\|_2} = \frac{\|\mathbf{r}\|_2}{\|\mathbf{A}\mathbf{x}_o\|_2}$$

Structural Part: Multiplicative Perturbation Bounds

If $\text{rank}(\mathbf{SA}) = n$ then

$$\begin{aligned}\|\tilde{\mathbf{x}}_o - \mathbf{x}_o\|_2 &\leq \eta \|\mathbf{A}^\dagger\|_2 \|\mathbf{r}\|_2 \\ \|\tilde{\mathbf{r}} - \mathbf{r}\|_2 &\leq \eta \|\mathbf{r}\|_2\end{aligned}$$

where

$$\eta = \left\| (\mathbf{SQ}_1)^\dagger (\mathbf{SQ}_2) \right\|_2$$

If also $\mathbf{A}^T \mathbf{b} \neq 0$ then

$$\frac{\|\tilde{\mathbf{x}}_o - \mathbf{x}_o\|_2}{\|\mathbf{x}_o\|_2} \leq \eta \kappa_2(\mathbf{A}) \tan \theta$$

Proof Idea: Change of Variable [DMMS 2011]

- Shift $\tilde{\mathbf{x}}_o$ to **absolute error** $\mathbf{z}_o = \tilde{\mathbf{x}}_o - \mathbf{x}_o$
- Apply to perturbed problem

$$\min_{\tilde{\mathbf{x}}} \|\mathbf{S}(\mathbf{A}\tilde{\mathbf{x}} - \mathbf{b})\|_2 = \min_{\mathbf{z}} \|\mathbf{S}(\mathbf{A}\mathbf{z} + \mathbf{r})\|_2$$

- Unique solution

$$\mathbf{z}_o = -(\mathbf{S}\mathbf{A})^\dagger (\mathbf{S}\mathbf{r}) = \mathbf{R}^{-1} \underbrace{(\mathbf{S}\mathbf{Q}_1)^\dagger (\mathbf{S}\mathbf{Q}_2)}_{\eta} (\mathbf{Q}_2^T \mathbf{b})$$

- Absolute error bound

$$\|\tilde{\mathbf{x}}_o - \mathbf{x}_o\|_2 = \|\mathbf{z}_o\|_2 \leq \|\mathbf{A}^\dagger\|_2 \underbrace{\|(\mathbf{S}\mathbf{Q}_1)^\dagger (\mathbf{S}\mathbf{Q}_2)\|_2}_{\eta} \|\mathbf{r}\|_2$$

Proof Idea: Change of Variable [DMMS 2011]

- Shift $\tilde{\mathbf{x}}_o$ to **absolute error** $\mathbf{z}_o = \tilde{\mathbf{x}}_o - \mathbf{x}_o$
- Apply to perturbed problem

$$\min_{\tilde{\mathbf{x}}} \|\mathbf{S}(\mathbf{A}\tilde{\mathbf{x}} - \mathbf{b})\|_2 = \min_{\mathbf{z}} \|\mathbf{S}(\mathbf{A}\mathbf{z} + \mathbf{r})\|_2$$

- Unique solution

$$\mathbf{z}_o = -(\mathbf{S}\mathbf{A})^\dagger (\mathbf{S}\mathbf{r}) = \mathbf{R}^{-1} \underbrace{(\mathbf{S}\mathbf{Q}_1)^\dagger (\mathbf{S}\mathbf{Q}_2)}_{\eta} (\mathbf{Q}_2^T \mathbf{b})$$

- Absolute error bound

$$\|\tilde{\mathbf{x}}_o - \mathbf{x}_o\|_2 = \|\mathbf{z}_o\|_2 \leq \|\mathbf{A}^\dagger\|_2 \underbrace{\|(\mathbf{S}\mathbf{Q}_1)^\dagger (\mathbf{S}\mathbf{Q}_2)\|_2}_{\eta} \|\mathbf{r}\|_2$$

- Residual

$$\tilde{\mathbf{r}} - \mathbf{r} = \mathbf{A}\mathbf{z}_o = \mathbf{Q}_1 \underbrace{(\mathbf{S}\mathbf{Q}_1)^\dagger (\mathbf{S}\mathbf{Q}_2)}_{\eta} (\mathbf{Q}_2^T \mathbf{b})$$

Interpretation of η

Effect of randomization confined to

$$\eta = \left\| \tilde{\mathbf{X}}_o \right\|_2 \quad \tilde{\mathbf{X}}_o = (\mathbf{S} \mathbf{Q}_1)^\dagger (\mathbf{S} \mathbf{Q}_2)$$

- $\tilde{\mathbf{X}}_o$ is solution of multiple rhs LS problem

$$\min_{\tilde{\mathbf{X}}} \left\| \mathbf{S} (\mathbf{Q}_1 \tilde{\mathbf{X}} - \mathbf{Q}_2) \right\|_F$$

- Corresponding exact LS problem is

$$\min_{\mathbf{X}} \left\| \mathbf{Q}_1 \mathbf{X} - \mathbf{Q}_2 \right\|_F$$

with solution $\mathbf{X}_o = \mathbf{0}$

η quantifies effect of \mathbf{S} on $\text{range}(\mathbf{A})$ and $\text{null}(\mathbf{A}^T)$

Interpretation of Multiplicative Perturbation Bounds

If $\text{rank}(\mathbf{SA}) = n$ then

$$\frac{\|\tilde{\mathbf{x}}_o - \mathbf{x}_o\|_2}{\|\mathbf{x}_o\|_2} \leq \eta \kappa_2(\mathbf{A}) \tan \theta$$
$$\|\tilde{\mathbf{r}} - \mathbf{r}\|_2 \leq \eta \|\mathbf{r}\|_2$$

- No amplification by $[\kappa_2(\mathbf{A})]^2$
- Perfect conditioning of residual
- Non-asymptotic bounds

LS problems **less sensitive** to **multiplicative** perturbations \mathbf{S} than to general additive perturbations

From Structural to Probabilistic

What to do about $\eta = \|(\mathbf{SQ}_1)^\dagger (\mathbf{SQ}_2)\|_2$?

Options

① $\eta \leq \|(\mathbf{SQ}_1)^\dagger\|_2 \|\mathbf{SQ}_2\|_2$

Have: Singular value bound for \mathbf{SQ}_1 ✓

But: Need to know coherence of \mathbf{Q}_2

From Structural to Probabilistic

What to do about $\eta = \|(\mathbf{SQ}_1)^\dagger (\mathbf{SQ}_2)\|_2$?

Options

① $\eta \leq \|(\mathbf{SQ}_1)^\dagger\|_2 \|\mathbf{SQ}_2\|_2$

Have: Singular value bound for \mathbf{SQ}_1 ✓

But: Need to know coherence of \mathbf{Q}_2

② $\eta \leq \left\| ((\mathbf{SQ}_1)^T (\mathbf{SQ}_1))^{-1} \right\|_2 \|(\mathbf{SQ}_1)^T (\mathbf{SQ}_2)\|_2$

Have: Singular value bound for \mathbf{SQ}_1 ✓

But: Need general matrix multiplication bound

From Structural to Probabilistic

What to do about $\eta = \|(\mathbf{SQ}_1)^\dagger (\mathbf{SQ}_2)\|_2$?

Options

① $\eta \leq \|(\mathbf{SQ}_1)^\dagger\|_2 \|\mathbf{SQ}_2\|_2$

Have: Singular value bound for \mathbf{SQ}_1 ✓

But: Need to know coherence of \mathbf{Q}_2

② $\eta \leq \left\| ((\mathbf{SQ}_1)^T (\mathbf{SQ}_1))^{-1} \right\|_2 \|(\mathbf{SQ}_1)^T (\mathbf{SQ}_2)\|_2$

Have: Singular value bound for \mathbf{SQ}_1 ✓

But: Need general matrix multiplication bound

Here {see also [DMMS 2011]}

$$\eta \leq \|(\mathbf{SQ}_1)^\dagger\|_2 \|\mathbf{SQ}_2\|_2 \quad \text{and assume} \quad \|\mathbf{SQ}_2\|_2 \leq \sqrt{\epsilon}$$

Probabilistic Part: Singular Value Bound for \mathbf{SQ}_1

- $\mathbf{Q}_1 \in \mathbb{R}^{m \times n}$ with $\mathbf{Q}_1^T \mathbf{Q}_1 = \mathbf{I}_n$ and $\mu_1 = \max_{1 \leq j \leq m} \|\mathbf{e}_j^T \mathbf{Q}_1\|_2^2$
- Failure probability $0 < \delta \leq 1$
- Relative tolerance $0 < \epsilon < 1$
- Sampling amount $c \geq \frac{8}{3} m \mu_1 \frac{\ln(n/\delta)}{\epsilon^2}$
- DKM algorithm samples c rows \mathbf{SQ}_1 with uniform probabilities $p_j = 1/m$

With probability $\geq 1 - \delta$

$$\text{rank}(\mathbf{SQ}_1) = n \quad \text{and} \quad \|(\mathbf{SQ}_1)^\dagger\|_2 \leq \frac{1}{\sqrt{1 - \epsilon}}$$

Recall the Ingredients for Least Squares

- $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $\text{rank}(\mathbf{A}) = n$ and $\mathbf{b} \in \mathbb{R}^m$
- $\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2$ has solution $\mathbf{x}_o \equiv \mathbf{A}^\dagger \mathbf{b}$ with $\mathbf{r} \equiv \mathbf{Ax}_o - \mathbf{b}$
- QR decompositions: $\mathbf{A} = (\mathbf{Q}_1 \quad \mathbf{Q}_2) \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} = \mathbf{Q}_1 \mathbf{R}$
- Coherence of \mathbf{Q}_1 : $\mu_1 = \max_{1 \leq i \leq m} \|\mathbf{e}_i^T \mathbf{Q}_1\|_2^2$

DMMS algorithm

- 1 Samples c rows $\mathbf{S}(\mathbf{A} \quad \mathbf{b})$ with uniform probabilities $p_j = 1/m$
- 2 Solves $\min_{\tilde{\mathbf{x}}} \|\mathbf{S}(\mathbf{A}\tilde{\mathbf{x}} - \mathbf{b})\|_2$
Minimal norm solution $\tilde{\mathbf{x}}_o = (\mathbf{SA})^\dagger (\mathbf{Sb})$
Residual $\tilde{\mathbf{r}} = \mathbf{A}\tilde{\mathbf{x}}_o - \mathbf{b}$ with regard to exact problem

Conclusion for DMMS Algorithm

- Failure probability $0 < \delta \leq 1$
- Relative tolerance $0 < \epsilon < 1$
- Assumption $\|\mathbf{S}\mathbf{Q}_2\|_2 \leq \sqrt{\epsilon}$
- Sampling amount $c \geq \frac{8}{3} m \mu_1 \frac{\ln(n/\delta)}{\epsilon^2}$

With probability $\geq 1 - \delta$

DMMS algorithm computes $\text{rank}(\mathbf{S}\mathbf{A}) = n$

$$\frac{\|\tilde{\mathbf{x}}_o - \mathbf{x}_o\|_2}{\|\mathbf{x}_o\|_2} \leq \kappa_2(\mathbf{A}) \tan \theta \sqrt{\frac{\epsilon}{1 - \epsilon}}$$
$$\|\tilde{\mathbf{r}} - \mathbf{r}\|_2 \leq \sqrt{\frac{\epsilon}{1 - \epsilon}} \|\mathbf{r}\|_2$$

Analysis of DMMS Algorithm

- Relative error bounds for $\tilde{\mathbf{x}}_o$ and $\tilde{\mathbf{r}} = \mathbf{A}\tilde{\mathbf{x}}_o - \mathbf{b}$
- Sampling with uniform probabilities $p_j = 1/m$
- Number of samples $c = \Omega\left(m\mu_1 \frac{\ln n}{\epsilon^2}\right)$ depends on
 - Large dimension m
 - Coherence $\mu_1 = \max_i \|\mathbf{e}_i^T \mathbf{Q}_1\|_2^2$
- LS problems less sensitive to randomization than to general, additive perturbations

Important Points

Two-part analysis of error due to randomization

- 1 **Structural**: Push linear algebra as far as possible
- 2 **Probabilistic**: Apply concentration inequality

Requirement of **matrix concentration inequalities**

Output of randomized algorithm

= **sum** of matrix-valued random variables

Rule of thumb for **uniform sampling**:

Number of required samples depends on

Large matrix dimension

Coherence of column space

Implementations of RLA algorithms at small, medium, and large-scale

Michael W. Mahoney

ICSI and Dept of Statistics, UC Berkeley

(For more info, see:
<http://www.stat.berkeley.edu/~mmahoney>
or Google on "Michael Mahoney")

ALA 2015

October 2015

Some recent observations

<http://www.hpcwire.com/2015/10/19/numerical-algorithms-and-libraries-at-exascale/>

Jack Dongarra, in HPC wire. Randomized algorithms for exascale computing:

- "... the other seismic force that's shaking the road to exascale computing ... is the **rise of large-scale data analytics** as fundamental for a wide range of application domains."
- "... **one of the most interesting developments in HPC** math libraries is taking place at the intersection of numerical linear algebra and data analytics, where a **new class of randomized algorithms is emerging** ..."
- "... **powerful tools for solving both least squares and low-rank approximation problems, which are ubiquitous** in large-scale data analytics and scientific computing"
- "... these algorithms are playing a **major role in the processing of the information that has previously lain fallow, or even been discarded**, because meaningful analysis of it was simply infeasible—this is the so called "Dark Data phenomenon."
- "The advent of tools capable of usefully processing such vast amounts of data has **brought new light to these previously darkened regions of the scientific data ecosystem.**"

Some recent observations

<http://www.hpcwire.com/2015/10/19/numerical-algorithms-and-libraries-at-exascale/>

Jack Dongarra, in HPC wire. Randomized algorithms for exascale computing:

- “... randomized algorithms should be expected to play a major role in future convergence of HPC and data analytics”
- “Randomized algorithms are not only fast, they are, as compared to traditional algorithms, easier to code, easier to map to modern computer architectures, and display higher levels of parallelism.”
- “... they often introduce implicit regularization, which makes them more numerically robust.
- “... they may produce results that are easier to interpret and analyze in downstream applications.”
- “While not a silver bullet for large-scale data analysis problems, random sampling and random projection algorithms may dramatically improve our ability to solve a wide range of large matrix problems, especially when they are coupled with domain expertise.”

Outline

- 1 Overview of implementation issues
- 2 Communication issues in LA and RLA
- 3 High-quality implementations in RAM
- 4 A theoretical aside: low-rank methods
- 5 Solving ℓ_2 regression in RAM
- 6 Solving ℓ_2 regression using MPI
- 7 Solving ℓ_2 regression in Spark

Lots of implementations

- Numerical linear algebra and scientific computing:
 - ▶ high-quality implementations, control over conditioning, precision, etc.,
 - ▶ often developed/parameterized for relatively “nice” input
- Machine learning and data analysis:
 - ▶ lower-quality implementations, applicable to much less “nice” input
 - ▶ conditioning, precision, etc. versus regularization or downstream objectives
- Tygert, Rokhlin, Martinsson, Halko, etc.
 - ▶ PCA, LS in (general or specialized) scientific computing applications
- Avron, Maymounkov, Toldeo, Meng, Mahoney, Saunders etc.
 - ▶ Blendenpik, LSRN
- Yang, Meng, Mahoney, Gittens, Avron, Bekas, etc.
 - ▶ L1 and then L2 regression in parallel/distributed environments
 - ▶ Skylark
 - ▶ PCA on \geq terabyte sized problems
- Lots more ...

Low-precision versus high-precision issues

Who cares about **10 digits of precision**?

- **SC/NLA**: all this RLA stuff is just to get a good starting point for iterations.
- **Everyone else**: low-precision is usually enough, otherwise iterate with some method.

Low versus high precision.

- Low precision ($\epsilon = 0.1$): get a sketch (typically subspace-preserving) and **solve the subproblem** with a black box
- High precision ($\epsilon = 10^{-10}$): get a sketch (downsampled more aggressively) and **construct a preconditioner** for the original problem.

Looking forward.

- **Many tradeoffs**: data-aware versus data-oblivious; communication versus computation; one machine versus distributed AEC2 versus supercomputer HPC; coupling with downstream objectives, etc.
- **Lots of theory-practice-gap questions** to be answered

Outline

- 1 Overview of implementation issues
- 2 Communication issues in LA and RLA**
- 3 High-quality implementations in RAM
- 4 A theoretical aside: low-rank methods
- 5 Solving ℓ_2 regression in RAM
- 6 Solving ℓ_2 regression using MPI
- 7 Solving ℓ_2 regression in Spark

Metrics for NLA and RLA implementations

Lots of metrics for implementations:

- Flops (dense versus sparse, etc.)
- Communication (single machine or many machines, HPC vs AEC2, for problem or for algorithm, etc.)
- Numerical stability (forward versus backward stability, etc.)
- Accuracy (objective vs certificate, $\epsilon = 10^{-1}$ vs $\epsilon = 10^{-10}$, etc.)

“Communication” means moving data—either between levels of memory hierarchy or between parallel processors over a network

- Flops is a good metric for small problems that fit in cache.
- Motivation for RLA is “massive data” problems.
- Communication costs orders of magnitude more per operation than a flop.

Communication in NLA

(Ballard, Demmel, Holtz, and Schwartz, 2011.)

- **Performance model for sending one “message”** of n contiguous words is

$$\alpha + n\beta = \text{latency} + n/\text{bandwidth}$$

where a “message” could be a cache line in a shared memory system, an actual message in a distributed memory system or a disk access.

- **Overall performance model** (along critical path of an algorithm) is then

$$\gamma F + \beta W + \alpha M = \gamma * \#\text{Flops} + \beta * \#\text{Words.sent} + \alpha * \#\text{Messages}$$

where $\gamma \ll \beta \ll \alpha$, with gaps growing over time (technological trends).

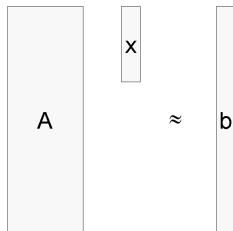
- **Communication-avoiding LA**: identify lower bounds (typically for dense input), and design algorithms that achieve lower bounds.

Communication in RLA

- **So far:** follow same high-level RLA theory as for RAM implementations but worry more about communication
- For very large problems: number of iterations is proxy for communication (Mapreduce versus Spark)
- Important difference: distributed data center (e.g., AEC2) versus parallel shared-memory (e.g., HPC)
- Fun fact: only communication-optimal algorithm for the non symmetric eigenvalue problem is a randomized algorithm
- Caveat: streaming model (developed in TCS for networking applications) is *not* a particularly good idealization for matrix problems

Large-scale environments and how they scale

- Shared memory
 - ▶ cores: $[10, 10^3]^*$
 - ▶ memory: $[100\text{GB}, 100\text{TB}]$
- Message passing
 - ▶ cores: $[200, 10^5]^\dagger$
 - ▶ memory: $[1\text{TB}, 1000\text{TB}]$
 - ▶ CUDA cores: $[5 \times 10^4, 3 \times 10^6]^\ddagger$
 - ▶ GPU memory: $[500\text{GB}, 20\text{TB}]$
- MapReduce
 - ▶ cores: $[40, 10^5]^\S$
 - ▶ memory: $[240\text{GB}, 100\text{TB}]$
 - ▶ storage: $[100\text{TB}, 100\text{PB}]^\P$
- Distributed computing
 - ▶ cores: $[-, 3 \times 10^5]^\parallel$.



* <http://www.sgi.com/pdfs/4358.pdf>

† <http://www.top500.org/list/2011/11/100>

‡ <http://i.top500.org/site/50310>

§ <http://www.cloudera.com/blog/2010/04/pushing-the-limits-of-distributed-processing/>

¶ <http://hortonworks.com/blog/an-introduction-to-hdfs-federation/>

|| <http://fah-web.stanford.edu/cgi-bin/main.py?qtype=osstats>

Outline

- 1 Overview of implementation issues
- 2 Communication issues in LA and RLA
- 3 High-quality implementations in RAM**
- 4 A theoretical aside: low-rank methods
- 5 Solving ℓ_2 regression in RAM
- 6 Solving ℓ_2 regression using MPI
- 7 Solving ℓ_2 regression in Spark

Least-squares and low-rank approximation via RLA

Low-rank approximation

- Tygert, Rokhlin, Martinsson, Halko, etc.
- high-quality numerical implementations
- Gaussian projections for structured matrices in scientific computing applications.

Least-squares approximation

- Tygert, Rokhlin: initial preconditioning iterative algorithm with $O(\log(1/\epsilon))$ non-worst-case complexity
- Avron, Maymounkov, and Toledo: Blendenpik (beats LAPACK clock time)
- Meng, Saunders, and Mahoney: LSRN (for parallel environments)
- Yang, Meng, and Mahoney: terabyte-sized (to low, medium, and high precision on nice and not nice data) in Spark

Many other problems, e.g., ℓ_1 regression, Nystrom/SPSD low-rank approximation, etc.

Outline

- 1 Overview of implementation issues
- 2 Communication issues in LA and RLA
- 3 High-quality implementations in RAM
- 4 A theoretical aside: low-rank methods**
- 5 Solving ℓ_2 regression in RAM
- 6 Solving ℓ_2 regression using MPI
- 7 Solving ℓ_2 regression in Spark

A theoretical aside: low-rank (1 of 2)

(Halko, Martinsson, and Tropp, 2011.)

In scientific computing, goal is to find a good basis for the span of A ...

Input: $m \times n$ matrix A , target rank k and **over-sampling parameter p**

Output: Rank- $(k + p)$ factors U , Σ , and V s.t. $A \approx U\Sigma V^T$.

- 1 Draw a $n \times (k + p)$ **Gaussian random matrix** Ω .
- 2 Form the $n \times (k + p)$ **sample matrix** $Y = A\Omega$.
- 3 Compute an **orthonormal matrix** Q s.t. $Y = QQ^T Y$.
- 4 Form the small matrix $B = Q^T A$.
- 5 Factor the small matrix $B = \hat{U}\Sigma V^T$.
- 6 Form $U = Q\hat{U}$.

Prove bounds of the form

$$\|A - QQ^T A\|_F \leq \left(1 + \frac{k}{p-1}\right)^{1/2} \left(\sum_{j=k+1}^{\min\{m,n\}} \sigma_j^2\right)^{1/2}$$

$$\|A - QQ^T A\|_2 \leq \left(1 + \sqrt{\frac{k}{p-1}}\right) \sigma_{k+1} + \frac{e\sqrt{k+p}}{p} \left(\sum_{j=k+1}^{\min\{m,n\}} \sigma_j^2\right)^{1/2}$$

Question: how does one prove bounds of this form?

A theoretical aside: low-rank (2 of 2)

(Boutsidis, Mahoney, Drineas, 2009.)

Answer: Basic structural result for RLA low-rank matrix approximation.

Lemma

Given $A \in \mathbb{R}^{m \times n}$, let $V_k \in \mathbb{R}^{n \times k}$ be the matrix of the top k right singular vectors of A . Let $\Omega \in \mathbb{R}^{n \times r}$ ($r \geq k$) be any matrix such that $Y^T \Omega$ has full rank. Then, for any unitarily invariant norm ξ ,

$$\|A - P_{A\Omega} A\|_{\xi} \leq \|A - A_k\|_{\xi} + \|\Sigma_{k,\perp} (V_{k,\perp}^T \Omega) (V_k^T \Omega)^{\dagger}\|_{\xi}.$$

Given this structural result, we obtain results for

- the Column Subset Selection Problem (BMD09)
- using random projections to approximate low-rank matrix approximations (RT10,HMT11,etc.)
- developing improved Nyström-based low-rank matrix approximations of SPSP matrices (GM13)
- developing improved feature selection methods (many)
- other low-rank matrix approximation methods

Outline

- 1 Overview of implementation issues
- 2 Communication issues in LA and RLA
- 3 High-quality implementations in RAM
- 4 A theoretical aside: low-rank methods
- 5 Solving ℓ_2 regression in RAM**
- 6 Solving ℓ_2 regression using MPI
- 7 Solving ℓ_2 regression in Spark

Two important notions: leverage and condition

(Mahoney, "Randomized Algorithms for Matrices and Data," FnTML, 2011.)

- **Statistical leverage.** (Think: **eigenvectors**. Important for **low-precision**.)
 - ▶ The **statistical leverage scores** of A (assume $m \gg n$) are the diagonal elements of the projection matrix onto the column span of A .
 - ▶ They equal the ℓ_2 -norm-squared of any orthogonal basis spanning A .
 - ▶ They measure:
 - ★ how well-correlated the singular vectors are with the canonical basis
 - ★ which constraints have largest "influence" on the LS fit
 - ★ a notion of "coherence" or "outlierness"
 - ▶ Computing them exactly is as hard as solving the LS problem.
- **Condition number.** (Think: **eigenvalues**. Important for **high-precision**.)
 - ▶ The **ℓ_2 -norm condition number** of A is $\kappa(A) = \sigma_{\max}(A)/\sigma_{\min}^+(A)$.
 - ▶ $\kappa(A)$ bounds the number of iterations; for ill-conditioned problems (e.g., $\kappa(A) \approx 10^6 \gg 1$), the convergence speed is very slow.
 - ▶ Computing $\kappa(A)$ is generally as hard as solving the LS problem.

These are for the ℓ_2 -norm. Generalizations exist for the ℓ_1 -norm.

Meta-algorithm for ℓ_2 -norm regression (1 of 3)

(Drineas, Mahoney, etc., 2006, 2008, etc., starting with SODA 2006; Mahoney FnTML, 2011.)

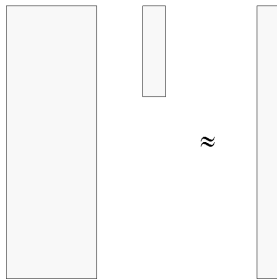
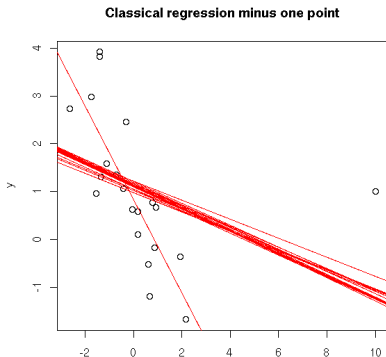
- 1: Using the ℓ_2 statistical leverage scores of A , construct an importance sampling distribution $\{p_i\}_{i=1}^m$.
- 2: Randomly sample a small number of constraints according to $\{p_i\}_{i=1}^m$ to construct a subproblem.
- 3: Solve the ℓ_2 -regression problem on the subproblem.

A naïve version of this meta-algorithm gives a $1 + \epsilon$ relative-error approximation in roughly $O(mn^2/\epsilon)$ time (DMM 2006, 2008). (Ugh.)

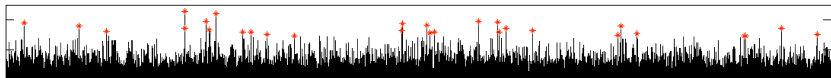
Meta-algorithm for ℓ_2 -norm regression (2 of 3)

(Drineas, Mahoney, etc., 2006, 2008, etc., starting with SODA 2006; Mahoney FnTML, 2011.)

- Randomly sample high-leverage constraints
- Solve the subproblem



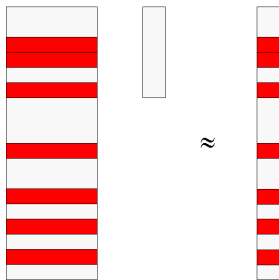
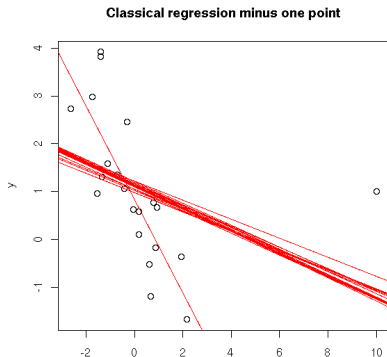
(In many moderately large-scale applications, one uses " ℓ_2 objectives," not since they are "right," but since other things are even more expensive.)



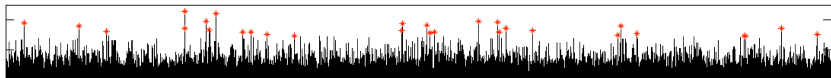
Meta-algorithm for ℓ_2 -norm regression (2 of 3)

(Drineas, Mahoney, etc., 2006, 2008, etc., starting with SODA 2006; Mahoney FnTML, 2011.)

- Randomly sample high-leverage constraints
- Solve the subproblem



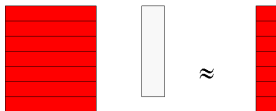
(In many moderately large-scale applications, one uses “ ℓ_2 objectives,” not since they are “right,” but since other things are even more expensive.)



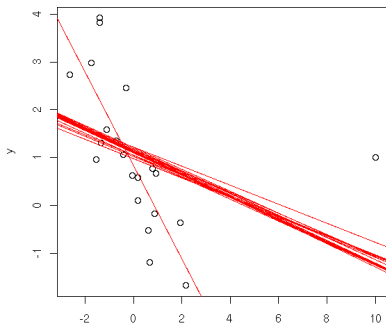
Meta-algorithm for ℓ_2 -norm regression (2 of 3)

(Drineas, Mahoney, etc., 2006, 2008, etc., starting with SODA 2006; Mahoney FnTML, 2011.)

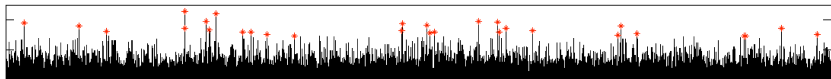
- Randomly sample high-leverage constraints
- Solve the subproblem



Classical regression minus one point



(In many moderately large-scale applications, one uses " ℓ_2 objectives," not since they are "right," but since other things are even more expensive.)



Meta-algorithm for ℓ_2 -norm regression (3 of 3)

(Drineas, Mahoney, etc., 2006, 2008, etc., starting with SODA 2006; Mahoney FnTML, 2011.††)

We can make this meta-algorithm “fast” in RAM:**

- This meta-algorithm runs in $O(mn \log n/\epsilon)$ time in RAM if:
 - ▶ we perform a Hadamard-based random projection and **sample uniformly in a randomly rotated basis**, or
 - ▶ we quickly computing **approximations to the statistical leverage scores** and using those as an importance sampling distribution.

We can make this meta-algorithm “high precision” in RAM:††

- This meta-algorithm runs in $O(mn \log n \log(1/\epsilon))$ time in RAM if:
 - ▶ we use the random projection/sampling basis to **construct a preconditioner and couple with a traditional iterative method**.

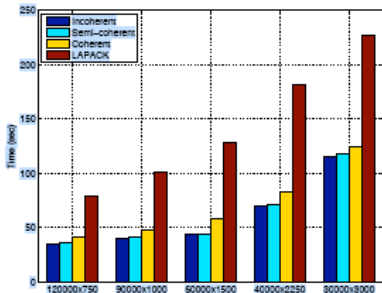
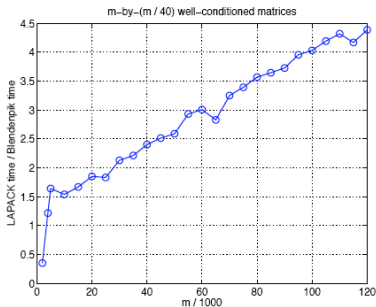
** (Sarlós 2006; Drineas, Mahoney, Muthu, Sarlós 2010; Drineas, Magdon-Ismail, Mahoney, Woodruff 2011.)

†† (Rokhlin & Tygert 2008; Avron, Maymounkov, & Toledo 2010; Meng, Saunders, & Mahoney 2011.)

††† (Mahoney, “Randomized Algorithms for Matrices and Data,” FnTML, 2011.)

Randomized regression in RAM: Implementations

Avron, Maymounkov, and Toledo, SISC, 32, 1217–1236, 2010.



Conclusions:

- *Randomized algorithms “beats Lapack’s direct dense least-squares solver by a large margin on essentially any dense tall matrix.”*
- *These results “suggest that random projection algorithms should be incorporated into future versions of Lapack.”*

Outline

- 1 Overview of implementation issues
- 2 Communication issues in LA and RLA
- 3 High-quality implementations in RAM
- 4 A theoretical aside: low-rank methods
- 5 Solving ℓ_2 regression in RAM
- 6 Solving ℓ_2 regression using MPI**
- 7 Solving ℓ_2 regression in Spark

Algorithm LSRN (for strongly over-determined systems)

(Meng, Saunders, and Mahoney 2011)

- 1: Perform a *Gaussian* random projection
- 2: Construct a preconditioner from the subsample
- 3: Iterate with a “traditional” iterative algorithm

Why a *Gaussian* random projection? Since it

- provides the strongest results for conditioning,
- uses level 3 BLAS on dense matrices and can be generated super fast,
- speeds up automatically on sparse matrices and fast operators,
- still works (with an extra “allreduce” operation) when A is partitioned along its bigger dimension.

Although it is “slow” (compared with “fast” Hadamard-based projections i.t.o. FLOPS), it allows for better communication properties.

Implementation of LSRN

(Meng, Saunders, and Mahoney 2011)

- Shared memory (C++ with MATLAB interface)
 - ▶ Multi-threaded ziggurat random number generator (Marsaglia and Tsang 2000), generating 10^9 numbers in less than 2 seconds on 12 CPU cores.
 - ▶ A naïve implementation of multi-threaded dense-sparse matrix multiplications.
- Message passing (Python)
 - ▶ Single-threaded BLAS for matrix-matrix and matrix-vector products.
 - ▶ Multi-threaded BLAS/LAPACK for SVD.
 - ▶ Using the Chebyshev semi-iterative method (Golub and Varga 1961) instead of LSQR.

Solving real-world problems

| matrix | m | n | nnz | rank | cond | DGELSD | $A \setminus b$ | Blendenpik | LSRN |
|----------|-------|-------|--------|------|-------|--------|-----------------|------------|-------|
| landmark | 71952 | 2704 | 1.15e6 | 2671 | 1.0e8 | 29.54 | 0.6498* | - | 17.55 |
| rail4284 | 4284 | 1.1e6 | 1.1e7 | full | 400.0 | > 3600 | 1.203* | OOM | 136.0 |
| tnimg_1 | 951 | 1e6 | 2.1e7 | 925 | - | 630.6 | 1067* | - | 36.02 |
| tnimg_2 | 1000 | 2e6 | 4.2e7 | 981 | - | 1291 | > 3600* | - | 72.05 |
| tnimg_3 | 1018 | 3e6 | 6.3e7 | 1016 | - | 2084 | > 3600* | - | 111.1 |
| tnimg_4 | 1019 | 4e6 | 8.4e7 | 1018 | - | 2945 | > 3600* | - | 147.1 |
| tnimg_5 | 1023 | 5e6 | 1.1e8 | full | - | > 3600 | > 3600* | OOM | 188.5 |

Table: Real-world problems and corresponding running times. DGELSD doesn't take advantage of sparsity. Though MATLAB's backslash may not give the min-length solutions to rank-deficient or under-determined problems, we still report its running times. Blendenpik either doesn't apply to rank-deficient problems or runs out of memory (OOM). LSRN's running time is mainly determined by the problem size and the sparsity.

Iterating with LSQR

(Paige and Saunders 1982)

Code snippet (Python):

```
u      = A.matvec(v) - alpha*u
beta  = sqrt(comm.allreduce(np.dot(u,u)))
...
v      = comm.allreduce(A.rmatvec(u)) - beta*v
```

Cost per iteration:

- two matrix-vector multiplications
- **two** cluster-wide synchronizations

Iterating with Chebyshev semi-iterative (CS) method

(Golub and Varga 1961)

The strong concentration results on $\sigma^{\max}(AN)$ and $\sigma^{\min}(AN)$ enable use of the CS method, which requires an accurate bound on the extreme singular values to work efficiently.

Code snippet (Python):

```
v = comm.allreduce(A.rmatvec(r)) - beta*v
x += alpha*v
r -= alpha*A.matvec(v)
```

Cost per iteration:

- two matrix-vector multiplications
- **one** cluster-wide synchronization

LSQR vs. CS on an Amazon EC2 cluster

(Meng, Saunders, and Mahoney 2011)

| solver | N_{nodes} | $N_{\text{processes}}$ | m | n | nnz | N_{iter} | T_{iter} | T_{total} |
|--------------|--------------------|------------------------|------|-----|-------|-------------------|-------------------|--------------------|
| LSRN w/ CS | 2 | 4 | 1024 | 4e6 | 8.4e7 | 106 | 34.03 | 170.4 |
| LSRN w/ LSQR | | | | | | 84 | 41.14 | 178.6 |
| LSRN w/ CS | 5 | 10 | 1024 | 1e7 | 2.1e8 | 106 | 50.37 | 193.3 |
| LSRN w/ LSQR | | | | | | 84 | 68.72 | 211.6 |
| LSRN w/ CS | 10 | 20 | 1024 | 2e7 | 4.2e8 | 106 | 73.73 | 220.9 |
| LSRN w/ LSQR | | | | | | 84 | 102.3 | 249.0 |
| LSRN w/ CS | 20 | 40 | 1024 | 4e7 | 8.4e8 | 106 | 102.5 | 255.6 |
| LSRN w/ LSQR | | | | | | 84 | 137.2 | 290.2 |

Table: Test problems on an Amazon EC2 cluster and corresponding running times in seconds. Though the CS method takes more iterations, it actually runs faster than LSQR by making only one cluster-wide synchronization per iteration.

Outline

- 1 Overview of implementation issues
- 2 Communication issues in LA and RLA
- 3 High-quality implementations in RAM
- 4 A theoretical aside: low-rank methods
- 5 Solving ℓ_2 regression in RAM
- 6 Solving ℓ_2 regression using MPI
- 7 Solving ℓ_2 regression in Spark

Distributed setting

“Apache Spark is a fast and general engine for large-scale data processing.”

Here, we assume that dataset is partitioned along the high dimension and stored in a distributed fashion.

The costs of computing in distributed settings:

- floating point operations
- bandwidth costs: \propto total bits transferred
- latency costs: \propto rounds of communication

$$\mathbf{FLOPS}^{-1} \ll \mathbf{bandwidth}^{-1} \ll \mathbf{latency}.$$

We want to make as few passes over the dataset as possible.

Solvers for ℓ_2 regression

- Low-precision solvers: compute a sketch (MapReduce, 1-pass)
+ solve the subproblem (local SVD)
- High-precision solvers: compute a sketch (MapReduce, 1-pass)
+ preconditioning (local QR)
+ invoke an iterative solver (only matrix-vector products are involved which can be well handled by Spark, # pass is proportional to # iteration)

Notes

- Methods for computing sketches are embarrassingly parallel and can be implemented under the MapReduce framework.
- Since the sketch is small, operations like SVD or QR can be performed exactly locally.
- Preconditioning is crucial because we want to reduce the number of iterations in the iterative solver.

Gaussian Transform

Here, $\Phi A = GA$ where $G \in \mathbb{R}^{r \times n}$ consisting of i.i.d. Gaussian random variables:

$$GA = (G_1 \quad \cdots \quad G_p) \begin{pmatrix} A_1 \\ \vdots \\ A_p \end{pmatrix} = \sum_{i=1}^p G_i A_i.$$

Mapper:

- 1: For a block of A , $A_i \in \mathbb{R}^{c \times d}$, generate $G_i \in \mathbb{R}^{r \times c}$ consisting of independent Gaussian random variables.
- 2: Compute $B = G_i A_i \in \mathbb{R}^{r \times d}$.
- 3: For $j = 1, \dots, r$, emit (j, B_j) .

Reducer:

- 1: Reduce vectors associated with key k to \mathbf{v}_k with addition operation.
- 2: Return \mathbf{v}_k .

Other sketches are similar; but lots of work to do to figure out the best in general.

Experimental setup

Sketches

- **PROJ CW** — Random projection with the input-sparsity time CW method (sparse)
- **PROJ GAUSSIAN** — Random projection with Gaussian transform (dense)
- **PROJ RADEMACHER** — Random projection with Rademacher transform (dense)
- **PROJ SRDHT** — Random projection with Subsampled randomized discrete Hartley transform (dense, no longer fast)
- **SAMP APPR** — Random sampling based on approximate leverage scores (fast approximate leverage scores)
- **SAMP UNIF** — Random sampling with uniform distribution (for completeness)

Experimental setup (cont.)

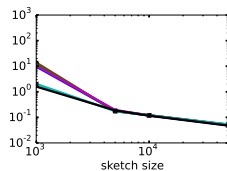
Datasets

- We used synthetic datasets with *uniform* or *nonuniform* leverage scores, *low* or *high* condition number, and *coherent* or *incoherent*.
- Recall that leverage scores can be viewed as a measurement of the importance of the rows in the LS fit.
- Coherence is the largest leverage score.
- These properties of the matrix have a strong influence on the solution quality.

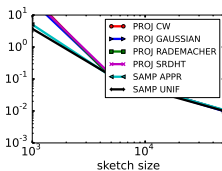
Resources

The experiments were performed in a cluster with 16 nodes (1 master and 15 slaves), each of which has 4 CPU cores at clock rate 2.5GHz with 25GB RAM.

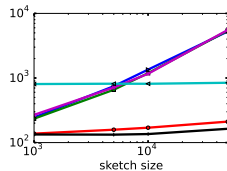
Low-precision solvers: overall evaluation



(a) $\|x - x^*\|_2 / \|x^*\|_2$

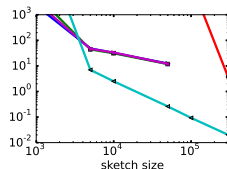


(b) $|f - f^*| / f^*$

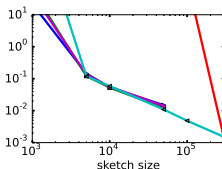


(c) Running time(sec)

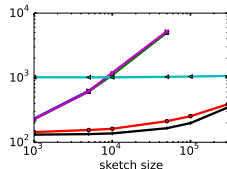
$1e7 \times 1000$ matrix with *uniform* leverage scores



(d) $\|x - x^*\|_2 / \|x^*\|_2$



(e) $|f - f^*| / f^*$

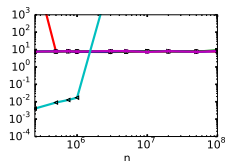


(f) Running time(sec)

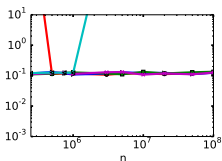
$1e7 \times 1000$ matrix with *nonuniform* leverage scores

Figure: Evaluation of all 6 of the algorithms on matrices with uniform and nonuniform leverage scores.

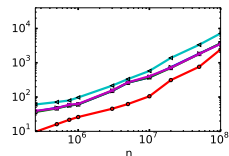
Low-precision solvers: on matrices with increasing n



(a) $\|x - x^*\|_2 / \|x^*\|_2$

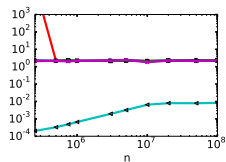


(b) $|f - f^*| / |f^*|$

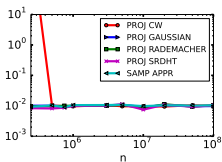


(c) Running time(sec)

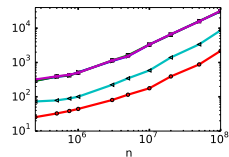
sketch size $s = 5e3$



(d) $\|x - x^*\|_2 / \|x^*\|_2$



(e) $|f - f^*| / |f^*|$

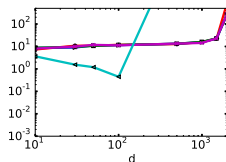


(f) Running time(sec)

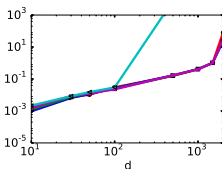
sketch size $s = 5e4$

Figure: Performance of all the algorithms on matrices with *nonuniform* leverage scores, *high* condition number, varying n from $2.5e5$ to $1e8$ and fixed $d = 1000$.

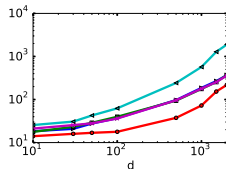
Low-precision solvers: on matrices with increasing d



(a) $\|x - x^*\|_2 / \|x^*\|_2$

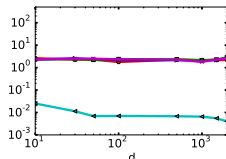


(b) $|f - f^*| / |f^*|$

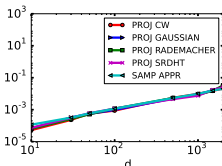


(c) Running time(sec)

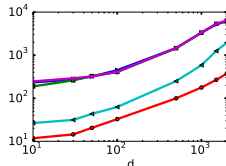
sketch size $s = 2e3$



(d) $\|x - x^*\|_2 / \|x^*\|_2$



(e) $|f - f^*| / |f^*|$



(f) Running time(sec)

sketch size $s = 5e4$

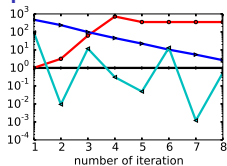
Figure: Performance of all the algorithms on matrices with *nonuniform* leverage scores, *high* condition number, varying d from 10 to 2000 and fixed $n = 1e7$.

High-precision solvers: preconditioning quality

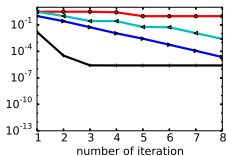
| r | PROJ CW | PROJ GAUSSIAN | SAMP APPR |
|-----|---------|---------------|-----------|
| 5E2 | 1.08E8 | 2.17E3 | 1.21E2 |
| 1E3 | 1.1E6 | 5.7366 | 75.0290 |
| 5E3 | 5.5E5 | 1.9059 | 25.8725 |
| 1E4 | 5.1E5 | 1.5733 | 17.0679 |
| 5E4 | 1.8E5 | 1.2214 | 6.9109 |
| 1E5 | 1.1376 | 1.1505 | 4.7573 |

Table: Quality of preconditioning, i.e., $\kappa(AR^{-1})$, on a matrix with size $1e6$ by 500 and condition number $1e6$ using several kinds of sketch.

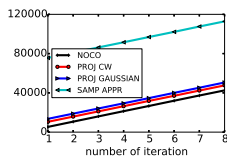
High-precision solvers: on badly conditioned matrix



(a) $\|x - x^*\|_2 / \|x^*\|_2$

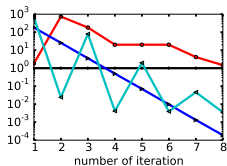


(b) $|f - f^*| / |f^*|$

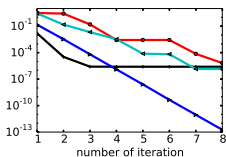


(c) Running time(sec)

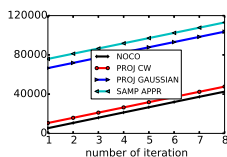
sketch size $s = 5e3$



(d) $\|x - x^*\|_2 / \|x^*\|_2$



(e) $|f - f^*| / |f^*|$

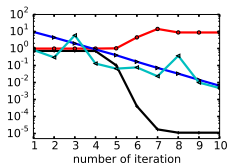


(f) Running time(sec)

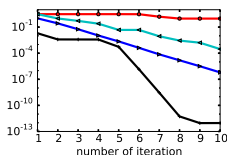
sketch size $s = 5e4$

Figure: LSQR with randomized preconditioner on a matrix with size $1e8$ by 1000 and condition number $1e6$. (Running time is artificially slow and *not* accurate.)

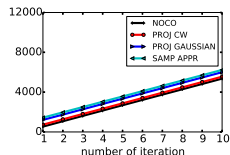
High-precision solvers: on well conditioned matrix



(a) $\|x - x^*\|_2 / \|x^*\|_2$

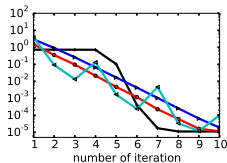


(b) $|f - f^*| / |f^*|$

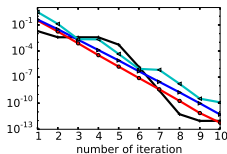


(c) Running time(sec)

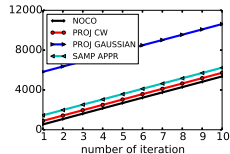
small sketch size



(d) $\|x - x^*\|_2 / \|x^*\|_2$



(e) $|f - f^*| / |f^*|$



(f) Running time(sec)

large sketch size

Figure: LSQR with randomized preconditioner on a matrix with size $1e7$ by 1000 and condition number 5 .

Conclusions and Future Directions

Conclusions

- Short answer: take the basic RLA theory and couple with traditional NLA methods and you get high-quality implementations
 - ▶ “beat” LAPACK in terms of clock time
 - ▶ better (more robust, better communication, faster, etc.) in SC apps
 - ▶ least-squares approximation, low-rank approximation, etc. on terabyte (soon hundreds of terabytes) sized data in AEC2/HPC environments
- Long answer: is much longer ...

Future Directions.

- Communication-computation tradeoffs
- Numerical precision versus statistical precision tradeoffs
- Interacting in nontrivial ways with large-scale data analysis objectives
- Interacting in nontrivial with traditional/nontraditional continuous optimization
- Developing more relevant theory
- ...

More readings ...

- Drineas, Kannan and Mahoney (2006) “Fast Monte Carlo Algorithms for Matrices I: Approximating Matrix Multiplication,” *SIAM J. Comput.*, vol. 36, no. 1, pp 132-157
- Drineas, Mahoney, Muthukrishnan and Sarlos (2011) “Faster Least Squares Approximation,” *Numer. Math.*, vol. 117, pp 219-249
- Mahoney (2011) *Randomized Algorithms for Matrices and Data*, Found. Trends Mach. Learning, vol. 3, no. 2, pp 123-224
- Tropp (2015) *An Introduction to Matrix Concentration Inequalities*, Found. Trends Mach. Learning, vol. 8, no 1-2, pp 1-230
- Holodnak and Ipsen (2015) “Randomized Approximation of the Gram Matrix: Exact Computation and Probabilistic Bounds,” *SIAM J. Matrix Anal. Appl.*, vol. 36, no. 1, pp 110-137