Randomized Numerical Linear Algebra: Sampling for linear algebra, statistics, and optimization

Michael W. Mahoney

ICSI and Dept of Statistics, UC Berkeley

http://www.stat.berkeley.edu/∼mmahoney/

August 2018

Outline

RandNLA: Randomized Numerical Linear Algebra

Matrices provide a natural structure with which to **model data**.

- $A \in \mathbb{R}^{m \times n}$ can encode information about *m* objects, each of which is described by *n features*; etc.
- A positive definite $A \in \mathbb{R}^{n \times n}$ can encode the correlations/similarities between all pairs of *n* objects; etc.

Motivated by data problems, recent years have witnessed **many exciting developments** in the theory and practice of matrix algorithms.

- Particularly remarkable is the use of *randomization*.
- Typically, it is assumed to be a property of the input data due (*e.g.*, to noise in the data generation mechanisms).
- Here, it is used as an algorithmic or computational resource.

RandNLA: Randomized Numerical Linear Algebra

An *interdisciplinary research area that exploits randomization as a computational resource to develop improved algorithms for large-scale linear algebra problems*.

- Foundational perspective: roots in theoretical computer science (TCS); deep connections with convex analysis, probability theory, and metric embedding theory, etc.; and strong connections with scientific computing, signal processing, and numerical linear algebra (NLA).
- Implementational perspective: well-engineered RandNLA algorithms beat highly-optimized software libraries for problems such as very over-determined least-squares and scale well to parallel/distributed environments.
- Data analysis perspective: strong connections with machine learning and statistics and many "non-methodological" applications of data analysis.

Growing interest in providing an *algorithmic and statistical foundation for modern large-scale data analysis*.

An historical perspective

Linear algebra has had a long history in large-scale (by the standards of the day) statistical data analysis.

- Method of least-squares (LS): due to Gauss, Legendre, and others; and used in early 1800s for fitting linear equations to determine planetary orbits.
- Principal Component Analysis (PCA) and low-rank approximations: due to Pearson, Hotelling, and others, and used in early 1900s for exploratory data analysis and predictive analytics.

These and related methods are of interest since, *e.g.*, if there is noise or randomness *in the data* then the leading principle components tend to capture the signal and remove the noise.

An historical perspective

Advent of the digital computer in the 1950s:

- Proto computer science and early applications of linear algebra focused on scientific computing problems (where computation was an essential tool)
- Even for "well-posed" problems, many algorithms perormed very poorly in the presence of the finite precision.
- Work by Turing, von Neumann, and others laid much of the foundations for scientific computing and NLA: this led to problem-specific complexity measures (*e.g.*, the condition number) that characterize the behavior of an input for a specific class of algorithms (*e.g.*, iterative algorithms).

But . . . (for various technical and nontechnical reasons), there then occured a split in the nascent field of computer science:

- Continuous linear algebra became the domain of applied mathematics.
- Computer science theory and practice became discrete and combinatorial.

An historical perspective

Linear algebra became the domain of continuous applied mathematics; and it focused on scientific applications.

- Nearly all work in scientific computing and NLA has been deterministic; this led to high-quality codes in the 1980s/1990s, *e.g.*, LAPACK.
- Most work focused on optimizing FLOPS—matrix-vector multiplies on dense matrices—in shared memory environments on matrices that arise in structured scientific computing applications.
- This code is now widely-used in NLA and scientific computing as well as in machine learning, statistics, data analysis, etc.

Computer science became discrete and combinatorial; and it focused on business and commerce applications.

- Turing, Church, and other studied computation *per se*—seemingly-different approaches (recursion theory, the λ-calculus, and Turing machines) defined the same class of functions
- Belief arose that the concept of computability is formally captured in a qualitative and robust way by these three equivalent processes, *independent of the input data*.
- Randomization (where the randomness is *inside the algorithm*, and the algorithm is applied to arbitrary or worst-case data) was introduced and exploited as a powerful computational resource.

An historical perspective: now and going forward . . .

Recently, a **convergence of these two very different perspectives**.

- Motivated by scientific, Internet, social media, financial, etc. applications.
- Computation *per se* is necessary but very insufficient.
- Most people want to *obtain insight* and/or *make predictions* from the data they generate to make downstream claims about the world.

Central to these developments RandNLA, including:

- Randomness in the data versus randomness in the algorithm.
- Continuous (mathematics) versus discrete (computer science).
- Worst-case algorithms versus problem-specific complexity measures.
- Scientific versus business/commerce applications.

Good "hydrogen atom" to consider algorithmic and statistical foundations of modern large-scale data analysis.

Basic RandNLA Principles

Drineas and Mahoney, CACM, 2016

Basic RandNLA method: given an input matrix:

- Construct a "sketch" (a smaller or sparser matrix matrix that represents the essential information in the original matrix) by random sampling.
- Use that sketch as a surrogate to compute quantities of interest.

Basic design principles* underlying RandNLA:

- Randomly sample (in a careful data-dependent manner) a small number of elements to create a much sparser sketch of the original matrix.
- Randomly sample (in a careful data-dependent manner) a small number of columns and/or rows to create a much smaller sketch of the original matrix.
- Preprocess an input matrix with a random-projection-type matrix and then do uniform sampling of rows/columns/elements in order to create a sketch.

*The first two principles deal with identifying nonuniformity structure. The third principle deals with preconditioning the input (*i.e.*, uniformizing nonuniformity structure) s.t. uniform random sampling performs well.

Element-wise Sampling

Drineas and Mahoney, CACM, 2016

- An $m \times n$ matrix A is an array of numbers, A_{ij}, $\forall i \in [m], \forall j \in [n]$.
- Randomly sample a small number of entries, each w.r.t. importance sampling probability distribution p_{ij}.
- Return a sparse matrix $\tilde{A}$ that contains precisely the (rescaled) entries.
- Uniform sampling easily leads to poor results; but non-uniform sampling w.r.t. magnitudes or element-wise leverage scores gives nontrivial results.
- Thm [AM01/AM07/DZ11]: If sample s elements with $p_{ij} = \frac{A_{ij}^2}{\sum_{i,j} A_{ij}^2}$, then

$$\|A - \tilde{A}\|_2 \leq O\left(\sqrt{\frac{(m+n)\ln(m+n)}{s}}\right)\|A\|_F.$$

  This gives *"additive-error" bounds* for low-rank matrix approximation.
- Proof method: $A - \tilde{A}$ is a random matrix; use random matrix theory, combinatorial moment methods, matrix measure concentration bounds.

Row/column Sampling

Drineas and Mahoney, CACM, 2016

- An $m \times n$ matrix A is a linear operator, with column/row spaces.
- Randomly sample a small number of rows, each w.r.t. importance sampling probability distribution $\{p_i\}_{i=1}^m$.
- Return $s \times n$ matrix $\tilde{A}$, an approximation to A, containing s (rescaled) rows.
- Uniform sampling easily leads to poor results; but non-uniform sampling w.r.t. magnitudes or leverage scores gives nontrivial results.
- Thm [FVK97/DKM05/RV06]: If sample s rows with $p_i = \frac{\|A_{(i)}\|^2}{\sum_{i,j} A_{ij}^2}$, then

$$\|A^T A - \tilde{A}^T \tilde{A}\|_F \leq \frac{1}{\sqrt{s}} \|A\|_F^2.$$

  This gives *"additive-error" bounds* for low-rank matrix approximation.
- Proof method: expectations and variances for $\|\cdot\|_F$; Khintchine inequality or matrix-Bernstein inequalities for $\|\cdot\|_2$ extension.

Row/column Sampling

Drineas and Mahoney, CACM, 2016

- Norm-squared sampling does only comparable to element-wise sampling.
  - (I.e., element-wise sampling does only comparable to *very coarse* norm-squared sampling.)
- Leverage score sampling does *much* better: say $m \gg n$, then let

$$p_i = \frac{1}{n}(P_A)_{ii} = \frac{1}{n}\|U_{(i)}\|_2^2,$$

  where U is any $m \times n$ orthogonal matrix spanning the column space of A.
- These *statistical leverage scores*
  - are useful in regression diagnostics to identify outliers
  - approximatable without computing U in "random projection time"
  - give *"relative-error" bounds* for least-squares & low-rank approximation
  - provide data-aware subspace embedding: fix $\epsilon \in (0,1)$, $s \gtrsim \frac{n \log(n)}{\epsilon}$ then

$$\|U^T U - (SU)^T SU\|_2 = \|I - (SU)^T SU\| \leq \epsilon.$$

  (For NLA, this is an acute perturbation; for TCS this is a subspace JL.)

Random Projections as Preconditioners[†]

Drineas and Mahoney, CACM, 2016

- **Main challenge** for uniform sampling: relevant information could be *localized* on a small number of rows/columns/elements.
- Main challenge for non-uniform sampling: construct sampling probabilities.
- **One solution**: *spread out* this information, so uniform sampling does well.
- Bicriteria:
  - Preprocessed matrix should be similar to the original matrix.
  - Preprocessing should be computationally efficient to perform.
- Do this preconditioning with random projections:
  - Pre-/post-multiply by appropriately-scaled random matrix (i.i.d. Gaussians, i.i.d. Rademacher, Hadamard-based constructions, etc.)
  - Can get data-oblivious subspace embedding: fix $\epsilon \in (0, 1)$, then

$$\|U^T U - (\Pi U)^T \Pi U\|_2 = \|I - (\Pi U)^T \Pi U\| \leq \epsilon.$$

(For NLA, this is an acute perturbation; for TCS this is a subspace JL.)

[†] Preconditioners: a transformation that converts a problem instance into another instance that is more-easily solved by a given class of algorithms.

Outline

Approximating Matrix Multiplication

Problem Statement: Given an $m \times n$ matrix A and an $n \times p$ matrix B, approximate the product $A \cdot B$.

Approximating Matrix Multiplication

Problem Statement: Given an $m \times n$ matrix A and an $n \times p$ matrix B, approximate the product $A \cdot B$.

OR, equivalently,

Problem Statement: Approximate the sum of n rank-one matrices.

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

Approximating Matrix Multiplication

A sampling approach:

1. Fix a set of probabilities p_i, $i = 1, \ldots, n$, summing up to 1.
2. For $t = 1, \ldots, c$,
   set $j_t = i$, where $\mathbb{P}[j_t = i] = p_i$.
   (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.

$$A \cdot B = \sum_{k=1}^{n} \begin{pmatrix} \\ A_{*k} \\ \\ \end{pmatrix} \cdot \begin{pmatrix} \quad B_{k*} \quad \end{pmatrix} \approx \sum_{t=1}^{c} \frac{1}{c p_{j_t}} \begin{pmatrix} \\ A_{*j_t} \\ \\ \end{pmatrix} \cdot \begin{pmatrix} \quad B_{j_t *} \quad \end{pmatrix}$$

Approximating Matrix Multiplication

The same algorithm, in matrix notation:

1. Pick c columns of A to form an $m \times c$ matrix C and the corresponding c rows of B to form a $c \times p$ matrix R.
2. Rescale the columns/rows prior to including them in C/R.
3. Approximate $A \cdot B$ by $C \cdot R$.

$$\underset{m\times n}{\left(\quad A \quad\right)}\left(\quad B \quad\right) \approx \left(\quad C \quad\right)\underset{c\times p}{\left(\quad R \quad\right)}$$

Can use a "sampling matrix" formalism:

- Let S be $n \times c$ matrix whose t^{th} column ($t = 1, \ldots, c$) has one non-zero:

$$S_{j_t t} = \frac{1}{\sqrt{c p_{j_t}}}$$

- Clearly: $A \cdot B \approx C \cdot R = (AS) \cdot \left(S^T B\right)$.

Approximating Matrix Multiplication

Some simple lemmas:

- For any sampling probabilities:

$$\begin{aligned} \mathbb{E}[(CR)_{ij}] &= (AB)_{ij} \\ \mathrm{Var}[(CR)_{ij}] &= \frac{1}{c}\sum_{k=1}^{n}\frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{c}(AB)_{ij}^2 \end{aligned}$$

- From these, it's easy to bound $\mathbb{E}\left[\|AB - CR\|_F\right]$.
- Remove the expectation with Markov's inequality or a martingale argument.
- To minimize $\mathbb{E}\left[\|AB - CR\|_F\right]$, use these probabilities:

$$\mathbb{P}[j_t = i] = \frac{\|A_{*i}\|_2 \|B_{i*}\|_2}{\sum_{j=1}^{n} \|A_{*j}\|_2 \|B_{j*}\|_2} \tag{1}$$

- This gives:

$$\mathbb{E}\left[\|AB - CR\|_F\right] = \mathbb{E}\left[\|AB - ASS^T B\|_F\right] \leq \frac{1}{\sqrt{c}}\|A\|_F \|B\|_F \tag{2}$$

- Similar bounds to (2) if approximate probabilities (1) in one of many ways.

Approximating Matrix Multiplication

This *Frobenius norm* bound is used in *many* places in RandNLA, but ...

a "better" *spectral norm* bound is possible via Chernoff/Bernstein inequalities.

Lemma (DMMS, Num Math 2011, Thm 4)

Assume:

- $\|A\|_2 \leq 1$: *("not important," just normalization)*
- $\|A\|_F \geq 0.2$: *("not important," simplifies bounds)*

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 $\epsilon \in (0,1)$, *w.p.* $\geq 1-\delta$, *we have:*

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

Approximating Matrix Multiplication

The spectral norm bound is "better," but:

- It only holds for $B = A^T$, so it doesn't hold for arbitrary AB.
- The "not important" conditions mean it doesn't hold for arbitrary A.

The "main use case" for the spectral norm bound:

- Let A^T be an $n \times d$ matrix U with orthonormal columns, where $n \gg d$.
- Then $U^T U = I_d$, and we want to show that

$$\|U^T S S^T U - U^T U\|_2 = \|U^T S S^T U - I_d\|_2 \leq \epsilon \in (0, 1).$$

- Using the Frobenius norm bound, we get

$$\|U^T S S^T U - I\|_2 \leq \|U^T S S^T U - I\|_F \leq \frac{1}{\sqrt{c}} \|U\|_F^2 = \frac{d}{\sqrt{c}}.$$

- Using the spectral norm bound, we get

$$\|U^T S S^T U - I\|_2 \lesssim \frac{\ln c}{\sqrt{c}} \|U\|_F \|U\|_2 = \frac{\sqrt{d} \ln c}{\sqrt{c}}.$$

Approximating Matrix Multiplication

Similar results for many "dense sampling matrix" constructions:

- Natural interpretation as a random projection or random sketch:
  - Recall David Woodruff's and Ken Clarkson's presentations yesterday.
- Natural interpretation in terms of preconditioning/preprocessing:
  - We'll discuss below for least-squares approximation.

Subspace Embeddings

(Mahoney, FnTML, 2011; Woodruff, FnTML, 2014.)

Definition

Let U be an $m \times n$ orthogonal matrix, and let S be any $n \times m$ matrix. Then, S is a *subspace embedding* if

$$\|U^T U - (SU)^T SU\|_2 = \|I - (SU)^T SU\|_2 \leq \epsilon.$$

Things to note:

- Many constructions (random sampling and projection methods, deterministic constructions, hasing functions, etc.) satisfy this condition.
- First used in data-aware context with leverage score sampling (DMM06, DMM08)
- Used in data-oblivious context with Hadamard-based projections (S06, DMMS08)
- For NLA, this is an acute perturbation.
- For TCS, this is a subspace analogue of JL lemma.

This is a "must must have" for TCS; for everyone else, it's optional.

- Numerical implementations: loosing rank still gives a good preconditioner.
- Statistics and machine learning: loosing rank introduces a bit of bias.

Outline

Least-squares approximation

Least-squares (LS) : given $m \times n$ matrix A and m-dimensional vector b, solve

$$x_{opt} = \arg\min_{x \in \mathbb{R}^n} \|Ax - b\|_2.$$

- If $m \gg n$, it is overdetermined/overconstrained.
- Compute solution in $O(mn^2)$ time (in RAM model) with one of several methods: normal equations; QR decompositions; or SVD.
- RandNLA provides faster algorithms for this ubiquitous problem.
  - **TCS**: faster in terms of low-precision asymptotic worst-case theory.
  - **NLA**: faster in terms of high-precision wall-clock time.
  - **Implementations**: can compute (in Spark/MPI/etc.) low, medium, and high precision solutions on up to terabyte-sized data.
  - **Data Applications**: faster algorithms and/or implicit regularization for many machine learning and data science problems.
- *The basic RandNLA approach extends to many other matrix problems.*

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, etc.

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, that fails with probability δ, in roughly $O(mn^2/\epsilon)$ time (DMM 2006, 2008). (Ugh—seems bad—why would one do this?)

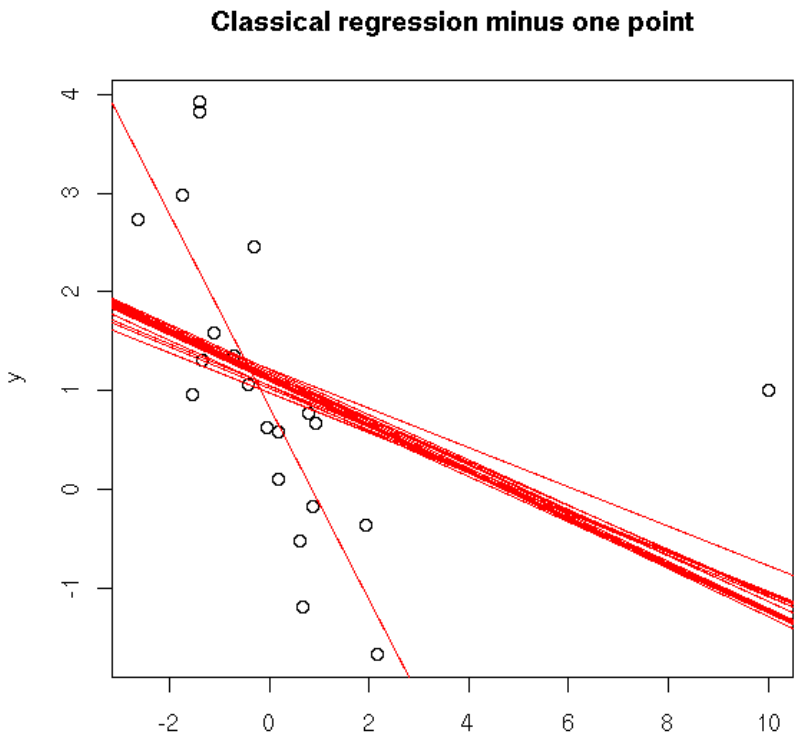
A *non-naïve* version of this meta-algorithm:

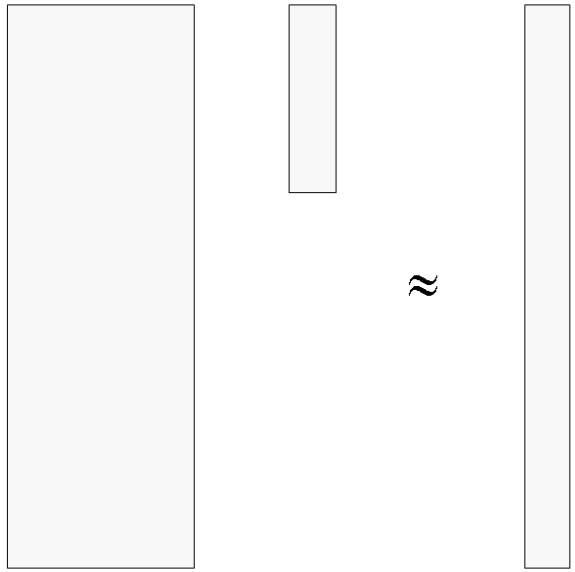
- gives the best worst-case algorithm in RAM.
- beats LAPACK for high precision in wall-clock time.
- super-terabyte-scale implementations in parallel/distributed environments.
- provides the foundation for low-rank approximations and the rest of RandNLA.

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

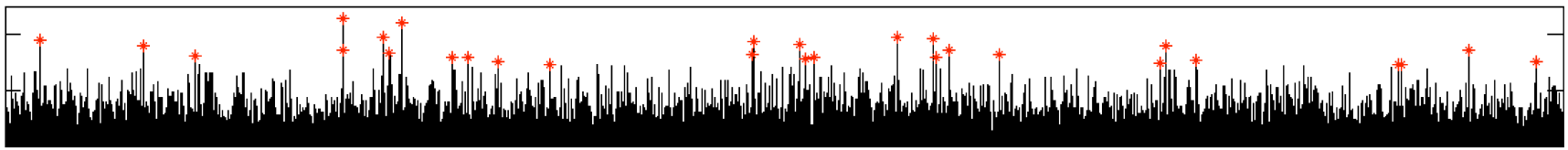
(Drineas, Mahoney, etc., 2006, 2008, etc., starting with SODA 2006; Mahoney FnTML, 2011.)

- Randomly sample high-leverage constraints
- Solve the subproblem





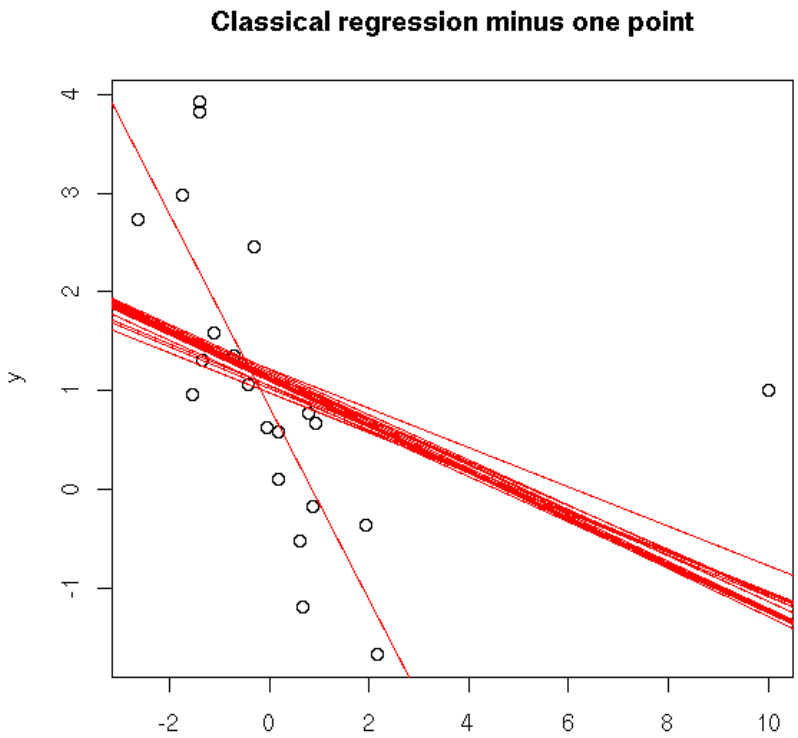
(In many moderately large-scale applications, one uses "ℓ_2 objectives," not since they are "right," but since other things are even more expensive.)

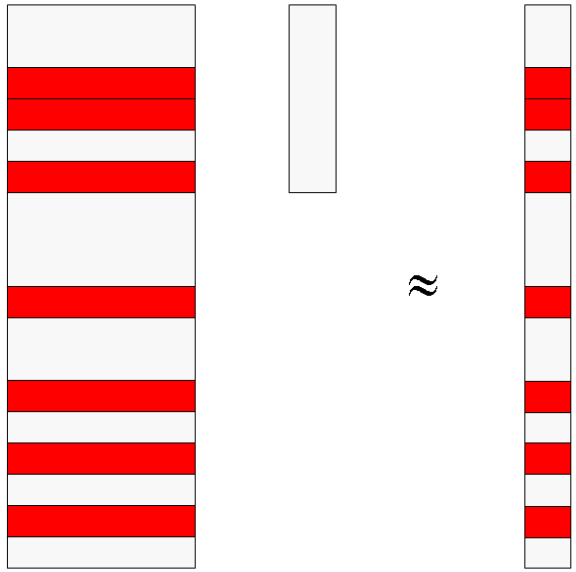


Meta-algorithm for ℓ_2-norm regression (2 of 3)

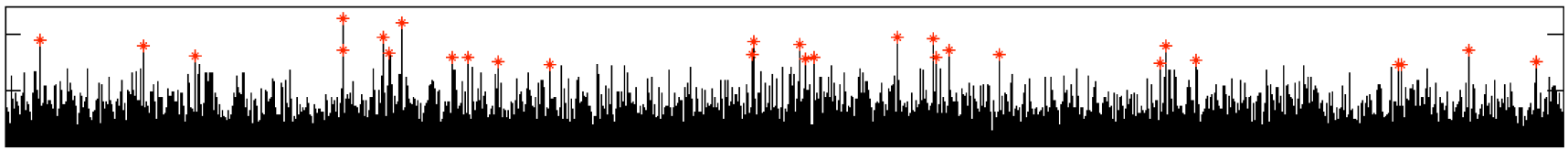
(Drineas, Mahoney, etc., 2006, 2008, etc., starting with SODA 2006; Mahoney FnTML, 2011.)

- Randomly sample high-leverage constraints
- Solve the subproblem





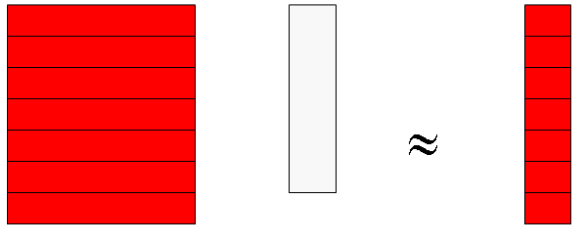
(In many moderately large-scale applications, one uses "ℓ_2 objectives," not since they are "right," but since other things are even more expensive.)

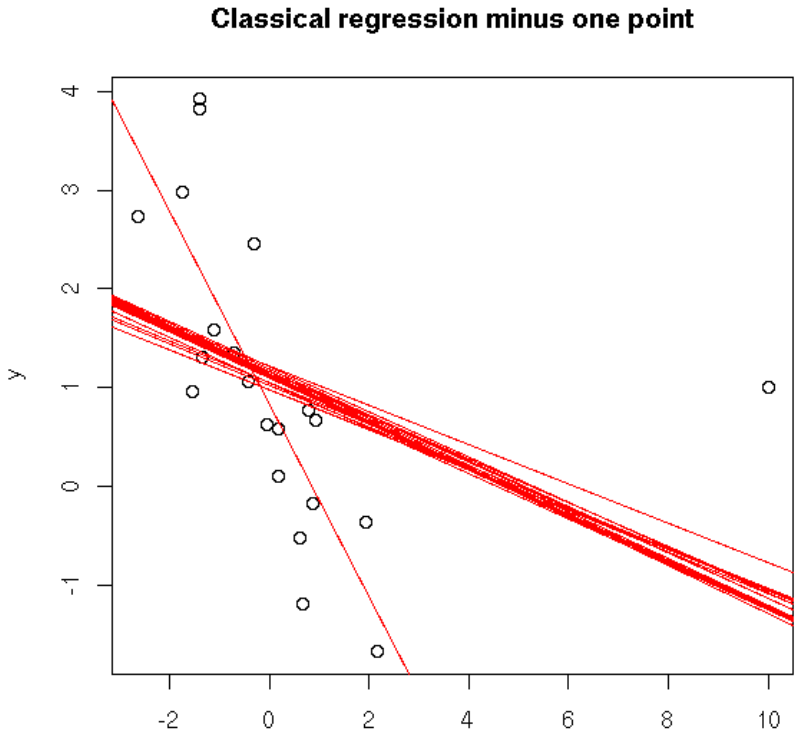


Meta-algorithm for ℓ_2-norm regression (2 of 3)

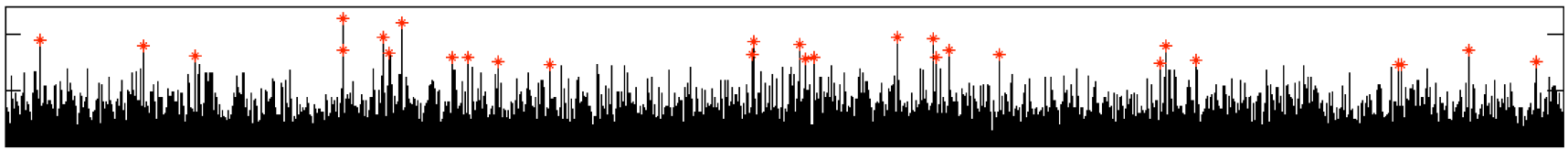
(Drineas, Mahoney, etc., 2006, 2008, etc., starting with SODA 2006; Mahoney FnTML, 2011.)

- Randomly sample high-leverage constraints
- Solve the subproblem





(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 random projection and sample uniformly sampling in the 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 algorithm.
- See Blendenpik/LSRN for NLA-style wall-clock time comparisons.

Both can be improved (in theory) to run in almost $O(\mathbf{nnz}(A))$ time.

[‡] (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.)

Least-squares approximation: the basic structural result

Consider the over-determined least-squares approximation problem:

$$\mathcal{Z}_2^2 = \min_{x \in \mathbb{R}^n} \|b - Ax\|_2^2 = \|b - Ax_{opt}\|_2^2$$

as well as the "preconditioned " the least-squares approximation problem:

$$\tilde{\mathcal{Z}}_2^2 = \min_{x \in \mathbb{R}^n} \|\Omega(b - Ax)\|_2^2 = \|b - A\tilde{x}_{opt}\|_2^2$$

where Ω is *any* matrix.

Theorem (Fundamental Structural Result for Least-Squares)

If Ω satisfies the two basic conditions (constants are somewhat arbitrary):

$$\begin{aligned} \sigma_{min}^2(\Omega U_A) &\geq 1/\sqrt{2} \\ \left\| U_A^T \Omega^T \Omega b^\perp \right\|_2^2 &\leq \epsilon \mathcal{Z}_2^2/2, \quad \text{where } b^\perp = b - U_A U_A^T A, \end{aligned}$$

then:

$$\begin{aligned} \|A\tilde{x}_{opt} - b\|_2 &\leq (1+\epsilon)\mathcal{Z}_2 \\ \|x_{opt} - \tilde{x}_{opt}\|_2 &\leq \frac{1}{\sigma_{min}(A)} \sqrt{\epsilon} \mathcal{Z}_2. \end{aligned}$$

Least-squares approximation: satisfying the two conditions

Both conditions are an approximate matrix-matrix multiplication result:

- First condition:

$$\|U_A^T U_A - U_A^T \Omega \Omega^T U_A\|_2^2 = \|I - U_A^T \Omega \Omega^T U_A\|_2^2 \leq \epsilon,$$

  w.p. $\geq 1 - \delta$, if $r = O\left(\frac{n}{\epsilon^2} \ln\left(\frac{n}{\epsilon^2 \sqrt{\delta}}\right)\right)$.

- Second condition:

$$\mathbb{E}\left[\|U_A^T \Omega \Omega^T b^\perp - U_A^T b^\perp\|_2^2\right] \leq \frac{1}{r}\|U_A\|_F^2 \|b^\perp\|_2^2 = \frac{n}{r}\mathcal{Z}_2^2,$$

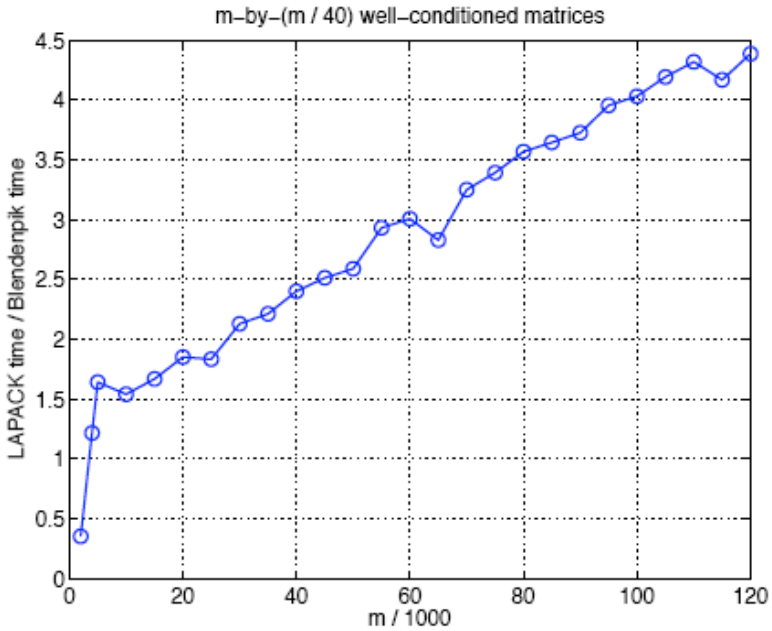
  and remove expectation with Markov.

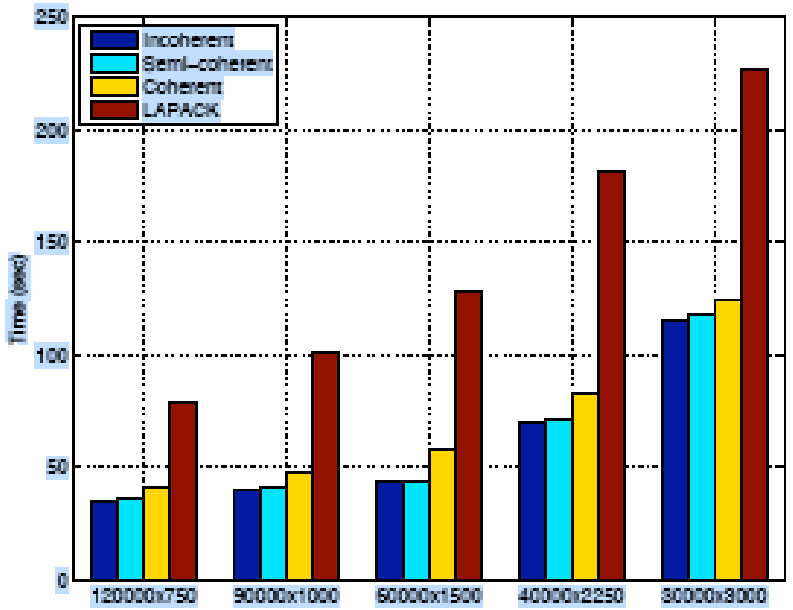
Things to note:

- Many constructions (random sampling and projection methods, deterministic constructions, hasing functions, etc.) satisfy these conditions.
- Which construction you use depends on which you like.
- ϵs don't matter: TCS people don't care; NLA people precondition; ML/DA poeple have different pain points

Least-squares approximation: 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

Extensions to Low-rank Approximation (Projections)

(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 $U = Q\hat{U}$.

Can 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?**

Extensions to Low-rank Approximation (Sampling)

(Boutsidis, Mahoney, Drineas, CSSP, 2009; Mahoney and Drineas, "Structural properties," 2016.)

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

Lemma (Fundamental Structural Result for Low-Rank)

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,

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

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 SPSD matrices (GM13)
- developing improved feature selection methods (many)
- other low-rank matrix approximation methods

Extensions to Low-rank Approximation (SPSD Matrices)

Gittens and Mahoney, "Revisiting the Nystrom Method ...," TR 2013; ICML 2014; JMLR 2015

- *SPSD Sketching Model.* Let A be an $n \times n$ positive semi-definite matrix, and let S be a matrix of size $n \times \ell$, where $\ell \ll n$. Take

$$C = AS \quad \text{and } W = S^T AS.$$

  Then CW^+C^T is a low-rank approximation to A with rank at most ℓ.

Lemma (Fundamental Structural Result for SPSD Low-Rank)

Let A be an $n \times n$ SPSD matrix s.t. $A = U\Sigma U^T$, where U_1 is top k eigenvalues, $\Omega_1 = U_1^T S$, etc., and let S be a sampling/sketching matrix of size $n \times \ell$. Then

$$\begin{aligned}
\|A - CW^\dagger C^T\|_2 &\leq \|\Sigma_2\|_2 + \|\Sigma_2^{1/2}\Omega_2\Omega_1^\dagger\|_2^2, \\
\|A - CW^\dagger C^T\|_F &\leq \|\Sigma_2\|_F + \sqrt{2}\|\Sigma_2\Omega_2\Omega_1^\dagger\|_F + \|\Sigma_2^{1/2}\Omega_2\Omega_1^\dagger\|_F^2 \\
\|A - CW^\dagger C^T\|_{Tr} &\leq Tr(\Sigma_2) + \|\Sigma_2^{1/2}\Omega_2\Omega_1^\dagger\|_F^2
\end{aligned}$$

assuming Ω_1 has full row rank.

- From this, easy to derive additive-error approximations for spectral and Frobenius norm (with scale set by Trace norm error) and relative-error approximation for Trace norm in "random projection time."

Outline

Extensions and Applications of Basic RandNLA Principles

Drineas and Mahoney, CACM, 2016

- High-precision numerical implementations:
  - Use sketches to construct preconditioners for iterative algorithms.
- Matrix completion:
  - Reconstruct unobserved entries from hypothesized matrix under incoherence assumptions with heavier-duty methods.
- Solving systems of Laplacian-based linear equations:
  - Approximate effective resistance with graph-theoretic techniques to get near linear time solvers for Laplacian SPSD matrices.
- Machine learning:
  - Interested in uses for kernel learning (then) and neural networks (now).
- Statistics:
  - Connections with factor models, GLMs, experimental design, regression diagnostics, asymptotic analysis, consistency issues, sparsity issues.
- Optimization:
  - Sample gradient and/or Hessian in first-order or second-order methods.

Outline

Statistics versus machine learning

Operationally, let's say the following.

- **Statistics** is what statisticians do.
- **Machine learning** is what machine learners do.

For us, the point is the following.

- Differences are often (not always) more cultural that technical.
- Cultural differences are significant.
- Differences are also nonstationary, with some convergence.
- The two groups so far interact with RandNLA in different ways.

Outline

Lots of related work

Historically, *a lot* of work in traditional statistics:

- Resampling methods such as the bootstrap and jackknife: Jaeckel (1972), Miller (1974), Efron (1979), Wu (1986), Shao and Tu (1995), etc.
- Goal is traditionally to perform statistical inference and not to improve the running time of an algorithm.
- Samples are of similar size to that of the full data, e.g., $\Omega(n)$, $\Omega(n^{1/2})$, etc.

More recently, in machine learning and data analysis:

- Kleiner, Talwalkar, Sarkar, and Jordan, ICML12.
- Qin and Rohe, NIPS13.
- Dhillon, Lu, Foster, and Ungar, NIPS13.
- Hsu, Kakade, and Zhang, FoCM14.
- Ma, Mahoney, and Yu, TR13, ICML14, JMLR15.
- Raskutti and Mahoney, TR14, ICML15, JMLR16.
- ...

Goal is improved inference and/or improved running time.

A statistical perspective on algorithmic leveraging

(Ma, Mahoney, and Yu 2013)

Consider the model

$$y = X\beta_0 + \epsilon,$$

where[‖] y is an $n \times 1$ response vector, X is an $n \times p$ *fixed* predictor/design matrix, β_0 is a $p \times 1$ coefficient vector, and the noise vector $\epsilon \sim N(0, \sigma^2 I)$. Then,

$$\begin{aligned}
\hat{\beta}_{ols} &= \text{argmin}_\beta ||y - X\beta||^2 = (X^T X)^{-1} X^T y \\
\hat{y} &= Hy, \text{ where } H = X(X^T X)^{-1} X^T \\
h_{ii} &= \sum_{j=1}^{p} U_{ij}^2 = ||U_{(i)}||^2 \text{ is the leverage of the } i^{th} \text{ point}
\end{aligned}$$

[‖] The hardest part is remembering $\min_x \|Ax - b\|_2 \Leftrightarrow \min_\beta \|X\beta - y\|_2$.

Recall the main "algorithmic leveraging" result

(Refs in Mahoney FnTML, 2011.)

1: Randomly sample $r > p$ constraints (rows of X and elements of y), using $\{\pi_i\}_{i=1}^n$ as an importance sampling distribution.
2: Rescale each sampled row/element by $1/r\pi_i$ to form a weighted LS subproblem $\text{argmin}_{\beta \in \mathbb{R}^p} ||DS_X^T y - DS_X^T X\beta||^2$.
3: Solve the weighted LS subproblem and return the solution $\tilde{\beta}_{ols}$.

Theorem (DMM06)

If $\pi_i \geq \gamma \frac{h_{ii}}{p}$, for a parameter $\gamma \in (0, 1]$, and if $r = O(p \log(p)/\gamma\epsilon)$, then, with constant probability (with respect to the random choices made by the algorithm), relative-error bounds of the form

$$\begin{aligned} ||y - X\tilde{\beta}_{ols}||_2 &\leq (1+\epsilon)||y - X\hat{\beta}_{ols}||_2 \quad \textit{and} \\ ||\hat{\beta}_{ols} - \tilde{\beta}_{ols}||_2 &\leq \sqrt{\epsilon}\left(\kappa(X)\sqrt{\xi^{-2} - 1}\right)||\hat{\beta}_{ols}||_2 \end{aligned}$$

hold, where $\xi = ||UU^T y||_2/||y||_2$.

Constructing the subsample

(Mahoney FnTML, 2011; Ma, Mahoney, and Yu 2013.)

1: Randomly sample $r > p$ constraints (rows of X and elements of y), using $\{\pi_i\}_{i=1}^n$ as an importance sampling distribution.
2: Rescale each sampled row/element by $1/r\pi_i$ to form a weighted LS subproblem $\text{argmin}_{\beta \in \mathbb{R}^p} ||DS_X^T y - DS_X^T X\beta||^2$.
3: Solve the weighted LS subproblem and return the solution $\tilde{\beta}_{ols}$.

We consider the empirical performance of several versions:

- UNIF: sample uniformly (rescaling doesn't matter)
- BLEV: sample (and rescale) with "expensive" *exact* leverage scores
- ALEV: sample (and rescale) with "fast" *approximate* leverage scores
- SLEV: sample (and rescale) with $0.9lev + 0.1unif$
- UNWL: sample with leverage scores but don't reweight subproblem

Bias and variance of subsampling estimators (1 of 3)

(Ma, Mahoney, and Yu 2013)

The estimate obtained by solving the subproblem is:

$$\begin{aligned}\tilde{\beta}_\Omega &= (X^T S_X D^2 S_X^T X)^{-1} X^T S_X^T D^2 S_X y \\ &= (X^T W X)^{-1} X^T W y,\end{aligned}$$

where Ω refers to the sampling/resacling process. This depends on subsampling through a nonlinear function, the inverse of random sampling matrix, so do a Taylor series expansion.

Lemma (MMY13)

A Taylor expansion of $\tilde{\beta}_\Omega$ around the point $w_0 = 1 = \mathbf{E}\{w\}$ yields

$$\tilde{\beta}_\Omega = \hat{\beta}_{ols} + (X^T X)^{-1} X^T Diag\{\hat{e}\}(w-1) + R_\Omega,$$

where $\hat{e} = y - X\hat{\beta}_{ols}$ is the LS residual vector, and where R_Ω is the Taylor expansion remainder.

Bias and variance of subsampling estimators (2 of 3)

(Ma, Mahoney, and Yu 2013)

Lemma (MMY13)

The **conditional** *expectation/variance for algorithmic leveraging procedure is given by:*

$$\mathbf{E_w}\left[\tilde{\beta}_\Omega|y\right]=\hat{\beta}_{ols}+\mathbf{E_w}\left[R_\Omega\right];$$

$$\mathbf{Var_w}\left[\tilde{\beta}_\Omega|y\right]=(X^TX)^{-1}X^T\left[Diag\left\{\hat{e}\right\}Diag\left\{\frac{1}{r\pi}\right\}Diag\left\{\hat{e}\right\}\right]X(X^TX)^{-1}+\mathbf{Var_w}\left[R_\Omega\right],$$

where Ω *specifies the sampling/rescaling probability distribution.*
The **unconditional** *expectation/variance for the is given by:*

$$\mathbb{E}\left[\tilde{\beta}_\Omega\right]=\beta_0+\mathbb{E}\left[R_\Omega\right];$$

$$\mathbf{Var}\left[\tilde{\beta}_\Omega\right]=\sigma^2(X^TX)^{-1}+\frac{\sigma^2}{r}(X^TX)^{-1}X^TDiag\left\{\frac{(1-h_{ii})^2}{\pi_i}\right\}X(X^TX)^{-1}+\mathbf{Var}\left[R_\Omega\right].$$

Bias and variance of subsampling estimators (3 of 3)

(Ma, Mahoney, and Yu 2013)

So, for any sampling/rescaling probability distribution:

- Conditional/unconditional estimates unbiased around $\hat{\beta}_{ols}/\beta_0$
- Variance depends on the details of sampling/rescaling
- This holds when higher-order terms in R_Ω are small—informally, when leverage-based sampling is used and rank is preserved.

We consider the empirical performance of several versions:

- UNIF: variance scales as $\frac{n}{r}$
- BLEV: variance scales as $\frac{p}{r}$, but have $\frac{1}{h_{ii}}$ terms in denominator of sandwich expression
- ALEV: faster but similar to or slightly better than BLEV
- SLEV: variance scales as $\frac{p}{r}$ but $\frac{1}{h_{ii}}$ terms in denominator are moderated since no probabilities are too small
- UNWL: $\frac{1}{h_{ii}}$ terms are *not* in denominator, but estimates unbiased around $\hat{\beta}_{wls}/\beta_0$.

BLEV and UNIF on data with different leverage scores

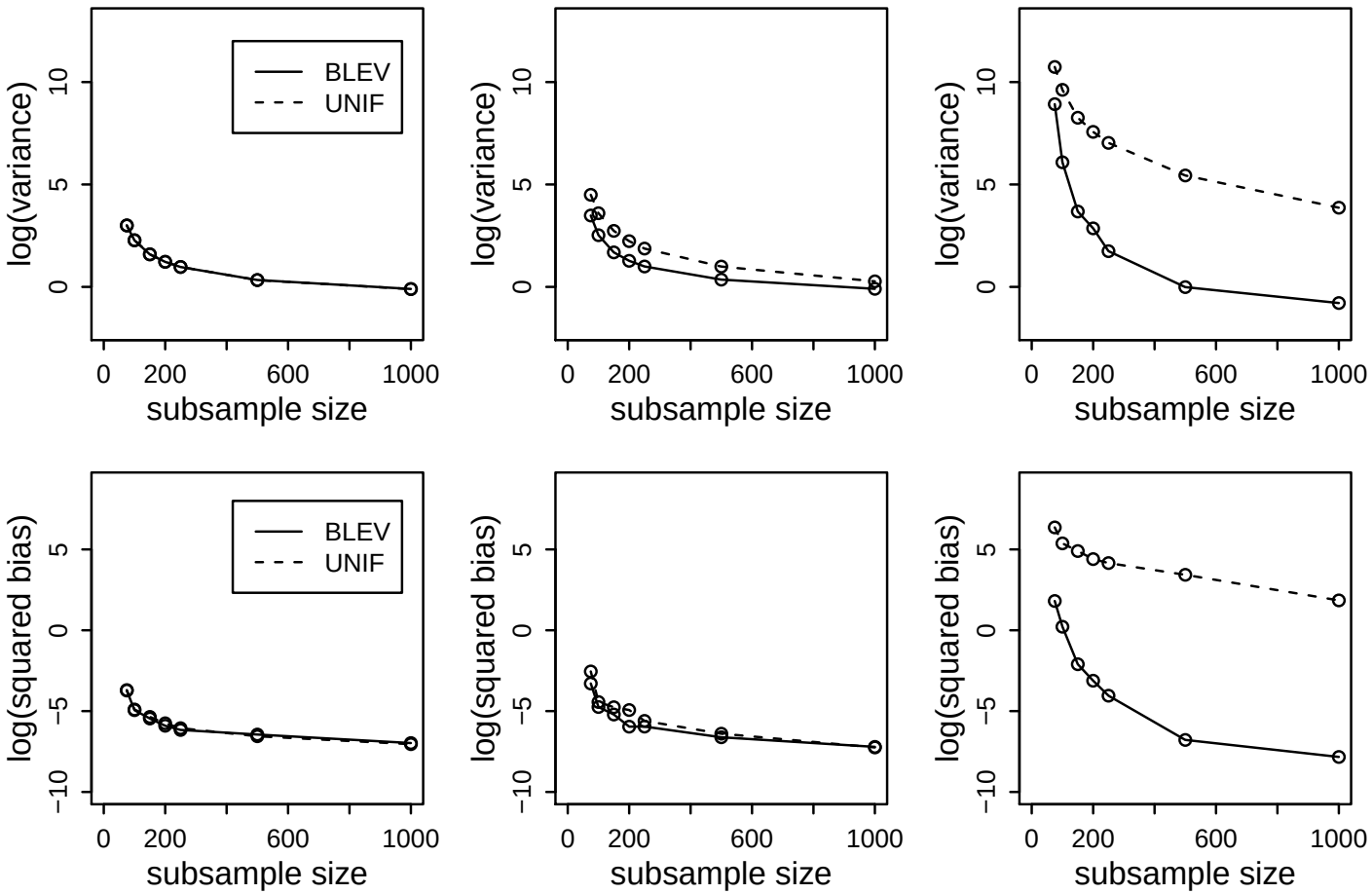


Figure: Empirical variances and squared biases of the BLEV and UNIF estimators in three data sets (left to right, Gaussian, multivariate-t with 3 d.o.f. (T3), and multivariate-t with 1 d.o.f. (T1)) for $n = 1000$ and $p = 50$. Black lines are BLEV; dash lines are UNIF.

BLEV and UNIF when rank is lost (1 of 2)

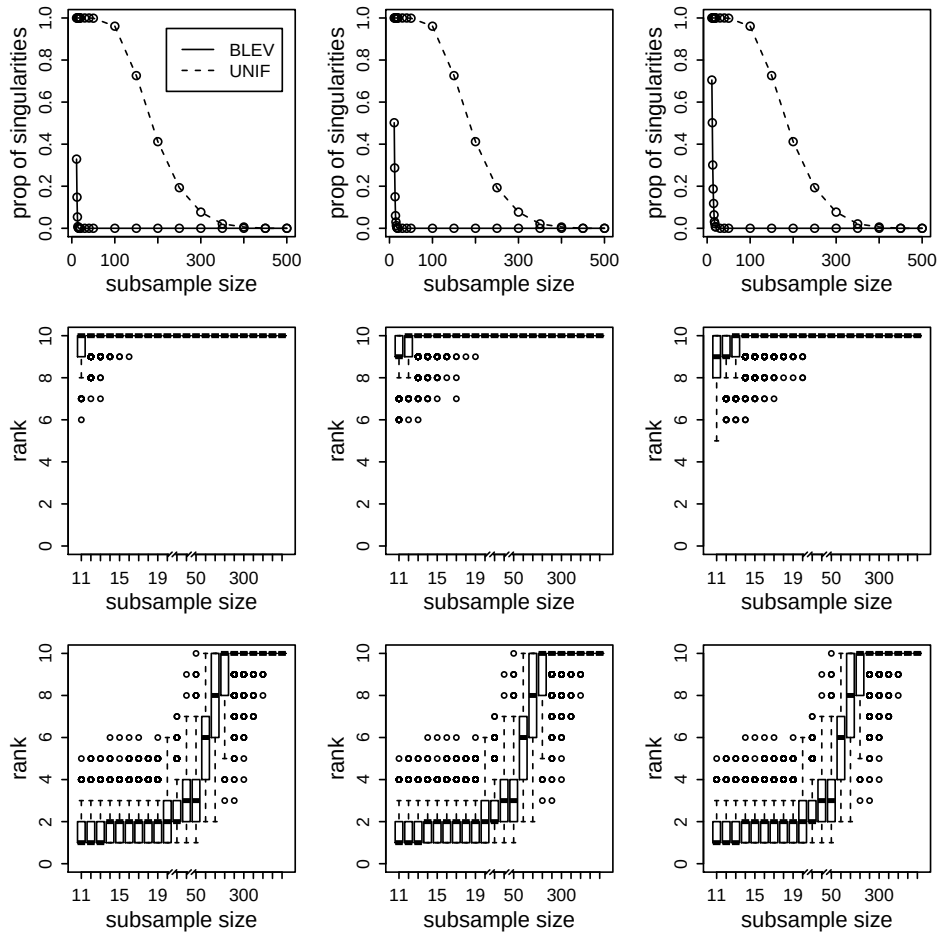


Figure: Comparison of BLEV and UNIF when rank is lost in the sampling process ($n = 1000$ and $p = 10$ here). Left panels: T3 data. Middle panels: T2 data. Right panels: T1 data. Upper panels: Proportion of singular $X^T WX$, out of 500 trials, for both BLEV (solid lines) and UNIF (dashed lines). Middle panels: Boxplots of ranks of 500 BLEV subsamples. Lower panels: Boxplots of ranks of 500 UNIF subsamples. Note the nonstandard scaling of the X axis.

BLEV and UNIF when rank is lost (2 of 2)

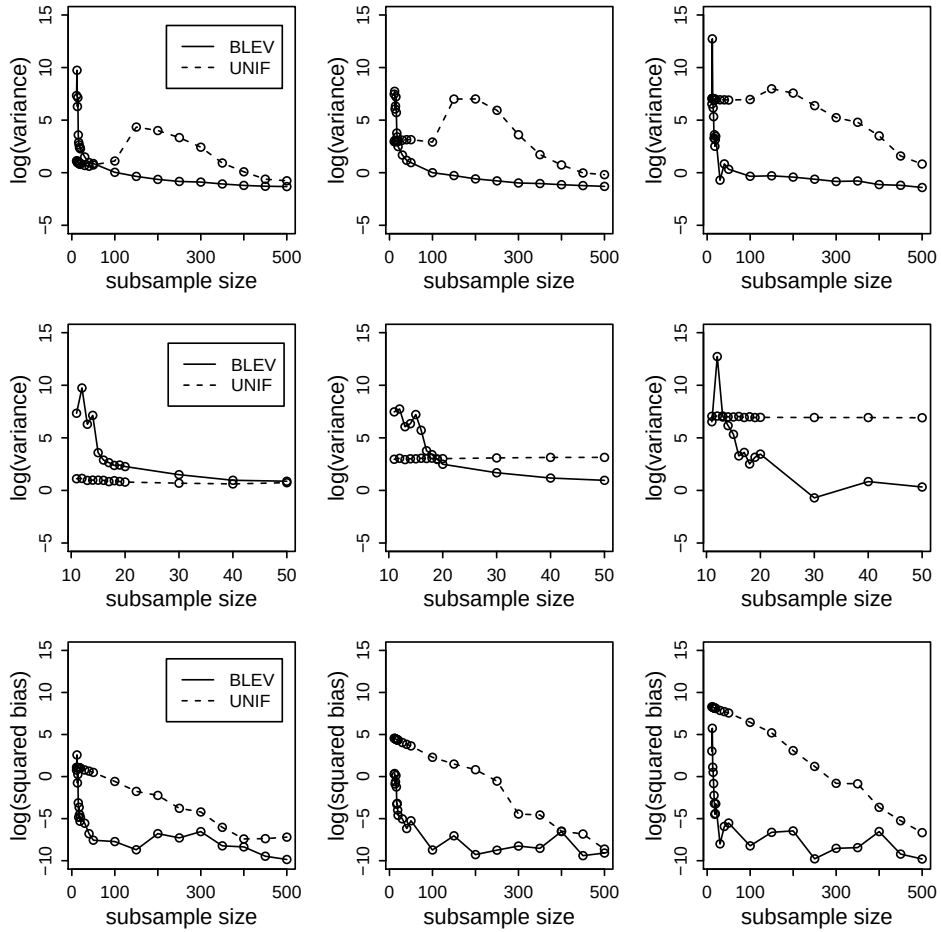


Figure: Comparison of BLEV and UNIF when rank is lost in the sampling process ($n = 1000$ and $p = 10$ here). Left panel: T3 data. Middle panels: T2 data. Right panels: T1 data. Upper panels: The logarithm of variances of the estimates. Middle panels: The logarithm of variances, zoomed-in on the X-axis. Lower panels: The logarithm of squared bias of the estimates.

Combining BLEV and UNIF into SLEV

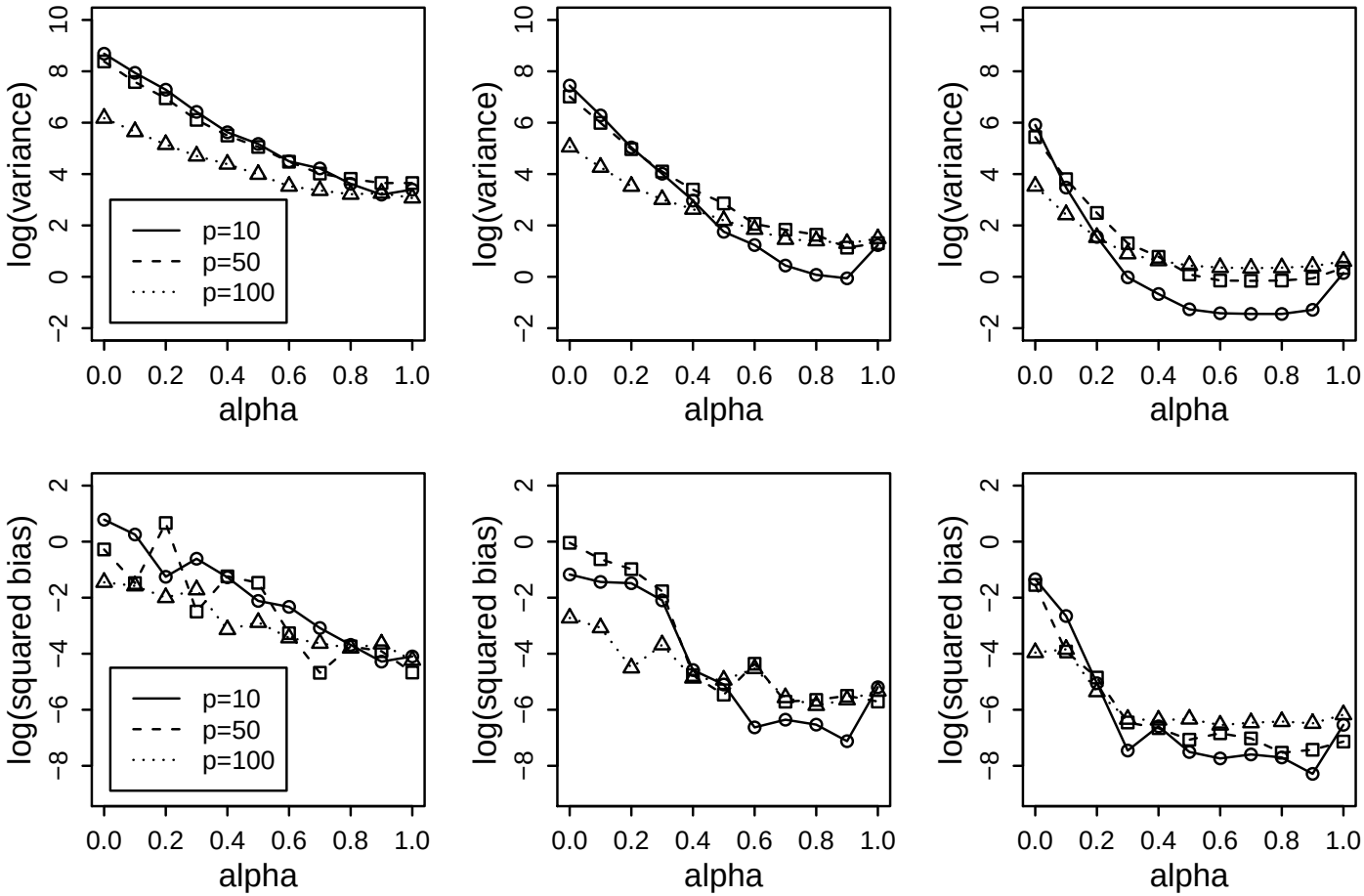


Figure: Empirical variances and squared biases (*unconditional*) of the SLEV estimator in data generated from T1 with $n = 1000$ and variable p. Circles connected by black lines are $p = 10$; squares connected by dash lines are $p = 50$; triangles connected by dotted lines are $p = 100$. Left panel: subsample size $r = 3p$. Middle panel: subsample size $r = 5p$. Right panel: subsample size $r = 10p$.

Outline

Tackling statistical properties of subsampling estimators

Challenges: $\tilde{\beta} = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{Y}$

- There are *two* parts of randomness involved: $\mathbf{Y}$ and $\mathbf{W}$.
- The random variable $\mathbf{W}$ enters the estimator in a *nonlinear* fashion.

- When direct study of some quantity has technical difficulties, one common practice in statistics is to use *asymptotic analysis*, e.g., we consider how the estimator behaves as $n \to \infty$.
- In asymptotic analysis, the intermediate we need to derive is the *asymptotic distribution* of estimator.

Asymptotic analysis in statistics

Example (Maximum likelihood estimator (MLE))

For generalized linear model,

$$\hat{\boldsymbol{\beta}}_{MLE} = \arg\max_{\boldsymbol{\beta}} \mathcal{L}(\theta) = \sum_{i=1}^{n} \{y_i u(\boldsymbol{x}_i^T \boldsymbol{\beta}) - b(u(\boldsymbol{x}_i^T \boldsymbol{\beta}))\},$$

where $u(\cdot)$ and $b(\cdot)$ are some distribution related functions.

- There exists even no explicit form for the MLE.

In statistics, the optimality of MLE is justified using asymptotic analysis. When $n \to \infty$, under mild regularity conditions,

- The variance of MLE achieves Cramér-Rao lower bound, which is a theoretical lower bound on the variance of unbiased estimators.
- In addition,

$$\sqrt{n}(\hat{\boldsymbol{\beta}}_{\text{mle}} - \boldsymbol{\beta}_0) \xrightarrow{d} \mathcal{N}(0,\ V^{-1}).$$

  This enables tasks such as hypothesis testing and confidence intervals.

MSE: recap

Let $\boldsymbol{T}_n$ be a $p \times 1$ estimator of a $p \times 1$ parameter $\boldsymbol{\nu}$, for every n.

In studying statistical properties, look directly at the random variable $(\boldsymbol{T}_n - \boldsymbol{\nu})$.

Characterize its bias $(\mathrm{E}(\boldsymbol{T}_n) - \boldsymbol{\nu})$ and variance $\mathrm{Var}(\boldsymbol{T}_n)$.

MSE

$$\begin{aligned} MSE(\boldsymbol{T}_n; \boldsymbol{\nu}) &= \mathrm{E}[(\boldsymbol{T}_n - \boldsymbol{\nu})^T (\boldsymbol{T}_n - \boldsymbol{\nu})] \\ &= \mathrm{tr}(\mathrm{Var}(\boldsymbol{T}_n)) + (\mathrm{E}(\boldsymbol{T}_n) - \boldsymbol{\nu})^T (\mathrm{E}(\boldsymbol{T}_n) - \boldsymbol{\nu}). \end{aligned}$$

AMSE: Basics in asymptotic analysis

We have no direct information in $(\boldsymbol{T}_n - \boldsymbol{\nu})$. From asymptotic analysis,

$$\boldsymbol{\Sigma}_n^{-1}(\boldsymbol{T}_n - \boldsymbol{\nu}) \xrightarrow{d} \boldsymbol{Z},$$

where $\boldsymbol{Z}$ is a $p \times 1$ random vector s.t. its i-th element Z_i satisfies $0 < \mathrm{E}(Z_i^2) < \infty$, $i = 1, \ldots, p$, and $\boldsymbol{\Sigma}_n$ is a sequence of $p \times p$ positive definite matrices.

Design AMSE using the variance and expectation of $\boldsymbol{Z}$.

AMSE

The AMSE of $\boldsymbol{T}_n$, denoted as $AMSE(\boldsymbol{T}_n; \boldsymbol{\nu})$, is defined as $\boldsymbol{T}_n - \boldsymbol{\nu}$

$$\begin{aligned} AMSE(\boldsymbol{T}_n; \boldsymbol{\nu}) &= \mathrm{E}(\boldsymbol{Z}^T \boldsymbol{\Sigma}_n \boldsymbol{Z}) = \mathrm{tr}(\boldsymbol{\Sigma}_n^{1/2} \mathrm{Var}(\boldsymbol{Z}) \boldsymbol{\Sigma}_n^{1/2}) + (\mathrm{E}(\boldsymbol{Z})^T \boldsymbol{\Sigma}_n \mathrm{E}(\boldsymbol{Z})) \\ &= \mathrm{tr}(\mathrm{AVar}(\boldsymbol{T}_n)) + (\mathrm{AE}(\boldsymbol{T}_n) - \boldsymbol{\nu})^T (\mathrm{AE}(\boldsymbol{T}_n) - \boldsymbol{\nu}), \end{aligned}$$

where $\mathrm{AVar}(\boldsymbol{T}_n) = \boldsymbol{\Sigma}_n^{1/2} \mathrm{Var}(\boldsymbol{Z}) \boldsymbol{\Sigma}_n^{1/2}$ and $\mathrm{AE}(\boldsymbol{T}_n) = \boldsymbol{\nu} + \boldsymbol{\Sigma}_n^{1/2} \mathrm{E}(\boldsymbol{Z})$ denote the asymptotic variance-covariance matrix and the asymptotic expectation of $\boldsymbol{T}_n$ in estimating $\boldsymbol{\nu}$, respectively.

Subsampling Estimators for Estimating the Parameter

(Zhang, Ma, Mahoney, and Yu 201X)

- Let $r = O(n^{1-\alpha})$, where $0 < \alpha < 1$; $\pi_{min} = O(n^{-\gamma_0})$, where $\gamma_0 \geq 1$.

Theorem (Asymptotic Normality of Subsampling Estimator)

Assume (A1). There exists positive constants b and B such that

$$b \leq \lambda_{min}(\mathbf{X}^T\mathbf{X}/n) \leq \lambda_{max}(\mathbf{X}^T\mathbf{X}/n) \leq B.$$

(A2). $\gamma_0 + \alpha < 2$.
As $n \to \infty$, we have

$$(\sigma^2\mathbf{\Sigma}_0)^{-\frac{1}{2}}(\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta}_0) \xrightarrow{d} \mathbf{N}(\mathbf{0}, \mathbf{I}_p).$$

where $\mathbf{\Sigma}_0 = (\mathbf{X}^T\mathbf{X})^{-1}[\mathbf{X}^T(\mathbf{I} + \mathbf{\Omega})\mathbf{X}](\mathbf{X}^T\mathbf{X})^{-1}$, and $\mathbf{\Omega} = diag\{1/r\pi_i\}_{i=1}^n$, $\mathbf{I}_p$ denotes a $p \times p$ identity matrix.

- Taylor expansion for nonlinear complication.
- Central Limit Theorem for multinomial sums.

Subsampling Estimators for Estimating the Parameter

(Zhang, Ma, Mahoney, and Yu 201X)

Theorem (Asymptotic Normality of Subsampling Estimator-cont'd)

Thus, in unconditional inference, $\tilde{\boldsymbol{\beta}}$ is an asymptotically unbiased estimator of $\boldsymbol{\beta}_0$, i.e.

$$AE(\tilde{\boldsymbol{\beta}}) = \boldsymbol{\beta}_0,$$

and the asymptotic variance-covariance matrix of $\tilde{\boldsymbol{\beta}}$ is

$$AVar(\tilde{\boldsymbol{\beta}}) = \sigma^2 \boldsymbol{\Sigma}_0.$$

Extensions to slowly diverging number of predictors, conditional inference, etc.

Minimum AMSE subsampling estimator

(Zhang, Ma, Mahoney, and Yu 201X)

Estimating β_0

The subsampling estimator with the subsampling probabilities

$$\pi_i = \frac{\|(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i\|}{\sum_{i=1}^n \|(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i\|}, i = 1, \ldots, n,$$

has the smallest $AMSE(\tilde{\beta}; \beta_0)$.

Estimating $\mathbf{Y} = \mathbf{X}\beta_0$

The subsampling estimator with the subsampling probabilities

$$\pi_i = \frac{\|\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i\|}{\sum_{i=1}^n \|\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i\|} = \frac{\sqrt{h_{ii}}}{\sum_{i=1}^n \sqrt{h_{ii}}}, i = 1, \ldots, n,$$

has the smallest $AMSE(\mathbf{X}\tilde{\beta}; \mathbf{X}\beta_0)$.

Estimating $\mathbf{X}^T\mathbf{X}\beta_0$

The subsampling estimator with the subsampling probabilities

$$\pi_i = \frac{\|\mathbf{x}_i\|}{\sum_{i=1}^n \|\mathbf{x}_i\|}, i = 1, \ldots, n,$$

has the smallest $AMSE(\mathbf{X}^T\mathbf{X}\tilde{\beta}; \mathbf{X}^T\mathbf{X}\beta_0)$.

Outline

Towards structural results for statistical objectives

Recall the original LS/OLS problem:

$$\beta_{OLS} = \arg\min_{\beta \in \mathbb{R}^p} \|Y - X\beta\|_2^2,$$

where $X \in \mathbb{R}^{n \times p}$. Assume $n \gg p$ and $\text{rank}(X) = p$. The LS solution is:

$$\beta_{OLS} = (X^T X)^{-1} X^T Y = X^\dagger Y.$$

Given the full data (X, Y), generate "sketched data" (SX, SY) where $S \in \mathbb{R}^{r \times n}$, with $r \ll n$, is an *arbitrary* (sketching) matrix "sketching matrix," and compute:

$$\beta_S \in \arg\min_{\beta \in \mathbb{R}^p} \|SY - SX\beta\|_2^2.$$

The LS/OLS solution** on the sketch (SX, SY) is:

$$\beta_S = (SX)^\dagger SY.$$

**This does *not* in general equal $((SX)^T SX)^{-1}(SX)^T SY$—why?

The statistical approach

(Raskutti and Mahoney, 2014)

Let $\beta \in \mathbb{R}^p$ is the "true" parameter , and assume a standard linear "model" on Y,

$$Y = X\beta + \epsilon,$$

where $\epsilon \in \mathbb{R}^n$ is a standardized noise vector, with $\mathbb{E}[\epsilon] = 0$ and $\mathbb{E}[\epsilon\epsilon^T] = I_{n\times n}$, where the expectation $\mathbb{E}[\cdot]$ is taken over the random noise ϵ.

- Relative *statistical prediction efficiency* (SPE), definedas follows:

$$C_{SPE}(S) = \frac{\mathbb{E}[\|X(\beta - \beta_S)\|_2^2]}{\mathbb{E}[\|X(\beta - \beta_{OLS})\|_2^2]}.$$

- Relative *statistical residual efficiency* (SRE), defined as follows:

$$C_{SRE}(S) = \frac{\mathbb{E}[\|Y - X\beta_S\|_2^2]}{\mathbb{E}[\|Y - X\beta_{OLS}\|_2^2]}.$$

A statistical perspective on the algorithmic approach

(Raskutti and Mahoney, 2014)

Consider "defining" Y in terms of X by the following "linear model":

$$Y = X\beta + \epsilon,$$

where

- $\beta \in \mathbb{R}^p$ is arbitrary "true parameter"
- $\epsilon \in \mathbb{R}^n$ is *any* vector that lies in the null-space of X^T

Consider the worst-case (due to supremum) criterion (analyzed in TCS).

- The *worst-case error* (WCE) is defined as follows:

$$\begin{aligned} C_{WCE}(S) &= \sup_Y \frac{\|Y - X\beta_S\|_2^2}{\|Y - X\beta_{OLS}\|_2^2} \\ &= \sup_{Y = X\beta + \epsilon,\ X^T\epsilon = 0} \frac{\|Y - X\beta_S\|_2^2}{\|Y - X\beta_{OLS}\|_2^2}. \end{aligned}$$

(I.e., supremum over ϵ, s.t. $X^T\epsilon = 0$, and not expectation over ϵ, s.t. $\mathbb{E}[\epsilon] = 0$.)

Comments on this approach

(Raskutti and Mahoney, 2014)

- $\beta_{OLS} = \beta + (X^T X)^{-1} X^T \epsilon$ for both "linear models," but
  - Statistical setting: β_{OLS} is a random variable (with $\mathbb{E}[\epsilon\epsilon^T] = I_{n \times n}$).
    - ★ $\mathbb{E}[\beta_{OLS}] = \beta$ and $\mathbb{E}[(\beta - \beta_{OLS})(\beta - \beta_{OLS})^T] = (X^T X)^{-1}$
  - Algorithmic setting: β_{OLS} is a deterministic.
    - ★ $\beta_{OLS} = \beta$ (since $X^T \epsilon = 0$).
- $C_{WCE}(S)$ is the worst-case algorithmic analogue of $C_{SRE}(S)$.
- The worst-case algorithmic analogue of $C_{SPE}(S)$ would be:

$$\sup_Y \frac{\|X(\beta - \beta_S)\|_2^2}{\|X(\beta - \beta_{OLS})\|_2^2},$$

  except that the denominator equals zero.
- Statistical subtleties: sketching matrices that are independent of both X and Y (e.g., uniform sampling) or depend only on X (e.g., leverage scores of X) or depend on X and Y (e.g., influence scores of (X, Y)).

Key structural lemma

(Raskutti and Mahoney, 2014)

Characterize how $C_{WCE}(S)$, $C_{SPE}(S)$, and $C_{SRE}(S)$ depend on different structural properties of SU and the *oblique projection* matrix $\Pi_S^U := U(SU)^\dagger S$.

Lemma (RM14)

For the algorithmic setting,

- $C_{WCE}(S) = 1 + \sup_{\delta \in \mathbb{R}^p, U^T\epsilon = 0} \left[\frac{\|(I_{p\times p} - (SU)^\dagger (SU))\delta\|_2^2}{\|\epsilon\|_2^2} + \frac{\|\Pi_S^U \epsilon\|_2^2}{\|\epsilon\|_2^2} \right].$

For the statistical setting,

- $C_{SPE}(S) = \frac{\|(I_{p\times p} - (SU)^\dagger SU)\Sigma V^T \beta\|_2^2}{p} + \frac{\|\Pi_S^U\|_F^2}{p}$
- $C_{SRE}(S) = 1 + \frac{\|(I_{p\times p} - (SU)^\dagger SU)\Sigma V^T \beta\|_2^2}{n-p} + \frac{\|\Pi_S^U\|_F^2 - p}{n-p} = 1 + \frac{C_{SPE}(S)-1}{n/p-1}$

Corollary of key structural lemma

(Raskutti and Mahoney, 2014)

Let $\alpha(S) > 0$, $\beta(S) > 0$, and $\gamma(S) > 0$ be such that

- $\tilde{\sigma}_{\min}(SU) \geq \alpha(S)$
- $\sup_{\epsilon,\, U^T\epsilon=0} \frac{\|U^T S^T S\epsilon\|_2}{\|\epsilon\|_2} \leq \beta(S)$
- $\|U^T S^T S\|_F \leq \gamma(S)$

Lemma (RM14)

- $C_{WCE}(S) \leq 1 + \sup_{\delta\in\mathbb{R}^p, U^T\epsilon=0} \frac{\|(I_{p\times p} - (SU)^\dagger (SU))\delta\|_2^2}{\|\epsilon\|_2^2} + \frac{\beta^2(S)}{\alpha^4(S)}$
- $C_{SPE}(S) \leq \frac{\|(I_{p\times p} - (SU)^\dagger SU)\Sigma V^T \beta\|_2^2}{p} + \frac{\gamma^2(S)}{\alpha^4(S)}$
- $C_{SRE}(S) \leq 1 + \frac{p}{n}\left[\frac{\|(I_{p\times p} - (SU)^\dagger SU)\Sigma V^T \beta\|_2^2}{p} + \frac{\gamma^2(S)}{\alpha^4(S)}\right].$

A statistical perspective on randomized sketching (1 of 2)

(Raskutti and Mahoney, 2014)

Things to note.

- Different properties of Π_S^U are needed.
  - Algorithmic setting: $\sup_{\epsilon \in \mathbb{R}^n / \{0\}, \Pi^U \epsilon = 0} \frac{\|\Pi_S^U \epsilon\|_2^2}{\|\epsilon\|_2^2}$
    - ★ Largest eigenvalue of Π_S^U, i.e., Spectral norm, enters to control the worst direction in the null-space of U^T.
  - Statistical setting: $\|\Pi_S^U\|_F^2$
    - ★ ℓ_2 norm of the eigenvalues of Π_S^U, i.e., Frobenius norm, enters to control an average over homoscedastic noise.
- The $(SU)^\dagger SU$ term is a "bias" term that is non-zero if $\text{rank}(SU) < p$.
  - Often introducing a small bias is a very good thing.
- Need many more samples r to obtain bounds on $C_{SPE}(S)$ than $C_{SRE}(S)$
  - since $C_{SRE}(S) = 1 + \frac{C_{SPE}(S)-1}{n/p-1}$ and so re-scales $C_{SPE}(S)$ by $p/n \ll 1$

A statistical perspective on randomized sketching (2 of 2)

(Raskutti and Mahoney, 2014)

Main theoretical conclusions.

- $C_{SRE}(S)$ can be well-bounded for $p \lesssim r \ll n$, for typical sampling/projection matrices S (consistent with previous results on $C_{WCE}(S)$).
- $C_{SPE}(S)$ typically requires the sample size $r \gtrsim \Omega(n)$ (consistent with the use of sampling in bootstrap).

Main empirical conclusions.

- Short answer: empirical results consistent with theory.
- Medium answer:
  - Getting good statistical results with RandNLA algorithms can be "easier" or "harder" than getting good algorithmic results.
  - Must control other structures: small leverage scores, non-spectral norms, etc.
  - Tradeoffs are very different than arise in TCS, NLA, ML, etc.
- Long answer: more work needed ...

Sketched ridge regression

(Wang, Gittens, and Mahoney (2017))

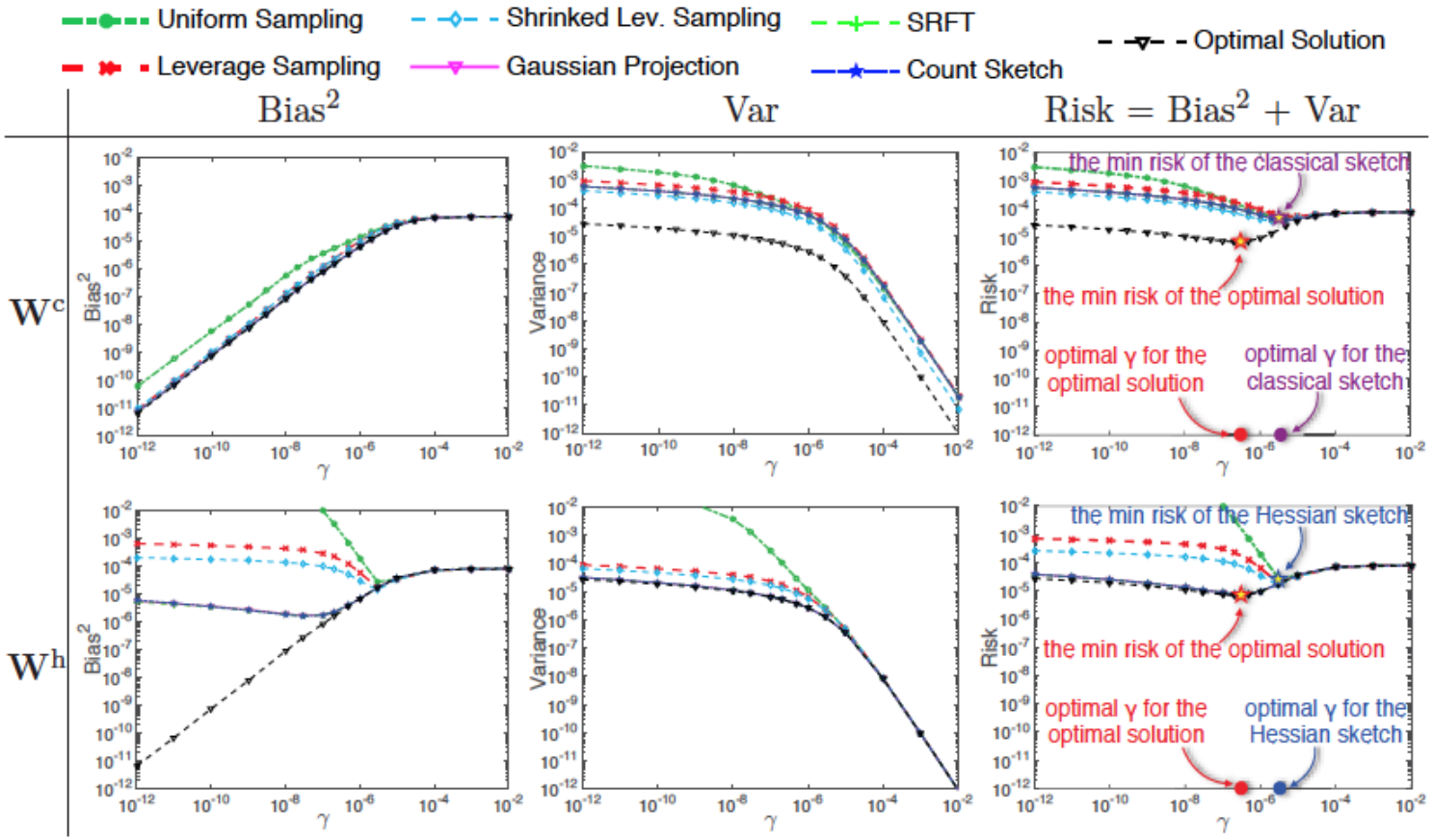


Figure 3: An empirical study of classical sketch and Hessian sketch from the statistical perspective. The x-axis is the regularization parameter γ (log-scale); the y-axes are respectively bias^2, variance, and risk (log-scale). We indicate the minimum risks and optimal choice of γ in the plots.

Outline

Review of randomized LS

Lopes, Wang, and Mahoney, 2018

Consider a deterministic matrix $A \in \mathbb{R}^{n \times d}$ and vector $b \in \mathbb{R}^n$, with $n \gg d$.

The exact solution $x_{\text{opt}} := \underset{x \in \mathbb{R}^d}{\text{argmin}} \, \|Ax - b\|_2$ is too costly to compute.

We reduce problem with a random sketching matrix $S \in \mathbb{R}^{t \times n}$ with $d \ll t \ll n$.
Define $\tilde{A} := SA$ and $\tilde{b} := Sb$.

We focus on two particular randomized LS algorithms:

---

1. **Classic Sketch (CS).** (Drineas et al, 2006)

$$\tilde{x} := \underset{x \in \mathbb{R}^d}{\text{argmin}} \, \left\|\tilde{A}x - \tilde{b}\right\|_2$$

---

2. **Iterative Hessian Sketch (IHS).** (Pilanci & Wainwright 2016)

$$\hat{x}_{i+1} := \underset{x \in \mathbb{R}^d}{\text{argmin}} \left\{ \tfrac{1}{2}\|\tilde{A}(x - \hat{x}_i)\|_2^2 + \langle A^\top (A\hat{x}_i - b) \, , \, x \rangle \right\}, \qquad i = 1, \ldots, k.$$

Problem formulation (error estimation)

Lopes, Wang, and Mahoney, 2018

We will estimate the errors of the random solutions $\tilde{x}$ and $\hat{x}_k$ in terms of high-probability bounds.

Let $\|\cdot\|$ denote any norm on $\mathbb{R}^d$, and let $\alpha \in (0,1)$ be fixed.

Goal: Compute numerical estimates $\tilde{q}(\alpha)$ and $\hat{q}_k(\alpha)$, such that the bounds

$$\|\tilde{x} - x_{\text{opt}}\| \leq \tilde{q}(\alpha)$$

$$\|\hat{x}_k - x_{\text{opt}}\| \leq \hat{q}_k(\alpha)$$

each hold with probability at least $1 - \alpha$.

Intuition for the bootstrap

Lopes, Wang, and Mahoney, 2018

Key idea: Artificially generate a bootstrapped solution $\tilde{x}^*$ such that the fluctuations of $\tilde{x}^* - \tilde{x}$ are statistically similar to the fluctuations of $\tilde{x} - x_{\text{opt}}$.

In the "bootstrap world", $\tilde{x}$ plays the role of x_{opt}, and $\tilde{x}^*$ plays the role of $\tilde{x}$.

The bootstrap sample $\tilde{x}^*$ is the LS solution obtained by "perturbing" $\tilde{A}$ and $\tilde{b}$.

(The same intuition also applies to the IHS solution $\hat{x}_k$.)

Algorithm (Error estimate for Classic Sketch)

Lopes, Wang, and Mahoney, 2018

Input: A positive integer B, and the sketches $\tilde{A}$, $\tilde{b}$, and $\tilde{x}$.

For: $l = 1, \ldots, B$ **do**

- Draw a random vector $\mathbf{i} := (i_1, \ldots, i_t)$ by sampling m numbers with replacement from $\{1, \ldots, t\}$.
- Form the matrix $\tilde{A}^* := \tilde{A}(\mathbf{i}, :)$, and vector $\tilde{b}^* := \tilde{b}(\mathbf{i})$.
- Compute the vector

$$\tilde{x}^* := \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \|\tilde{A}^* x - \tilde{b}^*\|_2,$$

  and the scalar $\varepsilon_l^* := \|\tilde{x}^* - \tilde{x}\|$.

Return: $\tilde{q}(\alpha) := \text{quantile}(\varepsilon_1^*, \ldots, \varepsilon_B^*; 1 - \alpha)$.

Note: A similar algorithm works for IHS.

Computational cost

Lopes, Wang, and Mahoney, 2018

1. Cost of error estimation is independent of large dimension n, whereas most randomized LS algorithms scale linearly in n.

2. In practice, as few as $B = 20$ bootstrap samples are sufficient.

3. Implementation is embarrassingly parallel.
(Per-processor cost is $\mathcal{O}(td^2)$, with modest communication.)

4. Bootstrap computations have free warm starts.

5. Error estimates can be extrapolated (similar to MM context).

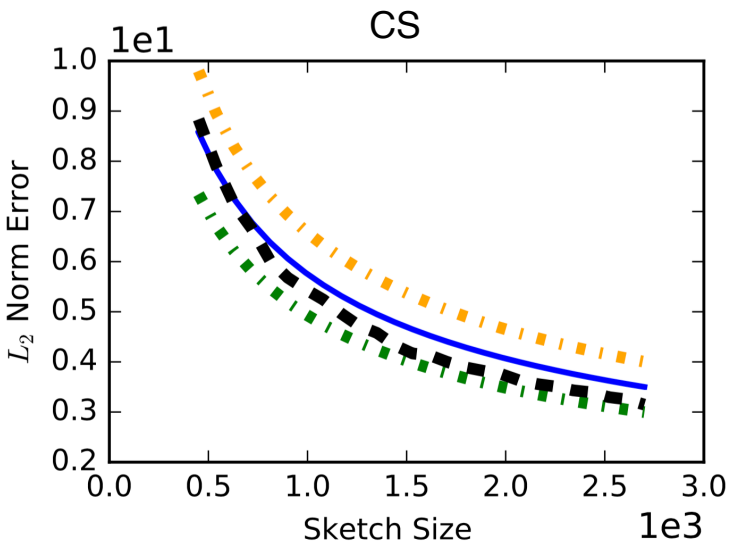
Theoretical and empirical performance

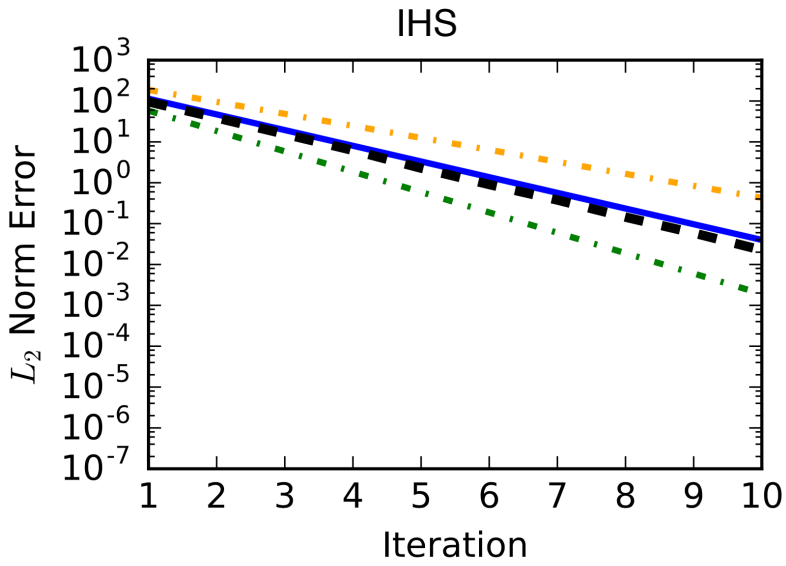
Lopes, Wang, and Mahoney, 2018

Theory: Guarantees are available for both CS and IHS (cf. arXiv paper).

Experiment: 'YearPredictionMSD' data from LIBSVM: $n \sim 4.6 \times 10^5$ and $d = 90$

- **CS:** fix initial sketch size $t_0 = 5d$ and extrapolate on $t \gg t_0$
- **IHS:** fix sketch size $t = 10d$ and extrapolate on number of iterations
- bootstrap samples $B = 20$





Summary of connection with Bootstrapping

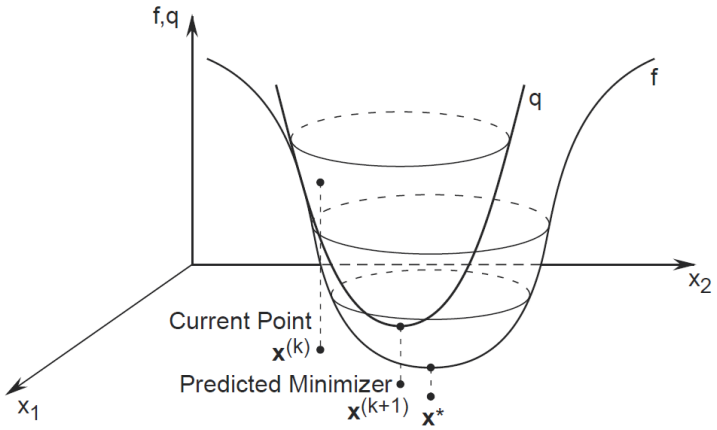
Lopes, Wang, and Mahoney, 2018

- Bootstrapping is a flexible approach to error estimation that can be adapted to a variety of RandNLA algorithms.
- This provides a practical alternative to worst-case error bounds, and adapts to the input at hand.
- The cost of bootstrapping does not outweigh the benefits of sketching.
- The bootstrap computations are highly scalable – since they do not depend on large dimension n, are easily parallelized, and can be extrapolated.
- Numerical performance is encouraging, and is supported by theoretical guarantees.

Outline

Optimization Overview

Consider optimizing $F : \mathbb{R}^d \to \mathbb{R}$:



$$x^{(k+1)} = \arg\min_{\mathbf{x}\in\mathcal{D}\cap\mathcal{X}} \left\{ F(\mathbf{x}^{(k)}) + (\mathbf{x}-\mathbf{x}^{(k)})^T \mathbf{g}(\mathbf{x}^{(k)}) + \frac{1}{2\alpha_k}(\mathbf{x}-\mathbf{x}^{(k)})^T H(\mathbf{x}^{(k)})(\mathbf{x}-\mathbf{x}^{(k)}) \right\}$$

Iterative optimization:

$$\text{First-order methods:} \quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla F(\mathbf{x}^{(k)})$$

$$\text{Second-order methods:} \quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - [\nabla^2 F(\mathbf{x}^{(k)})]^{-1} \nabla F(\mathbf{x}^{(k)})$$

Outline

RLA and SGD

- **SGD** (Stochastic Gradient Descent) methods[††]
  - Widely used in practice because of their scalability, efficiency, and ease of implementation.
  - Work for problems with general convex (or not) objective function.
  - Only provide an asymptotic bounds on convergence rate.
  - Typically formulated in terms of differentiability assumptions, smoothness assumptions, etc.
- **RLA** (Randomized Linear Algebra) methods[‡‡]
  - Better worst-case theoretical guarantees and better control over solution precision.
  - Less flexible (thus far), e.g., in the presence of constraints.
  - E.g., may use interior point method for solving constrained subproblem, and this may be less efficient than SGD.
  - Typically formulated (either TCS-style or NLA-style) for worst-case inputs.

[††] SGD: iteratively solve the problem by approximating the true gradient by the gradient at a single example.

[‡‡] RLA: construct (with sampling/projections) a random sketch, and use that sketch to solve the subproblem or construct preconditioners for the original problem.

Can we get the "best of both worlds"?

Consider problems where both methods have something nontrivial to say.

Definition

Given a matrix $A \in \mathbb{R}^{n \times d}$, where $n \gg d$, a vector $b \in \mathbb{R}^n$, and a number $p \in [1, \infty]$, the *overdetermined ℓ_p regression problem* is

$$\min_{x \in \mathcal{Z}} f(x) = \|Ax - b\|_p.$$

Important special cases:

- Least Squares: $\mathcal{Z} = \mathbb{R}^d$ and $p = 2$.
  - Solved by eigenvector methods with $\mathcal{O}(nd^2)$ worst-case running time; or by iterative methods with running time depending on $\kappa(A)$.
- Least Absolute Deviations: $\mathcal{Z} = \mathbb{R}^d$ and $p = 1$.
  - Unconstrained ℓ_1 regression problem can be formulated as a linear program and solved by an interior-point method.

Deterministic ℓ_p regression as stochastic optimization

- Let $U \in \mathbb{R}^{n \times (d+1)}$ be a basis of the range space of $\begin{pmatrix} A & b \end{pmatrix}$ in the form of

$$U = \begin{pmatrix} A & b \end{pmatrix} F,$$

where $F \in \mathbb{R}^{(d+1)\times(d+1)}$.

- The constrained overdetermined (deterministic) ℓ_p regression problem is equivalent to the (stochastic) optimization problem

$$\begin{aligned} \min_{x \in \mathcal{Z}} \|Ax - b\|_p^p &= \min_{y \in \mathcal{Y}} \|Uy\|_p^p \\ &= \min_{y \in \mathcal{Y}} \mathbb{E}_{\xi \sim P} \left[H(y, \xi) \right], \end{aligned}$$

where $H(y, \xi) = \frac{|U_\xi y|^p}{p_\xi}$ is the randomized integrand and ξ is a random variable over $\{1, \ldots, n\}$ with distribution $P = \{p_i\}_{i=1}^n$.

- The constraint set of y is given by

$$\mathcal{Y} = \{y \in \mathbb{R}^k | y = F^{-1} v, v \in \mathcal{C}\},$$

where $\mathcal{C} = \{v \in \mathbb{R}^{d+1} | v_{1:d} \in \mathcal{Z}, v_{d+1} = -1\}$.

Brief overview of stochastic optimization

The *standard stochastic optimization problem* is of the form

$$\min_{x \in \mathcal{X}} f(x) = \mathbb{E}_{\xi \sim P} \left[F(x, \xi) \right], \tag{3}$$

where ξ is a random data point with underlying distribution P.

Two computational approaches for solving stochastic optimization problems of the form (3) based on Monte Carlo sampling techniques:

- **SA** (Stochastic Approximation):
  - start with an initial x_0, and solve (3) iteratively. In each iteration, a new sample point ξ_t is drawn from distribution P and the current weight is updated by its information (e.g., (sub)gradient of $F(x, \xi_t)$).
- **SAA** (Sampling Average Approximation):
  - sample n points from distribution P independently, $\xi_1, \ldots, \xi_n$, and solve the following "Empirical Risk Minimization" problem,

$$\min_{x \in \mathcal{X}} \hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} F(x, \xi_i).$$

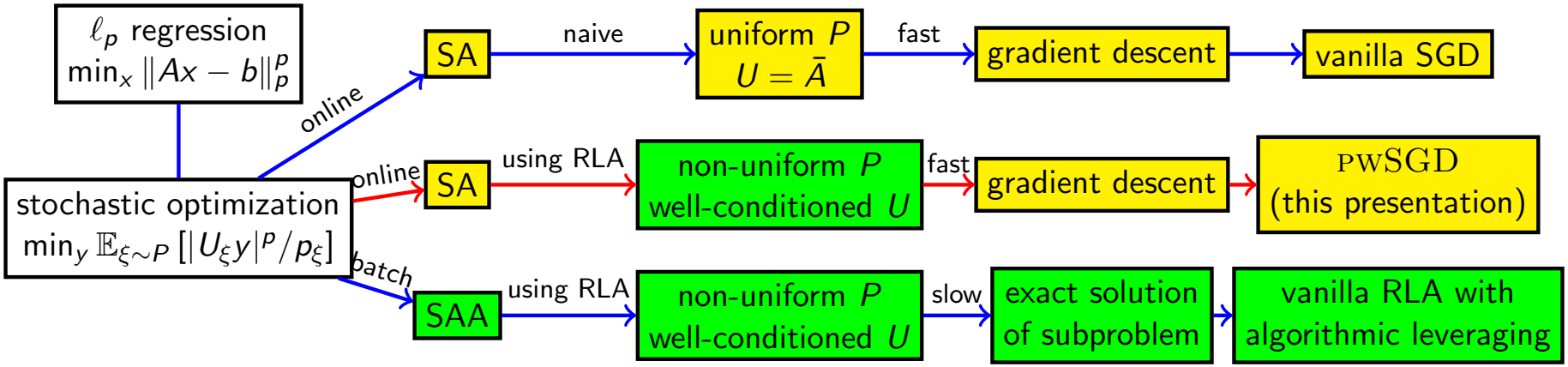
Solving ℓ_p regression via stochastic optimization

To solve this stochastic optimization problem, typically one needs to answer the following three questions.

- ($C1$): How to sample: SAA (i.e., draw samples in a batch mode and deal with the subproblem) or SA (i.e., draw a mini-batch of samples in an online fashion and update the weight after extracting useful information)?
- ($C2$): Which probability distribution P (uniform distribution or not) and which basis U (preconditioning or not) to use?
- ($C3$): Which solver to use (e.g., how to solve the subproblem in SAA or how to update the weight in SA)?

A unified framework for RLA and SGD

("Weighted SGD for Lp Regression with Randomized Preconditioning," Yang, Chow, Re, and Mahoney, 2015.)



Main relationships:

- SA + "naive" P and U: **vanilla SGD** whose convergence rate depends (without additional niceness assumptions) on n
- SA + "smart" P and U: **pwSGD**
- SAA + "naive" P: **uniform sampling RLA algorithm** which may fail if some rows are extremely important (not shown)
- SAA + "smart" P: **RLA (with algorithmic leveraging or random projections)** which has strong worst-case theoretical guarantee and high-quality numerical implementations

A combined algorithm: pwSGD

("Weighted SGD for Lp Regression with Randomized Preconditioning," Yang, Chow, Re, and Mahoney, 2015.)

pwSGD: Preconditioned weighted SGD consists of two main steps:

1. Apply RLA techniques for preconditioning and construct an importance sampling distribution.
2. Apply an SGD-like iterative phase with weighted sampling on the preconditioned system.

pwSGD has the following properties:

- After "batch" preconditioning (on arbitrary input), unlike vanilla SGD, the convergence rate of the SGD phase only depends on the low dimension d, i.e., it is independent of the high dimension n.
- With proper preconditioner, pwSGD runs in $\mathcal{O}(\log n \cdot \text{nnz}(A) + \text{poly}(d)/\epsilon^2)$ time (for arbitrary input) to return an approximate solution with ϵ relative error in terms of the objective.
- Empirically, pwSGD performs favorably compared to other competing methods, as it converges to a medium-precision solution, e.g., with ϵ roughly 10^{-2} or 10^{-3}, much more quickly.

Question: Connecting SAA with TCS coresets and RLA?

Can we use stochastic optimization ideas to combine RLA and SGD for other optimization/regression problems?

- To do so, we need to define "leverage scores" or "outlier scores" for them, since these scores play a crucial role in this stochastic framework.
- In [Feldman and Langberg, 2011] (in TCS), a framework for computing a "coreset" of $\mathcal{F}$ to a given optimization problem of the form:

$$\text{cost}(\mathcal{F}, x) = \min_{x \in \mathcal{X}} \sum_{f \in \mathcal{F}} f(x),$$

  where $\mathcal{F}$ is a set of function from a set $\mathcal{X}$ to $[0, \infty)$.
- The ℓ_p regression problem can be written as

$$\min_{x \in \mathcal{C}} \sum_{i=1}^{n} f_i(x),$$

  where $f_i(x) = |\bar{A}_i x|^p$, in which case $\mathcal{F} = \{f_i\}_{i=1}^n$.

Algorithm for computing a coreset

Sensitivities

Given a set of function $\mathcal{F} = \{f\}$,

- the *sensitivity* $m(f)$ *of* f is $m(f) = \lfloor \sup_{x \in \mathcal{X}} n \cdot \frac{f(x)}{\text{cost}(\mathcal{F},x)} \rfloor + 1$, and
- and the *total sensitivity* $M(\mathcal{F})$ *of* $\mathcal{F}$ is $M(\mathcal{F}) = \sum_{f \in \mathcal{F}} m(f)$.

1. Initialize $\mathcal{D}$ as an empty set.
2. Compute the sensitivity $m(f)$ for each function $f \in \mathcal{F}$.
3. $M(\mathcal{F}) \leftarrow \sum_{f \in \mathcal{F}} m(f)$.
4. For $f \in \mathcal{F}$
   Compute probabilities
   $$p(f) = \frac{m(f)}{M(\mathcal{F})}.$$
5. For $i = 1, \ldots, s$
   Pick f from $\mathcal{F}$ with probability $p(f)$.
   Add $f/(s \cdot p(f))$ to $\mathcal{D}$.
6. Return $\mathcal{D}$.

Theoretical guarantee

Dimension of subspaces

The dimension of $\mathcal{F}$ is defined as the smallest integer d, such that for any $G \subset \mathcal{F}$,

$$|\{\textbf{Range}(G, x, r) | x \in \mathcal{X}, r \geq 0\}| \leq |G|^d,$$

where $\textbf{Range}(G, x, r) = \{g \in G | g(x) \leq r\}$.

Theorem

Given a set of functions $\mathcal{F} : \mathcal{X} \to [0, \infty]$, *if* $s \geq \frac{cM(\mathcal{F})}{\epsilon^2}(\dim(\mathcal{F}') + \log\left(\frac{1}{\delta}\right))$, *then with probability at least* $1 - \delta$,

$$(1 - \epsilon) \sum_{f \in \mathcal{F}} f(x) \leq \sum_{f \in \mathcal{D}} f(x) \leq (1 + \epsilon) \sum_{f \in \mathcal{F}} f(x).$$

That is, the coreset method returns ϵ*-coreset for* $\mathcal{F}$.

Connection with RLA methods

("Weighted SGD for Lp Regression with Randomized Preconditioning," Yang, Chow, Re, and Mahoney, 2015.)

Fact. Coreset methods coincides the *RLA algorithmic leveraging* approach on LA problems; sampling complexities are the same up to constants!
Apply this to ℓ_p regression, with matrix $\bar{A} \in \mathbb{R}^{n \times (d+1)}$.

- Let $f_i(x) = |\bar{A}_i x|^p$, for $i \in [n]$. If λ_i be the i-th leverage score of $\bar{A}$, then

$$m(f_i) \leq n\beta^p \lambda_i + 1,$$

  for $i \in [n]$, and

$$M(\mathcal{F}) \leq n((\alpha\beta)^p + 1).$$

- Let $\mathcal{A} = \{|a^T x|^p | a \in \mathbb{R}^d\}$. We have

$$\dim(\mathcal{A}) \leq d + 1.$$

Fact. More generally, coreset methods work for *any* convex loss function.

- But they are *not* necessarily small (they depend on the total sensitivity)
- For other function classes, e.g., hinge loss, the size of the coreset $\sim 2^d$.
  - Define $f_i(x) = f(x, a_i) = (x^T a_i)^+$,where $x, a_i \in \mathbb{R}^d$ for $i \in [n]$.
    Then $\exists$ a set of vectors $\{a_i\}_{i=1}^d$ such that $M(\mathcal{F})$ of $\mathcal{F} = \{f_i\}_{i=1}^n$ is $\sim 2^d$.

Outline

Sub-sampled second-order optimization

Roosta-Khorasani and Mahoney "SSN I & II" 2016; Xu et al. "Inexact" 2017; and Yao et al. "Inexact" 2018 (nonconvex)

Consider optimizing $F : \mathbb{R}^d \to \mathbb{R}$:

$$\min_{\mathbf{x} \in \mathbb{R}^d} F(\mathbf{x}),$$

- $F(\mathbf{x}) \triangleq f(\mathbf{x}) + h(\mathbf{x})$, where $f(\mathbf{x})$ is convex and smooth, and $h(\mathbf{x})$ is non-smooth.
- $F(\mathbf{x}) \triangleq \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{x})$, with each $f_i(\mathbf{x})$ smooth and possibly non-convex.
- $F(\mathbf{x}) \triangleq \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{a}_i^T \mathbf{x})$, where $\mathbf{a}_i \in \mathbb{R}^d, i = 1, \ldots, n$, are given.

Definition ((ϵ_g, ϵ_H)-Optimality)

Given $0 < \epsilon_g, \epsilon_H < 1$, x is an (ϵ_g, ϵ_H)-optimal solution if

$$\|\nabla F(x)\| \leq \epsilon_g, \quad \text{and} \quad \lambda_{\min}(\nabla^2 F(x)) \geq -\epsilon_H.$$

Approximate everything one can approximate

To increase efficiency, incorporate *approximations* of:

- *gradient information*, and
- *Hessian information*, and
- *inexact solutions* of the underlying *sub-problems*.

Sub-sample gradient and/or Hessian as:

$$\mathbf{g} \triangleq \frac{1}{|\mathcal{S}_g|} \sum_{i \in \mathcal{S}_g} \nabla f_i(x) \text{ and } \mathbf{H} \triangleq \frac{1}{|\mathcal{S}_H|} \sum_{i \in \mathcal{S}_H} \nabla^2 f_i(x),$$

where $\mathcal{S}_g, \mathcal{S}_H \subset \{1, \cdots, n\}$ are the sub-sample batches for gradient and Hessian.

Also consider, at step t, approximate solution of underlying sub-problem:

$$x^{(k+1)} = \arg \min_{\mathbf{x} \in \mathcal{D} \cap \mathcal{X}} \left\{ F(\mathbf{x}^{(k)}) + (\mathbf{x} - \mathbf{x}^{(k)})^T \mathbf{g}(\mathbf{x}^{(k)}) + \frac{1}{2\alpha_k} (\mathbf{x} - \mathbf{x}^{(k)})^T H(\mathbf{x}^{(k)}) (\mathbf{x} - \mathbf{x}^{(k)}) \right\}$$

Key result qua RandNLA

Xu, Roosta-Khorasani, and Mahoney "Inexact" 2017; and Yao, Xu, Roosta-Khorasani, and Mahoney "Inexact" 2018

Approximate gradient, $\mathbf{g}_t$, and inexact Hessian, $\mathbf{H}_t$, at each step t, must satisfy:

Condition (C1: Gradient and Hessian Approximation Error)

For some $0 < \delta_g, \delta_H < 1$, the approximate gradient/Hessian at step t must satisfy,

$$\begin{aligned} \|\mathbf{g}_t - \nabla F(x_t)\| &\leq \delta_g, \\ \|\mathbf{H}_t - \nabla^2 F(x_t)\| &\leq \delta_H. \end{aligned}$$

With uniform sampling (improvements possible with sketching & nonuniform sampling):

Lemma

For any $0 < \delta_g, \delta_H, \delta < 1$, let $\mathbf{g}$ and $\mathbf{H}$ be as in (96) with

$$|\mathcal{S}_g| \geq \frac{16K_g^2}{\delta_g^2} \log \frac{1}{\delta} \quad \text{and} \quad |\mathcal{S}_H| \geq \frac{16K_H^2}{\delta_H^2} \log \frac{2d}{\delta},$$

where $0 < K_g, K_H < \infty$ are such that $\|\nabla f_i(x)\| \leq K_g$ and $\left\|\nabla^2 f_i(x)\right\| \leq K_H$. Then, with probability at least $1 - \delta$, Condition C1 holds with the corresponding δ_g and δ_H.

Non-convex methods

Xu, Roosta-Khorasani, and Mahoney "Inexact" 2017; and Yao, Xu, Roosta-Khorasani, and Mahoney "Inexact" 2018

- Trust Region: Classical Method for Non-Convex Problem [Sorensen, 1982, Conn et al., 2000]

$$\mathbf{s}^{(k)} = \arg \min_{\|\mathbf{s}\| \leq \Delta_k} \langle \mathbf{s}, \nabla F(\mathbf{x}^{(k)}) \rangle + \frac{1}{2} \langle \mathbf{s}, \nabla^2 F(\mathbf{x}^{(k)}) \mathbf{s} \rangle$$

- Cubic Regularization: More Recent Method for Non-Convex Problem [Griewank, 1981, Nesterov et al., 2006, Cartis et al., 2011a, Cartis et al., 2011b]

$$\mathbf{s}^{(k)} = \arg \min_{\mathbf{s} \in \mathbb{R}^d} \langle \mathbf{s}, \nabla F(\mathbf{x}^{(k)}) \rangle + \frac{1}{2} \langle \mathbf{s}, \nabla^2 F(\mathbf{x}^{(k)}) \mathbf{s} \rangle + \frac{\sigma_k}{3} \|\mathbf{s}\|^3$$

A structural result for optimization

Xu, Roosta-Khorasani, and Mahoney "Inexact" 2017; and Yao, Xu, Roosta-Khorasani, and Mahoney "Inexact" 2018

- To get iteration complexity, all previous work required:

$$\left\| \left(H(\mathbf{x}^{(k)}) - \nabla^2 F(\mathbf{x}^{(k)}) \right) \mathbf{s}^{(k)} \right\| \leq C \|\mathbf{s}^{(k)}\|^2 \tag{4}$$

A structural result for optimization

Xu, Roosta-Khorasani, and Mahoney "Inexact" 2017; and Yao, Xu, Roosta-Khorasani, and Mahoney "Inexact" 2018

- To get iteration complexity, all previous work required:

$$\left\| \left(H(\mathbf{x}^{(k)}) - \nabla^2 F(\mathbf{x}^{(k)}) \right) \mathbf{s}^{(k)} \right\| \leq C \|\mathbf{s}^{(k)}\|^2 \tag{4}$$

- Stronger than "Dennis-Moré"

$$\lim_{k \to \infty} \frac{\| \left(H(\mathbf{x}(k)) - \nabla^2 F(\mathbf{x}(k)) \right) \mathbf{s}(k) \|}{\|\mathbf{s}(k)\|} = 0$$

A structural result for optimization

Xu, Roosta-Khorasani, and Mahoney "Inexact" 2017; and Yao, Xu, Roosta-Khorasani, and Mahoney "Inexact" 2018

- To get iteration complexity, all previous work required:

$$\left\| \left(H(\mathbf{x}^{(k)}) - \nabla^2 F(\mathbf{x}^{(k)}) \right) \mathbf{s}^{(k)} \right\| \leq C \|\mathbf{s}^{(k)}\|^2 \tag{4}$$

- Stronger than "Dennis-Moré"

$$\lim_{k \to \infty} \frac{\| \left(H(\mathbf{x}(k)) - \nabla^2 F(\mathbf{x}(k)) \right) \mathbf{s}(k) \|}{\|\mathbf{s}(k)\|} = 0$$

- Can relax (4) to

$$\left\| \left(H(\mathbf{x}^{(k)}) - \nabla^2 F(\mathbf{x}^{(k)}) \right) \mathbf{s}^{(k)} \right\| \leq \epsilon \|\mathbf{s}^{(k)}\| \tag{5}$$

permitting us to apply a large body of RandNLA sketching results.

A structural result for optimization

Xu, Roosta-Khorasani, and Mahoney "Inexact" 2017; and Yao, Xu, Roosta-Khorasani, and Mahoney "Inexact" 2018

- To get iteration complexity, all previous work required:

$$\left\| \left(H(\mathbf{x}^{(k)}) - \nabla^2 F(\mathbf{x}^{(k)}) \right) \mathbf{s}^{(k)} \right\| \leq C \|\mathbf{s}^{(k)}\|^2 \tag{4}$$

- Stronger than "Dennis-Moré"

$$\lim_{k \to \infty} \frac{\| \left(H(\mathbf{x}(k)) - \nabla^2 F(\mathbf{x}(k)) \right) \mathbf{s}(k) \|}{\|\mathbf{s}(k)\|} = 0$$

- Can relax (4) to

$$\left\| \left(H(\mathbf{x}^{(k)}) - \nabla^2 F(\mathbf{x}^{(k)}) \right) \mathbf{s}^{(k)} \right\| \leq \epsilon \|\mathbf{s}^{(k)}\| \tag{5}$$

permitting us to apply a large body of RandNLA sketching results.

- Quasi-Newton, Sketching, Sub-Sampling satisfy Dennis-Moré and (5) but not necessarily (4).

For more details ...

... see Fred Roosta-Khorasani's talk tomorrow.

Outline

Conclusions

- RandNLA—combining linear algebra and probability—is at the center of the foundations of data.
- Sampling—in the given basis or in a randomly-rotated basis—is a core primitive in RandNLA.
- Randomness can be in the data and/or in the algorithm, and there can be interesting/fruitful interactions between the two:
  - Best works-case algorithms (TCS-style) for very overdetermined least-squares problems.
  - Implementations (NLA-style) are competitive with and can beat the best high-quality NLA libraries.
  - Implementations (in Spark/MPI) can compute low, medium, and high precision solutions on up to terabyte-sized data.
  - Inferential guarantees in statistics, machine learning, and data science applications $\Rightarrow$ require going beyond core RandNLA.
  - Improvements in first-order/second-order convex/non-convex optimization theory/practice $\Rightarrow$ require going beyond core RandNLA.