Randomized Numerical Linear Algebra (**RandNLA**): Past, Present, and Future, cont.

Petros Drineas[1] & Michael W. Mahoney[2]

[1] Department of Computer Science, Rensselaer Polytechnic Institute
[2] Department of Mathematics, Stanford University

To access our web pages:

Google Drineas

Google Michael Mahoney

Roadmap of the tutorial

Focus: sketching matrices (i) by sampling rows/columns and (ii) via "random projections."

Machinery: (i) Approximating matrix multiplication and (ii) Decoupling "randomization" from "matrix perturbation."

Overview of the tutorial:

(i) Motivation: computational efficiency, interpretability

(ii) Approximating matrix multiplication

(iii) From matrix multiplication to CX/CUR factorizations and the SVD

(iv) Improvements and recent progress

(v) Algorithmic approaches to least-squares problems

(vi) Statistical perspectives on least-squares algorithms

(vii) Theory and practice of: extending these ideas to kernels and SPSD matrices

(viii) Theory and practice of: implementing these ideas in large-scale settings

Why randomized matrix algorithms?

- Faster algorithms: worst-case theory and/or numerical code
- Simpler algorithms: easier to analyze and reason about
- More-interpretable output: useful if analyst time is expensive
- Implicit regularization properties: and more robust output
- Exploit modern computer architectures: by reorganizing steps of alg
- Massive data: matrices that they can be stored only in slow secondary memory devices or even not at all

Already a big success … but why do they work?

Already a big success …

Avron, Maymounkov, and Toledo 2010:

- "Randomization is arguably the most exciting and innovative idea to have hit linear algebra in a long time"
- Blendenpik "beats Lapack's direct dense least-squares solver by a large margin on essentially any dense tall matrix"
- Empirical results "show the potential of random sampling algorithms and suggest that random projection algorithms should be incorporated into future versions of Lapack."

Already a big success … but why do they work?

Already a big success ...

- Better worse-case theory: for L2 regression, L1 regression, low-rank matrix approximation, column subset selection, Nystrom approximation, etc.
- Implementations "beat" Lapack: for L2 regression on nearly any non-tiny tall dense matrix
- Low-rank implementations "better": in terms of running time and/or robustness for dense/sparse scientific computing matrices
- Parallel and distributed implementations: exploit modern computer architectures to do computations on up to a tera-byte of data
- Genetics, astronomy, etc.: applications to choose good SNPs, wavelengths, etc. for genotype inference, galaxy identification, etc.

Already a big success … but why do they work?

A typical result: (1+ε)-CX/CUR

Theorem: Let $T_{SVD,k}$ time* be the time to compute an *exact or approximate* rank-k approximation to the SVD (e.g., with a random projection). Then, given an m-by-n matrix A, there exists** an algorithm that runs in $O(T_{SVD,k})$ time that picks

at most roughly 3200*** $(k/\varepsilon^2$****$)\log(k/\varepsilon)$ columns of A

such that with probability at least 0.9*****

$$\| A - P_C A \|_F \le (1+\varepsilon) \| A - A_k \|_F$$

*Isn't that too expensive?

**What is it?

***Isn't 3200 to big? Why do you *need* 3200?

****Isn't $1/\varepsilon^2$ too bad for $\varepsilon \cong 10^{-15}$?

*****Isn't 0.1 too large a failure probability?

Why do these algorithms work?

They decouple randomness from vector space structure.

Today, explain this in the context of.

- Least squares regression -> CX/CUR approximation
- CSSP -> Random Projections parameterized more flexibly
- Nystrom approximation of SPSD matrices

Permits finer control in applying the randomization.

- Much better worst-case theory
- Easier to map to ML and statistical ideas
- Easier to parameterize problems in ways that are more natural to numerical analysts, scientific computers, and software developers

The devil is in the details ...

Decouple the randomization from the linear algebra:

- originally within the analysis, then made explicit
- permits much finer control in application of randomization

Importance of statistical leverage scores:

- historically used in regression diagnostics to identify outliers
- best random sampling algorithms use them as importance sampling distribution
- best random projection algorithms go to a random basis where they are roughly uniform

Couple with domain expertise—to get best results!

Statistical leverage, coherence, etc.

Mahoney and Drineas (2009, PNAS); Drineas, Magdon-Ismail, Mahoney, and Woodruff (2012, ICML)

Definition: Given a "tall" n x d matrix A, i.e., with n > d, let U be *any* n x d orthogonal basis for span(A), & let the d-vector $U_{(i)}$ be the ith *row* of U. Then:

- the statistical leverage scores are $\lambda_i = ||U_{(i)}||_2^2$, for $i \in \{1,\dots,n\}$
- the coherence is $\gamma = \max_{i \in \{1,\dots,n\}} \lambda_i$
- the (i,j)-cross-leverage scores are $U_{(i)}^T U_{(j)} = \langle U_{(i)}, U_{(j)} \rangle$

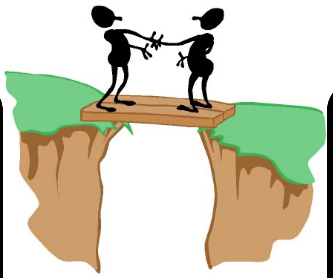
Note: There are extension of this to:

- "fat" matrices A, with n, d are large and low-rank parameter k
- L1 and other p-norms

History of Randomized Matrix Algs

Theoretical origins

- theoretical computer science, convex analysis, etc.
- Johnson-Lindenstrauss
- Additive-error algs
- Good worst-case analysis
- No statistical analysis



Practical applications

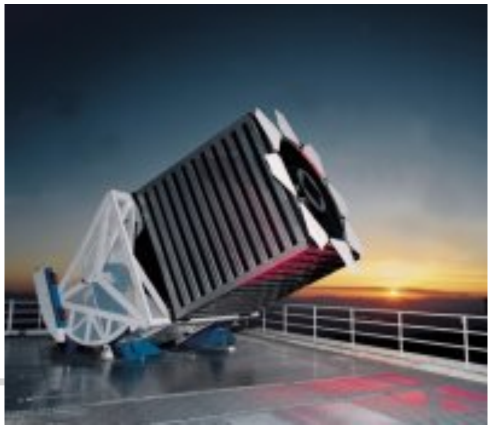
- NLA, ML, statistics, data analysis, genetics, etc
- Fast JL transform
- Relative-error algs
- Numerically-stable algs
- Good statistical properties

How to "bridge the gap"?

- decouple randomization from linear algebra
- importance of statistical leverage scores!

Applications in: Astronomy

Szalay (2012, MMDS)



CMB Surveys (pixels)

1990	COBE	1000
2000	Boomerang	10,000
2002	CBI	50,000
2003	WMAP	1 Million
2008	Planck	10 Million

Angular Galaxy Surveys (obj)

1970	Lick	1M
1990	APM	2M
2005	SDSS	200M
2011	PS1	1000M
2020	LSST	30000M

Time Domain

- QUEST
- SDSS Extension survey
- Dark Energy Camera
- Pan-STARRS
- LSST…

Galaxy Redshift Surveys (obj)

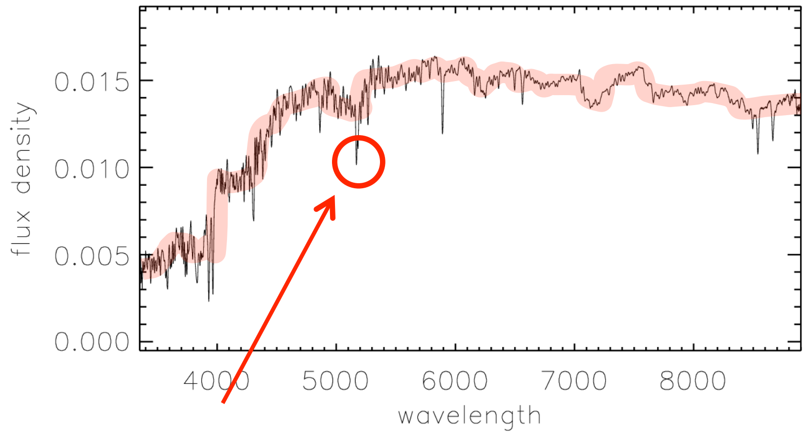
1986	CfA	3500
1996	LCRS	23000
2003	2dF	250000
2008	SDSS	1000000
2012	BOSS	2000000
2012	LAMOST	2500000

"The Age of Surveys" – generate petabytes/year …

Galaxy properties from galaxy spectra

Szalay (2012, MMDS)

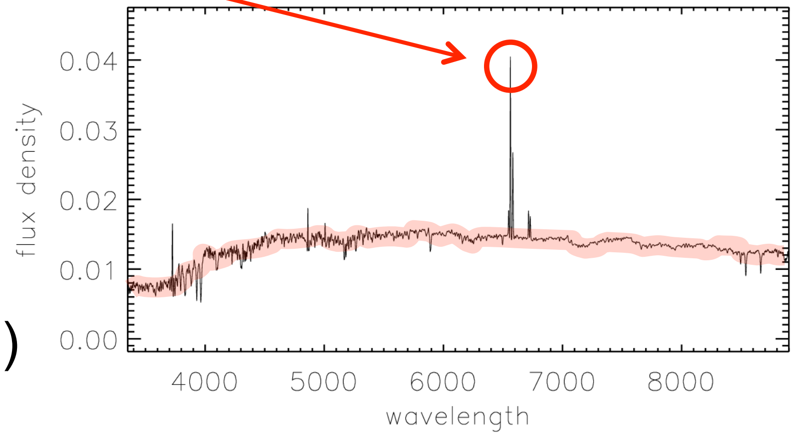
4K x 1M SVD Problem: ideal for randomized matrix algorithms



Spectral Lines

Continuum Emissions

Can we select "informative" frequencies (columns) or images (rows) "objectively"?



Galaxy diversity from PCA

PC

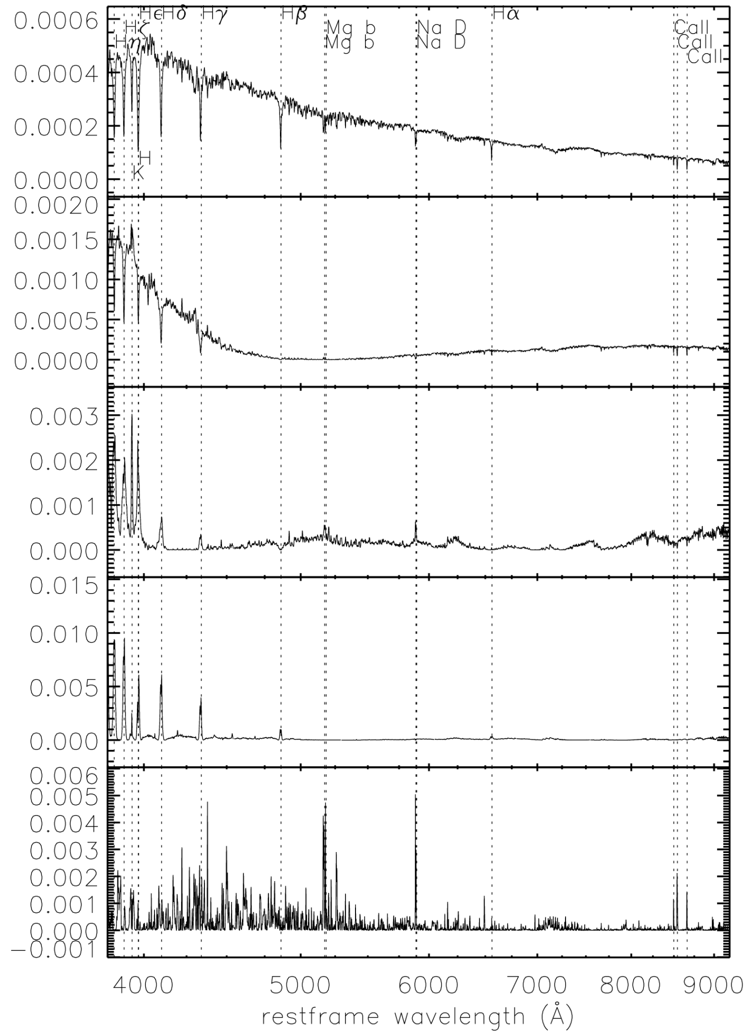
1st

2nd

3rd

4th

5th



[Average Spectrum]

[Stellar Continuum]

[Finer Continuum Features + Age]

[Age]
Balmer series hydrogen lines

[Metallicity]
Mg b, Na D, Ca II Triplet

Roadmap of the tutorial

Focus: sketching matrices by (i) sampling rows/columns and (ii) via "random projections."

Machinery: (i) Approximating matrix multiplication and (ii) Decoupling "randomization" from "matrix perturbation."

Overview of the tutorial:

(i) Motivation (computational efficiency, interpretability)

(ii) Approximating matrix multiplication

(iii) From matrix multiplication to CX/CUR factorizations and the SVD

(iv) Improvements and recent progress

(v) Algorithmic approaches to least-squares problems

(vi) Statistical perspectives on least-squares algorithms

(vii) Theory and practice of: extending these ideas to kernels and SPSD matrices

(viii) Theory and practice of: implementing these ideas in large-scale settings

Least Squares (LS) Approximation

$$\begin{pmatrix} \\ \\ A \\ \\ \\ \end{pmatrix} \begin{pmatrix} \widehat{x} \end{pmatrix} \approx \begin{pmatrix} \\ \\ b \\ \\ \\ \end{pmatrix} \quad \Rightarrow \quad \begin{aligned} \mathcal{Z}_2 &= \min_{x \in \mathbb{R}^d} ||b - Ax||_2 \\ &= ||b - A\hat{x}||_2 \end{aligned}$$

$n \times d \quad , \quad n \gg d$

We are interested in over-constrained Lp regression problems, *n* >> *d*.

Typically, there is no *x* such that *Ax* = *b*.

Want to find the “best” x such that *Ax* ≈ *b*.

Ubiquitous in applications & central to theory:

Statistical interpretation: best linear unbiased estimator.

Geometric interpretation: orthogonally project b onto span(A).

Exact solution to LS Approximation

Cholesky Decomposition:

If A is full rank and well-conditioned,

decompose $A^TA = R^TR$, where R is upper triangular, and

solve the normal equations: $R^TRx=A^Tb$.

QR Decomposition:

Slower but numerically stable, esp. if A is rank-deficient.

Write $A=QR$, and solve $Rx = Q^Tb$.

Singular Value Decomposition:

Most expensive, but best if A is very ill-conditioned.

Write $A=U\Sigma V^T$, in which case: $\mathbf{x_{OPT} = A^+b = V\Sigma^{-1}_k U^T b}$.

Complexity is O(nd²) for all of these, but constant factors differ.

$$\begin{aligned}\mathcal{Z}_2 &= \min_{x\in R^d} ||b-Ax||_2 \\ &= ||b-A\hat{x}||_2\end{aligned}$$

Projection of *b* on the subspace spanned by the columns of *A*

$$\begin{aligned}\mathcal{Z}_2^2 &= ||b||_2^2 - ||AA^+b||_2^2 \\ \hat{x} &= A^+b\end{aligned}$$

Pseudoinverse of *A*

Modeling with Least Squares

Assumptions underlying its use:

- Relationship between “outcomes” and “predictors is (roughly) linear.
- The error term ε has mean zero.
- The error term ε has constant variance.
- The errors are uncorrelated.
- The errors are normally distributed (or we have adequate sample size to rely on large sample theory).

Should always check to make sure these assumptions have not been (too) violated!

Statistical Issues and Regression Diagnostics

Model: $b = Ax+\varepsilon$	b = response; $A^{(i)}$ = carriers;
	ε = error process s.t.: mean zero, const. varnce, (i.e., $E(e)=0$ and $Var(e)=\sigma^2 I$), uncorrelated, normally distributed
$x_{opt} = (A^TA)^{-1}A^Tb$	(what we computed before)
$b' = Hb$	$H = A(A^TA)^{-1}A^T$ = "hat" matrix
	H_{ij} - measures the leverage or influence exerted on b'_i by b_j, *regardless* of the value of b_j (since H depends only on A)
$e' = b-b' = (I-H)b$	vector of residuals - note: $E(e')=0$, $Var(e')=\sigma^2(I-H)$
$Trace(H)=d$	Diagnostic Rule of Thumb: Investigate if $H_{ii} > 2d/n$
$H=UU^T$	U is from SVD ($A=U\Sigma V^T$), or *any* orthogonal matrix for span(A)
$H_{ii} = \lvert U^{(i)}\rvert_2^2$	leverage scores = row "lengths" of spanning orthogonal matrix

A “classic” randomized algorithm (1of3)

Drineas, Mahoney, and Muthukrishnan (2006, SODA & 2008, SIMAX)

Over-constrained least squares (n x d matrix A,n >>d)

- Solve: $\mathcal{Z} = \min_{x \in R^d} ||Ax - b||_2$
- Solution: $x_{opt} = A^{\dagger} b$

Randomized Algorithm:

- For all i ε {1,…,n}, compute $p_i = \frac{1}{d} ||U_{(i)}||_2^2$
- Randomly sample O(d log(d)/ ε) rows/elements fro A/b, using *{p$_i$} as importance sampling probabilities.*
- Solve the induced subproblem: $\tilde{x}_{opt} = (SA)^{\dagger} Sb$

A "classic" randomized algorithm (2of3)

Drineas, Mahoney, and Muthukrishnan (2006, SODA & 2008, SIMAX)

Theorem: Let $\gamma = ||U_A U_A^T b||_2 / ||b||_2$. Then:

- $||A\tilde{x}_{opt} - b||_2 \leq (1+\epsilon)\mathcal{Z}$
- $||x_{opt} - \tilde{x}_{opt}||_2 \leq \sqrt{\epsilon}\left(\kappa(A)\sqrt{\gamma^{-2}-1}\right)||x_{opt}||_2$

This naïve algorithm runs in O(nd^2) time

- But it can be improved !!!

This algorithm is bottleneck for Low Rank Matrix Approximation and many other matrix problems.

A "classic" randomized algorithm (3of3)

Drineas, Mahoney, and Muthukrishnan (2006, SODA & 2008, SIMAX)

Sufficient condition for relative-error approximation.

For the "preprocessing" matrix X:

$$\sigma^2_{min}(XU_A) \geq 1/\sqrt{2}; \text{ and}$$
$$||U_A^T X^T X b^{\perp}||_2^2 \leq \epsilon \mathcal{Z}^2/2,$$

• *Important: this condition decouples the randomness from the linear algebra.*

• Random sampling algorithms with leverage score probabilities and random projections satisfy it!

Theoretically "fast" algorithms

Drineas, Mahoney, Muthukrishnan, and Sarlos (2007); Drineas, Magdon-Ismail, Mahoney, and Woodruff (2011)

Algorithm 1: Fast Random Projection Algorithm for LS Problem

- Preprocess input (in $o(nd^2)$time) with Fast-JL transform, uniformizes leverage scores, and sample uniformly in the randomly-rotated space
- Solve the induced subproblem

Algorithm 2: Fast Random Sampling Algorithm for LS Problem

- Compute $1\pm\varepsilon$ approximation to statistical leverage scores (in $o(nd^2)$ time), and use them as importance sampling probabilities
- Solve the induced subproblem

<u>Main theorem</u>: For both of these randomized algorithms, we get:

- *$(1\pm\varepsilon)$-approximation*
- *in roughly* $O\left(nd\log\left(d\log(n)/\epsilon\right) + d^3\log(n)\log(d\log n)/\epsilon\right)$ *time!!*

Practically “fast” implementations (1of2)

Use “randomized sketch” to construct preconditioner for traditional iterative methods:

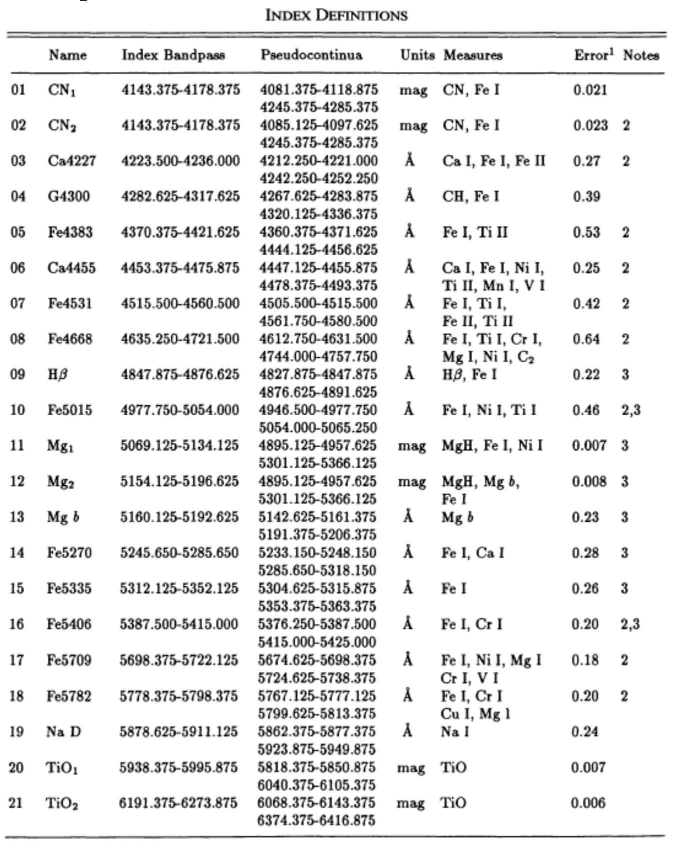
- RT08: preconditioned iterative method improves $1/\varepsilon$ dependence to $\log(1/\varepsilon)$, important for high precision
- AMT10: much more detailed evaluation, different Hadamard-type preconditioners, etc.
- CRT11: use Gaussian projections to compute orthogonal projections with normal equations
- MSM11: use Gaussian projections and LSQR or Chebyshev semi-iterative method to minimize communication, e.g., for parallel computation in Amazon EC2 clusters!

Practically “fast” implementations (2of2)

Avron, Maymounkov, and Toledo 2010:

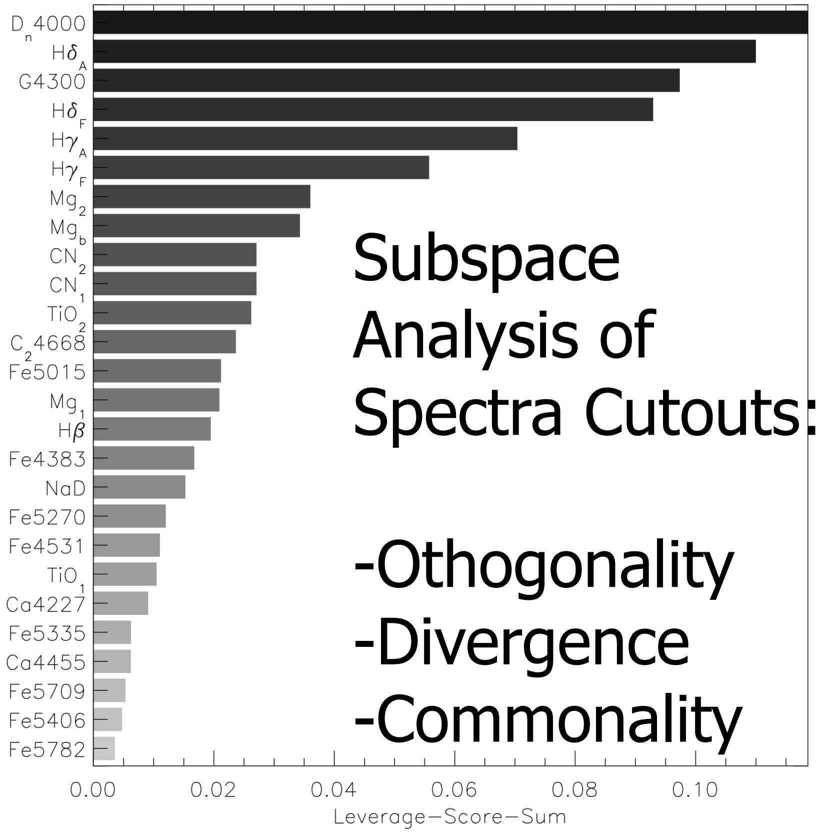
- *Blendenpik "beats Lapack's direct dense least-squares solver by a large margin on essentially any dense tall matrix"*
- *Empirical results "show the potential of random sampling algorithms and suggest that random projection algorithms should be incorporated into future versions of Lapack."*

Ranking Astronomical Line Indices



INDEX DEFINITIONS

	Name	Index Bandpass	Pseudocontinua	Units	Measures	Error[1]	Notes
01	CN_1	4143.375-4178.375	4081.375-4118.875 4245.375-4285.375	mag	CN, Fe I	0.021	
02	CN_2	4143.375-4178.375	4085.125-4097.625 4245.375-4285.375	mag	CN, Fe I	0.023	2
03	Ca4227	4223.500-4236.000	4212.250-4221.000 4242.250-4252.250	Å	Ca I, Fe I, Fe II	0.27	2
04	G4300	4282.625-4317.625	4267.625-4283.875 4320.125-4336.375	Å	CH, Fe I	0.39	
05	Fe4383	4370.375-4421.625	4360.375-4371.625 4444.125-4456.625	Å	Fe I, Ti II	0.53	2
06	Ca4455	4453.375-4475.875	4447.125-4455.875 4478.375-4493.375	Å	Ca I, Fe I, Ni I, Ti II, Mn I, V I	0.25	2
07	Fe4531	4515.500-4560.500	4505.500-4515.500 4561.750-4580.500	Å	Fe I, Ti I, Fe II, Ti II	0.42	2
08	Fe4668	4635.250-4721.500	4612.750-4631.500 4744.000-4757.750	Å	Fe I, Ti I, Cr I, Mg I, Ni I, C_2	0.64	2
09	$H\beta$	4847.875-4876.625	4827.875-4847.875 4876.625-4891.625	Å	$H\beta$, Fe I	0.22	3
10	Fe5015	4977.750-5054.000	4946.500-4977.750 5054.000-5065.250	Å	Fe I, Ni I, Ti I	0.46	2,3
11	Mg_1	5069.125-5134.125	4895.125-4957.625 5301.125-5366.125	mag	MgH, Fe I, Ni I	0.007	3
12	Mg_2	5154.125-5196.625	4895.125-4957.625 5301.125-5366.125	mag	MgH, Mg *b*, Fe I	0.008	3
13	Mg *b*	5160.125-5192.625	5142.625-5161.375 5191.375-5206.375	Å	Mg *b*	0.23	3
14	Fe5270	5245.650-5285.650	5233.150-5248.150 5285.650-5318.150	Å	Fe I, Ca I	0.28	3
15	Fe5335	5312.125-5352.125	5304.625-5315.875 5353.375-5363.375	Å	Fe I	0.26	3
16	Fe5406	5387.500-5415.000	5376.250-5387.500 5415.000-5425.000	Å	Fe I, Cr I	0.20	2,3
17	Fe5709	5698.375-5722.125	5674.625-5698.375 5724.625-5738.375	Å	Fe I, Ni I, Mg I Cr I, V I	0.18	2
18	Fe5782	5778.375-5798.375	5767.125-5777.125 5799.625-5813.375	Å	Fe I, Cr I Cu I, Mg l	0.20	2
19	Na D	5878.625-5911.125	5862.375-5877.375 5923.875-5949.875	Å	Na I	0.24	
20	TiO_1	5938.375-5995.875	5818.375-5850.875 6040.375-6105.375	mag	TiO	0.007	
21	TiO_2	6191.375-6273.875	6068.375-6143.375 6374.375-6416.875	mag	TiO	0.006	



Subspace Analysis of Spectra Cutouts:

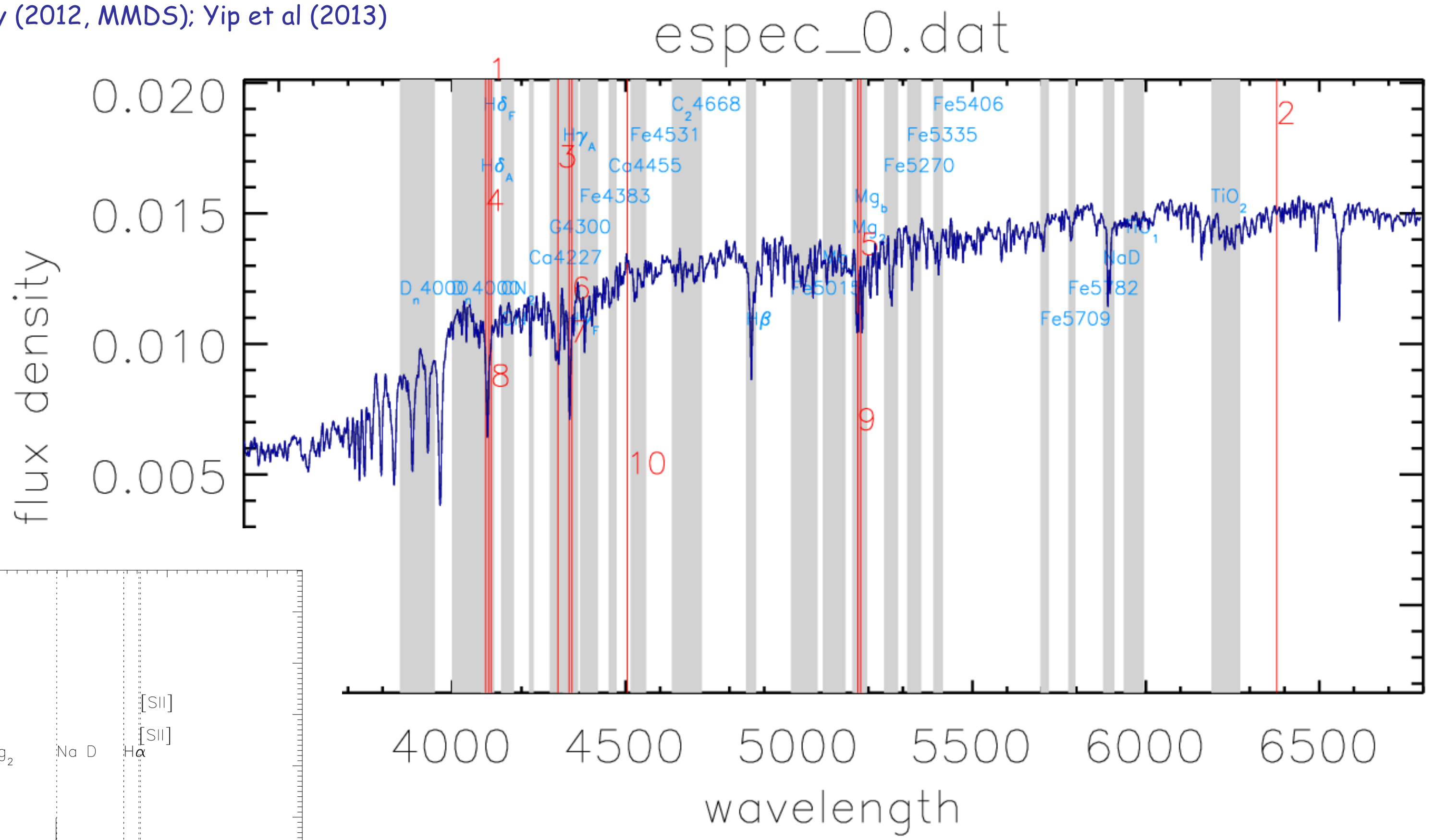
- -Othogonality
- -Divergence
- -Commonality

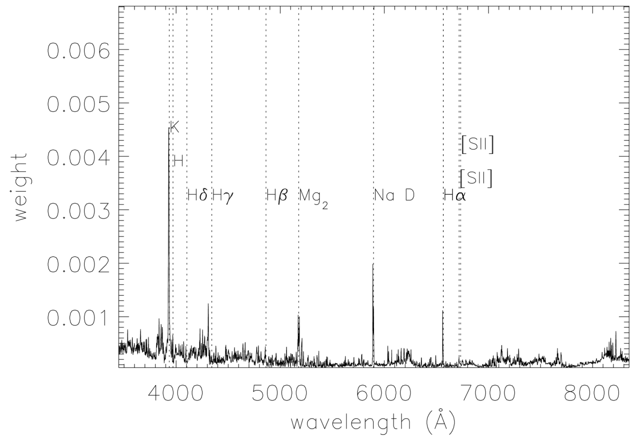
(Worthey et al. 94; Trager et al. 98)

(Yip et al. 2013 subm.)

Identifying new line indices objectively

Szalay (2012, MMDS); Yip et al (2013)





(Yip et al. 2013 subm.)

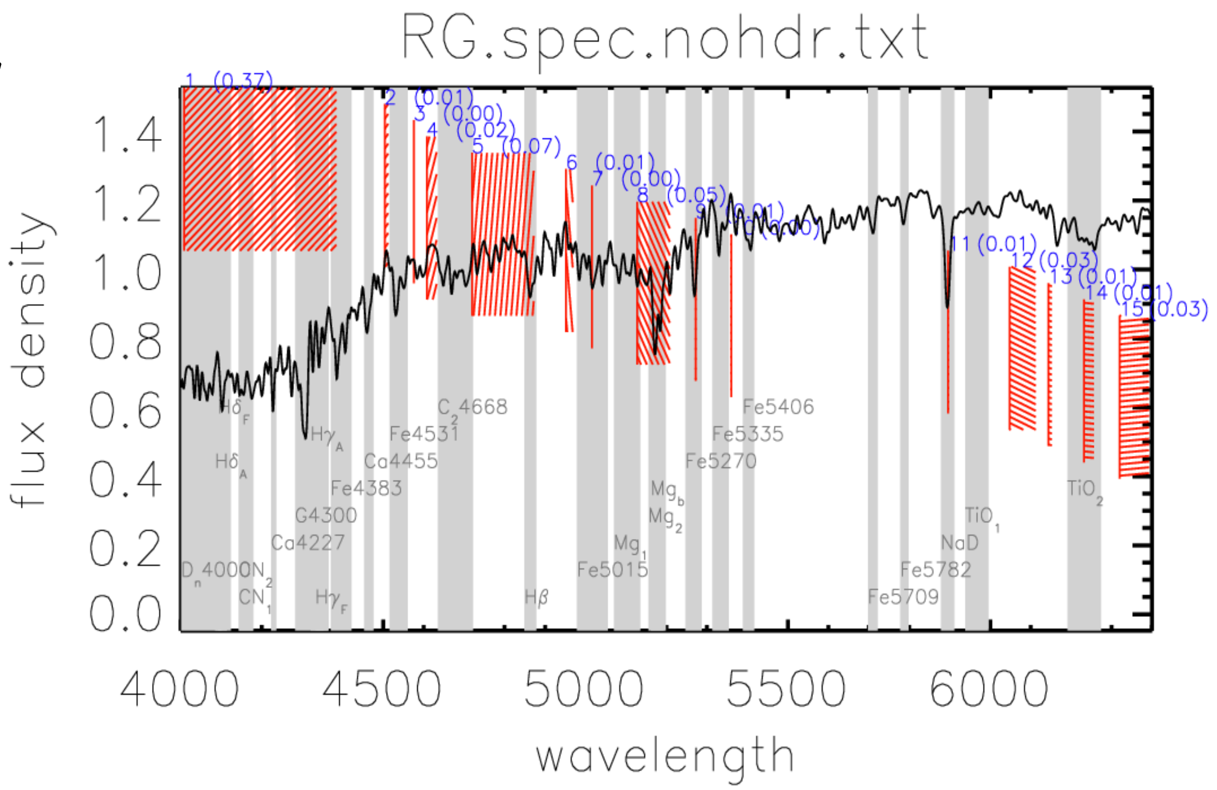
New Spectral Regions (M2;k=5; overselecting 10X; combine if <30A)

Szalay (2012, MMDS); Yip et al (2013)

Old Lick indices are "ad hoc"

New indices are "objective"

- Recover atomic lines
- Recover molecular bands
- Recover Lick indices
- Informative regions are orthogonal to each other, in contrast to Lick regions



(Yip et al. 2013 subm.)

Roadmap of the tutorial

<u>Focus:</u> sketching matrices by (i) sampling rows/columns and (ii) via "random projections."

<u>Machinery:</u> (i) Approximating matrix multiplication and (ii) Decoupling "randomization" from "matrix perturbation."

<u>Overview of the tutorial:</u>

(i) Motivation (computational efficiency, interpretability)

(ii) Approximating matrix multiplication

(iii) From matrix multiplication to CX/CUR factorizations and the SVD

(iv) Improvements and recent progress

(v) Algorithmic approaches to least-squares problems

(vi) Statistical perspectives on least-squares algorithms

(vii) Theory and practice of: extending these ideas to kernels and SPSD matrices

(viii) Theory and practice of: implementing these ideas in large-scale settings

A statistical perspective on "leveraging"

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 or design matrix, β_0 is a $p \times 1$ coefficient vector, and the noise vector $\epsilon \sim N(0, \sigma^2 I)$. In this case,

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

Constructing the subsample

Main "Algorithmic Leveraging" Algorithm:

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 $\text{argmin}_{\beta \in R^p} ||DS_X^T y - DS_X^T X\beta||^2$.
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
- 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)

"A statistical perspective on algorithmic leveraging," 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)

"A statistical perspective on algorithmic leveraging," Ma, Mahoney, and Yu 2013

Lemma. (MMY13) The *conditional* expectation and conditional 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 and unconditional variance for the is given by:

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

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

Bias and variance of subsampling estimators (3 of 3)

"A statistical perspective on algorithmic leveraging," Ma, Mahoney, and Yu 2013

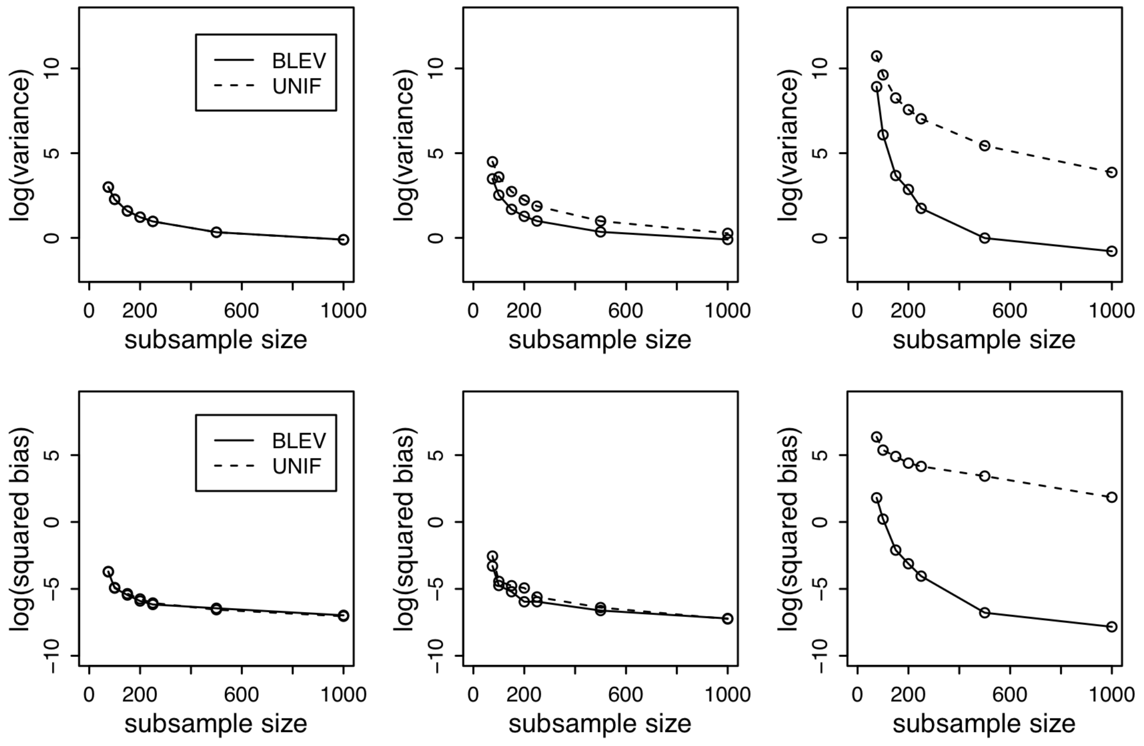
Consider empirical performance of several versions:

- UNIF: variance scales as n/r
- BLEV: variance scales as p/r but have $1/h_{ii}$ terms in denominator of sandwich expression
- SLEV: variance scales as p/r but $1/h_{ii}$ terms in denominator are moderated since no probabilities are too small
- UNWL: $1/h_{ii}$ terms are not in denominator, but estimates unbiased around β_{wls}/β_0

Estimates are unbiased (around β_{ols}/β_0), but variance depends on sampling probabilities.

BLEV and UNIF on data with different leverage scores

"A statistical perspective on algorithmic leveraging," Ma, Mahoney, and Yu 2013



Empirical variances and squared biases of the BLEV and UNIF estimators in three data sets for n=1000 and p=50. Left to right, Gaussian, multivariate-t with 3 d.o.f. (T3), and multivariate-t with 1 d.o.f. (T1).

BLEV and UNIF when rank is lost, 1

Comparison of BLEV and UNIF when rank is lost in the sampling process (n=1000 and p=10).

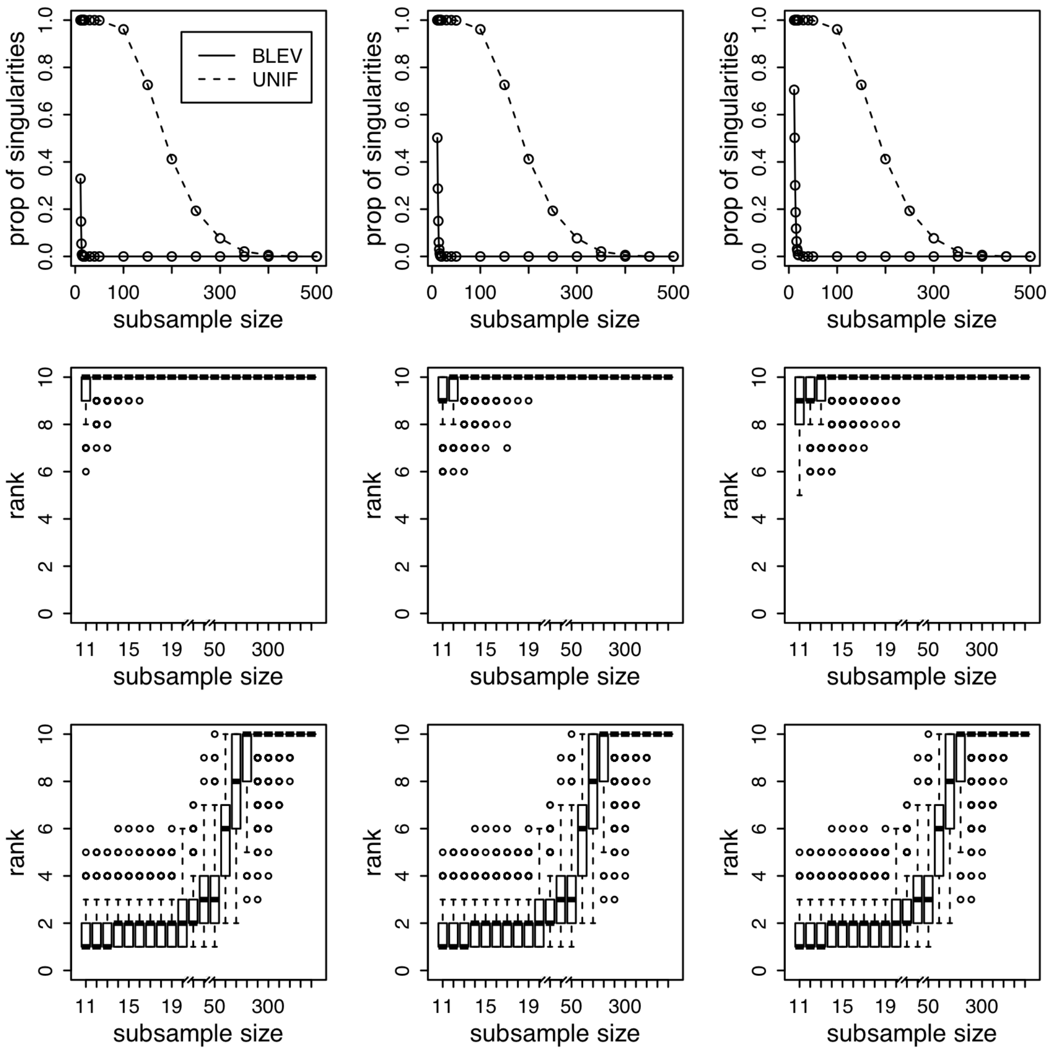
Left/middle/right panels: T3/T2/T1 data.

Upper panels: Proportion of singular X^TWX, out of 500 trials, for BLEV and UNIF .

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

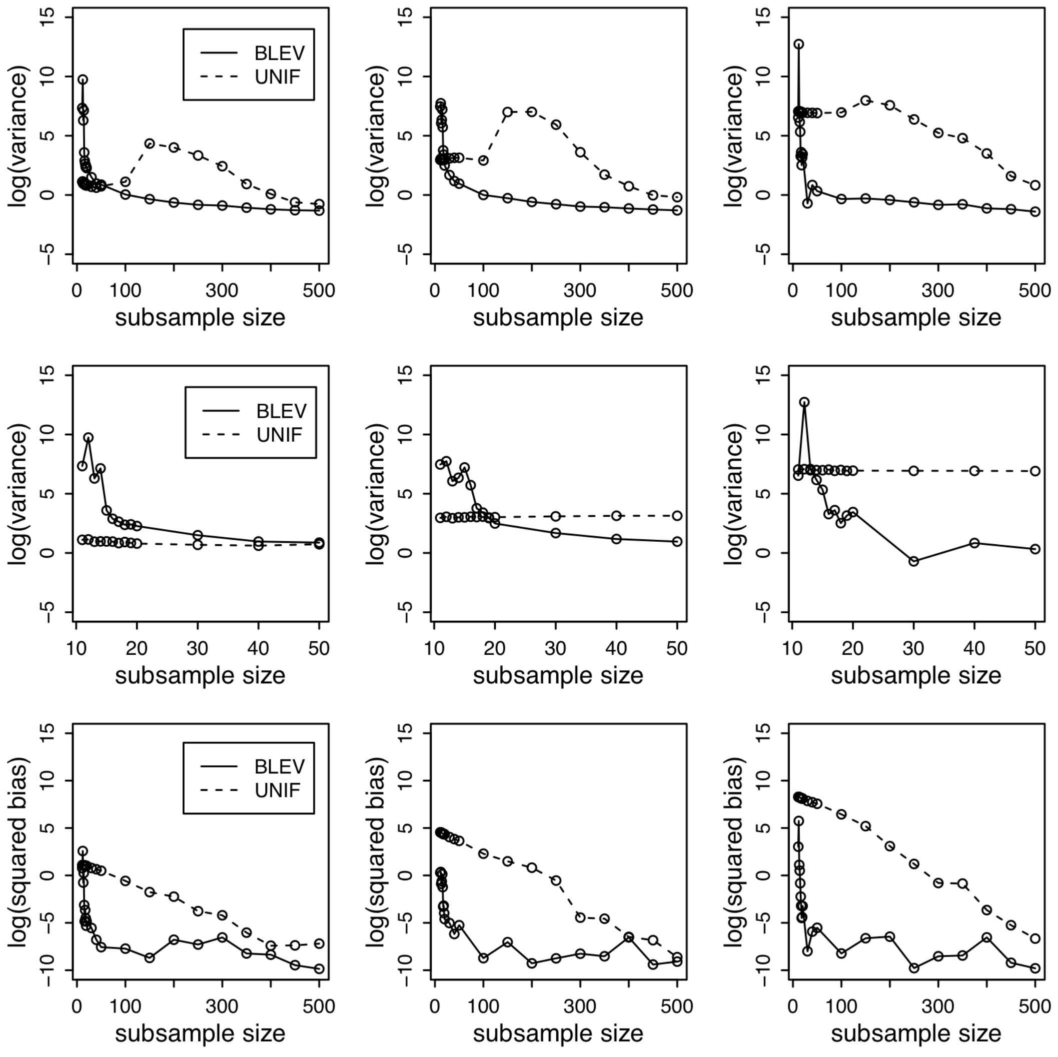
Comparison of BLEV and UNIF when rank is lost in the sampling process (n=1000 and p=10).

Left/middle/right panels: T3/T2/T1 data.

Upper panels: The logarithm of variances of the estimates.

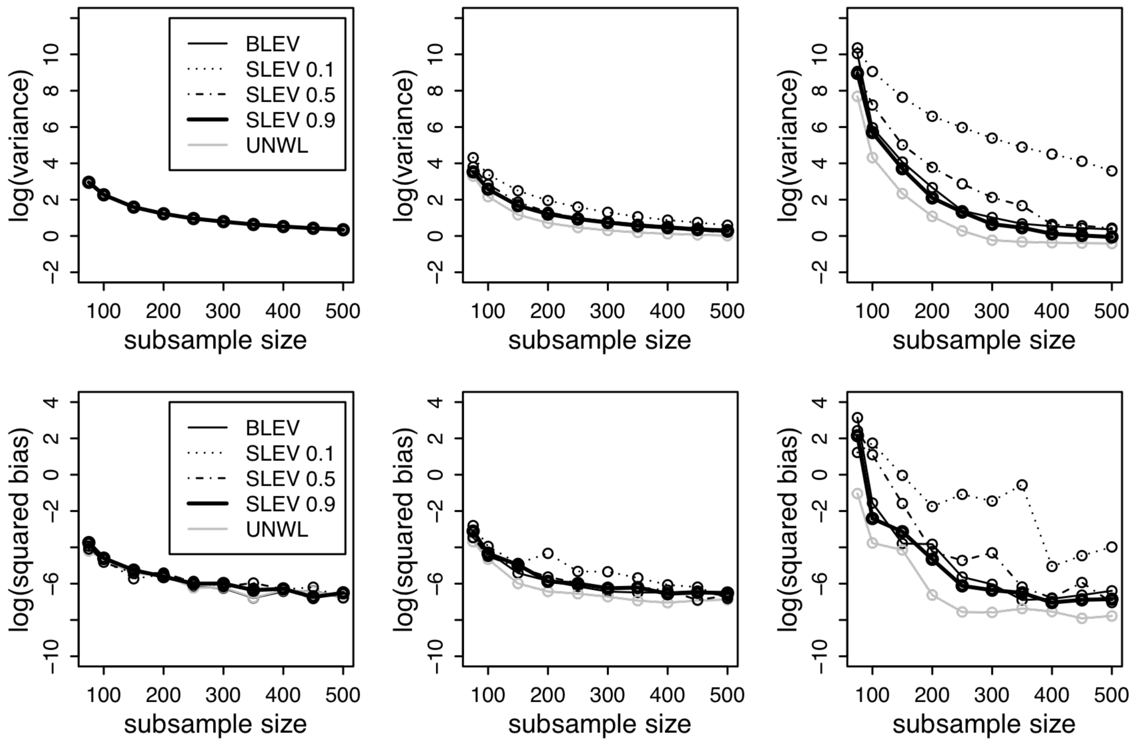
Middle panels: The logarithm of variances, zoomed-in on the X-axis.

Lower panels: The logarithm of squared bias of the estimates.



Combining BELV and UNIF into SLEV, 1

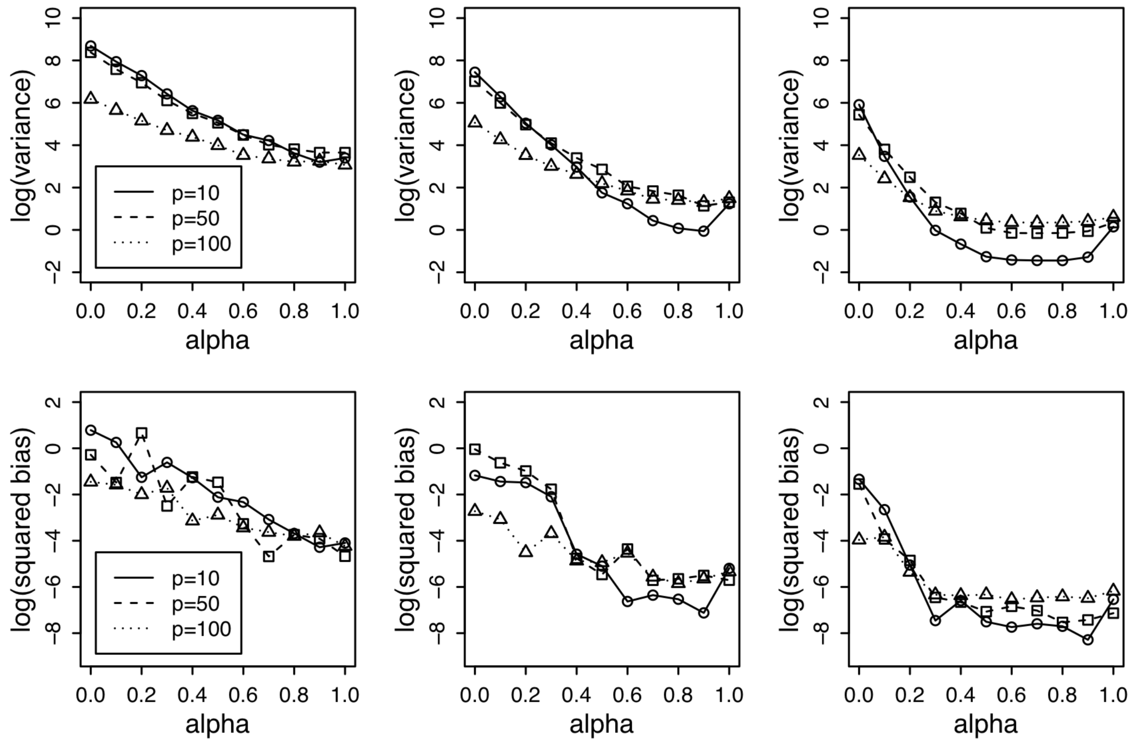
"A statistical perspective on algorithmic leveraging," Ma, Mahoney, and Yu 2013



Empirical variances and squared biases (*unconditional*) of the BLEV, SLEV, and UNWL estimators in three data sets (GA, T3, and T1) for n=1000 and p=50. Left/middle/right panels: GA/T3/T1 data.

Combining BLEV and UNIF into SLEV, 2

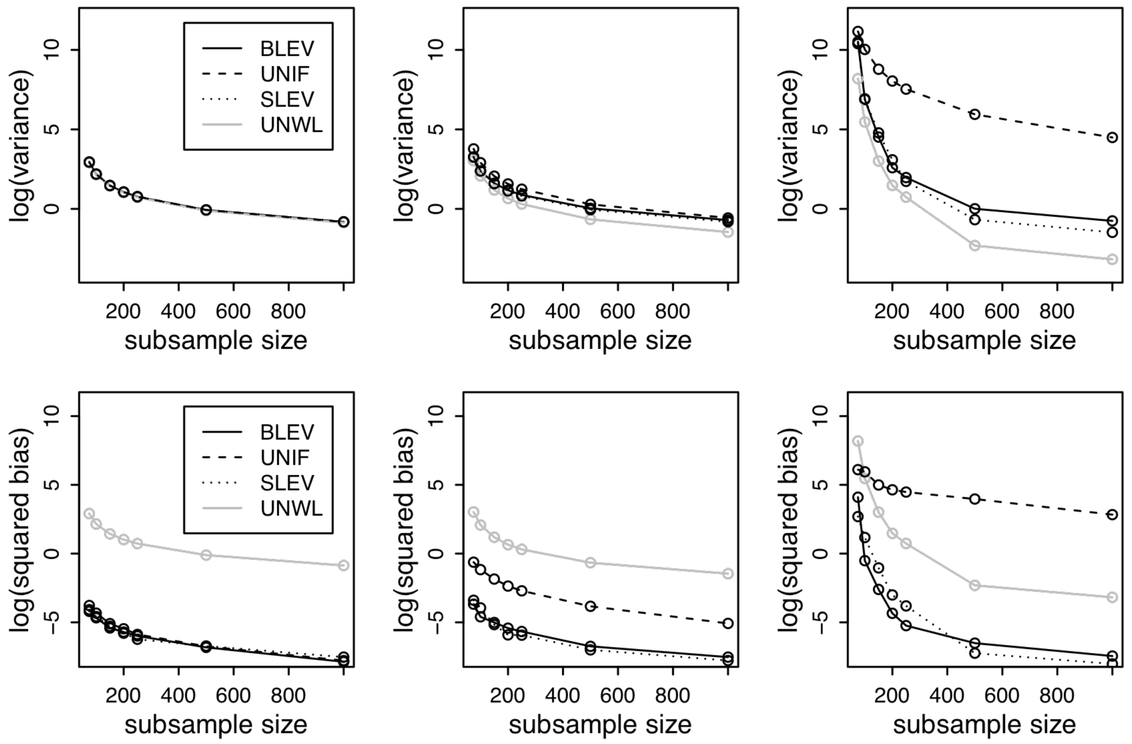
"A statistical perspective on algorithmic leveraging," Ma, Mahoney, and Yu 2013



Empirical variances and squared biases (*unconditional*) of the SLEV estimator in data generated from T1 with n=1000 and variable p. Left/middle/right panel: subsample size r=3p/r=5p/r=10p.

Results conditioned on the data

"A statistical perspective on algorithmic leveraging," Ma, Mahoney, and Yu 2013



Empirical variances and squared biases (*conditional*) of the BLEV and UNIF estimators in 3 data sets (GA, T3, and T1) for n=1000 & p=50. Upper/lower panels: Variances/Squared bias. Left/middle/lower panels: GA/T3/T1 data.

Roadmap of the tutorial

Focus: sketching matrices by (i) sampling rows/columns and (ii) via "random projections."

Machinery: (i) Approximating matrix multiplication and (ii) Decoupling "randomization" from "matrix perturbation."

Overview of the tutorial:

(i) Motivation (computational efficiency, interpretability)

(ii) Approximating matrix multiplication

(iii) From matrix multiplication to CX/CUR factorizations and the SVD

(iv) Improvements and recent progress

(v) Algorithmic approaches to least-squares problems

(vi) Statistical perspectives on least-squares algorithms

(vii) Theory and practice of: extending these ideas to kernels and SPSD matrices

(viii) Theory and practice of: implementing these ideas in large-scale settings

Motivation (1 of 2)

Methods to extract linear structure from the data:

- Support Vector Machines (SVMs).
- Gaussian Processes (GPs).
- Singular Value Decomposition (SVD) and the related PCA.

Kernel-based learning methods to extract non-linear structure:

- Choose features to define a (dot product) space F.
- Map the data, X, to F by $\phi: X \to F$.
- Do classification, regression, and clustering in F with linear methods.

Motivation (2 of 2)

- Use dot products for information about mutual positions.
- Define the kernel or Gram matrix: $G_{ij}=k_{ij}=(\phi(X^{(i)}), \phi(X^{(j)}))$.
- Algorithms that are expressed in terms of dot products can be given the Gram matrix G instead of the data covariance matrix X^TX.

If the Gram matrix G -- $G_{ij}=k_{ij}=(\phi(X^{(i)}), \phi(X^{(j)}))$ -- is dense but (nearly) low-rank, then calculations of interest still need $O(n^2)$ space and $O(n^3)$ time:

- matrix inversion in GP prediction,
- quadratic programming problems in SVMs,
- computation of eigendecomposition of G.

Idea: use random sampling/projections to speed up these computations!

This “revisiting” is particularly timely …

“Revisiting the Nystrom Method …,” Gittens and Mahoney (2013)

Prior existing theory was *extremely* weak:

- Especially compared with very strong $1\pm\varepsilon$ results for low-rank approximation, least-squares approximation, etc. of general matrices
- In spite of the empirical success of Nystrom-based and related randomized low-rank methods

Conflicting claims about uniform versus leverage-based sampling:

- Some claim “ML matrices have low coherence” based on one ML paper
- Contrasts with proven importance of leverage scores is genetics, astronomy, and internet applications

High-quality numerical implementations of random projection and random sampling algorithms now exist:

- For L2 regression, L1 regression, low-rank matrix approximation, etc. in RAM, parallel environments, distributed environments, etc.

$$\left(\quad G \quad \right) \approx \left(\quad \tilde{G} \quad \right) = \left(C \right) \left(W \right)^{+} \left(\quad C^T \quad \right)$$

Some basics

Leverage scores:

- Diagonal elements of projection matrix onto the best rank-k space
- Key structural property needed to get 1±ε approximation of general matrices

Spectral, Frobenius, and Trace norms:

- Matrix norms that equal {∞,2,1}-norm on the vector of singular values

$$||\mathbf{A}||_2 \leq ||\mathbf{A}||_F \leq ||\mathbf{A}||_* \leq \sqrt{n}||\mathbf{A}||_F \leq n||\mathbf{A}||_2$$

Basic SPSD Sketching Model:

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

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

Then $\mathbf{CW^+C^T}$ is a low-rank approximation to $\mathbf{A}$ with rank at most ℓ.

Strategy for improved theory

Decouple the randomness from the vector space structure

- This used previously with least-squares and low-rank CSSP approximation

This permits much finer control in the application of randomization

- Much better worst-case theory
- Easier to map to ML and statistical ideas
- Has led to high-quality numerical implementations of LS and low-rank algorithms
- Much easier to parameterize problems in ways that are more natural to numerical analysts, scientific computers, and software developers

This implicitly looks at the "square root" of the SPSD matrix

Main structural result

Gittens and Mahoney (2013)

Theorem. Let $\mathbf{A}$ be an $n \times n$ SPSD matrix with eigenvalue decomposition $\mathbf{A} = \mathbf{U\Sigma U}^T$, where $\mathbf{U}_1$ is top k eigenvalues, $\mathbf{\Omega}_1 = \mathbf{U}_1^T\mathbf{S}$ etc., and let $\mathbf{S}$ be a sampling matrix of size $n \times \ell$. Then when $\mathbf{C} = \mathbf{AS}$ and $\mathbf{W} = \mathbf{S}^T\mathbf{AS}$, the corresponding low-rank SPSD approximation satisfies

$$\begin{aligned}
\|\mathbf{A} - \mathbf{CW}^+\mathbf{C}^T\|\|\|_2 &\leq \|\mathbf{\Sigma}_2\|_2 + \|\mathbf{\Sigma}_2^{1/2}\mathbf{\Omega}_2\mathbf{\Omega}_1^+\|_2^2 \\
\|\mathbf{A} - \mathbf{CW}^+\mathbf{C}^T\|_F &\leq \|\mathbf{\Sigma}_2\|_F + \sqrt{2}\|\mathbf{\Sigma}_2\mathbf{\Omega}_2\mathbf{\Omega}_1^+\|_F + \|\mathbf{\Sigma}_2^{1/2}\mathbf{\Omega}_2\mathbf{\Omega}_1^+\|_F^2 \\
\|\mathbf{A} - \mathbf{CW}^+\mathbf{C}^T\|_{Tr} &\leq \mathrm{Tr}\mathbf{\Sigma}_2 + \|\mathbf{\Sigma}_2^{1/2}\mathbf{\Omega}_2\mathbf{\Omega}_1^+\|_F^2,
\end{aligned}$$

assuming $\mathbf{\Omega}_1$ has full row rank.

Algorithmic applications (1 of 2)

Gittens and Mahoney (2013)

Lemma. Let $\mathbf{S}$ be a sampling matrix of size $n \times \ell$ corresponding to a leverage-based probability distribution derived from the top k-dimensional eigenspace of $\mathbf{A}$ s.t. for some $\beta \in (0, 1]$. If $\ell \geq 3200(\beta\varepsilon^2)^{-1} k \ln(4k/(\beta\delta))$, then w.p. $1 - \delta$ the corresponding low-rank SPSD approximation satisfies

$$\begin{aligned} \|\mathbf{A} - \mathbf{C}\mathbf{W}^+\mathbf{C}^T\|_2 &\leq \|\mathbf{A} - \mathbf{A}_k\|_2 + \varepsilon^2 \|\mathbf{A} - \mathbf{A}_k\|_{Tr}, \\ \|\mathbf{A} - \mathbf{C}\mathbf{W}^+\mathbf{C}^T\|_F &\leq (1 + \sqrt{2}\varepsilon)\|\mathbf{A} - \mathbf{A}_k\|_F + \varepsilon^2 \|\mathbf{A} - \mathbf{A}_k\|_{Tr}, \\ \|\mathbf{A} - \mathbf{C}\mathbf{W}^+\mathbf{C}^T\|_{Tr} &\leq (1 + \varepsilon^2)\|\mathbf{A} - \mathbf{A}_k\|_{Tr}. \end{aligned}$$

Similar bounds for uniform sampling, except that need to sample proportional to the coherence (the largest leverage score).

Algorithmic applications (2 of 2)

Gittens and Mahoney (2013)

Lemma. Let $\mathbf{S} = \sqrt{\frac{n}{\ell}}\mathbf{DFR}$ be a structured random projection of size $n \times \ell$. If $\ell \geq 24\varepsilon^{-1}[\sqrt{k} + \sqrt{8\ln(8n/\delta)}]^2 \ln(8k/\delta)$, then w.p. $1-\delta$ the corresponding low-rank SPSD approximation satisfies

$$\begin{aligned}
\|\mathbf{A} - \mathbf{CW}^{+}\mathbf{C}^T\|_2 &\leq \left(1 + \frac{1}{1-\sqrt{\varepsilon}} \cdot \left(5 + \frac{16\ln(n/\delta)^2}{\ell}\right)\right)\|\mathbf{A} - \mathbf{A}_k\|_2 \\
&+ \frac{2\ln(n/\delta)}{(1-\sqrt{\varepsilon})\ell}\|\mathbf{A} - \mathbf{A}_k\|_{Tr}, \\
\|\mathbf{A} - \mathbf{CW}^{+}\mathbf{C}^T\|_F &\leq (1 + \sqrt{44\varepsilon})\|\mathbf{A} - \mathbf{A}_k\|_F + 22\varepsilon\|\mathbf{A} - \mathbf{A}_k\|_{Tr}, \\
|\mathbf{A} - \mathbf{CW}^{+}\mathbf{C}^T\|_{Tr} &\leq (1 + 22\varepsilon)\|\mathbf{A} - \mathbf{A}_k\|_{Tr}.
\end{aligned}$$

Similar bounds for Gaussian-based random projections.

Data considered (1 of 2)

Name	Description	n	d	%nnz
Laplacians				
HEP	arXiv High Energy Physics collaboration graph	9877	NA	0.06
GR	arXiv General Relativity collaboration graph	5242	NA	0.12
Enron	subgraph of the Enron email graph	10000	NA	0.22
Gnutella	Gnutella peer to peer network on Aug. 6, 2002	8717	NA	0.09
Linear Kernels				
Dexter	bag of words	2000	20000	83.8
Protein	derived feature matrix for S. cerevisiae	6621	357	99.7
SNPs	DNA microarray data from cancer patients	5520	43	100
Gisette	images of handwritten digits	6000	5000	100
Dense RBF Kernels				
AbaloneD	physical measurements of abalones	4177	8	100
WineD	chemical measurements of wine	4898	12	100
Sparse RBF Kernels				
AbaloneS	physical measurements of abalones	4177	8	82.9/48.1
WineS	chemical measurements of wine	4898	12	11.1/88.0

Table 1: The data sets used in our empirical evaluation. The %nnz for the Sparse RBF Kernels depends on the σ parameter.

Data considered (2 of 2)

Name	%nnz	$\left\lceil \frac{\|\|\mathbf{A}\|\|_F^2}{\|\|\mathbf{A}\|\|_2^2} \right\rceil$	k	$\frac{\lambda_{k+1}}{\lambda_k}$	$100\frac{\|\|\mathbf{A}-\mathbf{A}_k\|\|_F}{\|\|\mathbf{A}\|\|_F}$	$100\frac{\|\|\mathbf{A}-\mathbf{A}_k\|\|_*}{\|\|\mathbf{A}\|\|_*}$	k^{th}-lrgst lev
HEP	0.06	3078	20	0.998	7.8	0.4	0.261
HEP	0.06	3078	60	0.998	13.2	1.1	0.278
GR	0.12	1679	20	0.999	10.5	0.74	0.286
GR	0.12	1679	60	1	17.9	2.16	0.289
Enron	0.22	2588	20	0.997	7.77	0.352	0.492
Enron	0.22	2588	60	0.999	12.0	0.94	0.298
Gnutella	0.09	2757	20	1	8.1	0.41	0.381
Gnutella	0.09	2757	60	0.999	13.7	1.20	0.340
Dexter	83.8	176	8	0.963	14.5	.934	0.067
Protein	99.7	24	10	0.987	42.6	7.66	0.008
SNPs	100	3	5	0.928	85.5	37.6	0.002
Gisette	100	4	12	0.90	90.1	14.6	0.005
AbaloneD (dense, $\sigma = .15$)	100	41	20	0.992	42.1	3.21	0.087
AbaloneD (dense, $\sigma = 1$)	100	4	20	0.935	97.8	59	0.012
WineD (dense, $\sigma = 1$)	100	31	20	0.99	43.1	3.89	0.107
WineD (dense, $\sigma = 2.1$)	100	3	20	0.936	94.8	31.2	0.009
AbaloneS (sparse, $\sigma = .15$)	82.9	400	20	0.989	15.4	1.06	0.232
AbaloneS (sparse, $\sigma = 1$)	48.1	5	20	0.982	90.6	21.8	0.017
WineS (sparse, $\sigma = 1$)	11.1	116	20	0.995	29.5	2.29	0.2
WineS (sparse, $\sigma = 2.1$)	88.0	39	20	0.992	41.6	3.53	0.098

Table 1: Summary statistics for data sets used in our empirical evaluation.

Weakness of previous theory (1 of 2)

Drineas and Mahoney (COLT 2005, JMLR 2005):

- If sample $\Omega(k\, \varepsilon^{-4} \log(1/\delta))$ columns according to diagonal elements of A, then

$$\|\mathbf{A}-\mathbf{C}\mathbf{W}^{+}\mathbf{C}^{T}\|_{2,F} \leq \|\mathbf{A}-\mathbf{A}_k\|_{2,F} + \varepsilon \sum_{k=1}^{n} (\mathbf{A})_{ii}^{2}$$

Kumar, Mohri, and Talwalker (ICML 2009, JMLR 2012):

- If sample $\Omega(\tau\, k \log(k/\delta))$ columns uniformly, where $\tau \approx$ coherence and A has exactly rank k, then can reconstruct A, i.e., $\mathbf{A} = \mathbf{C}\mathbf{W}^{+}\mathbf{C}^{T}$

Gittens (arXiv, 2011):

- If sample $\Omega(\mu\, k \log(k/\delta))$ columns uniformly, where μ = coherence, then

$$\begin{aligned}
\|\mathbf{A}-\mathbf{C}\mathbf{W}^{+}\mathbf{C}^{T}\|_2 &\leq \|\mathbf{A}-\mathbf{A}_k\|_2 \left(1+\frac{2n}{\ell}\right) \\
\|\mathbf{A}-\mathbf{C}\mathbf{W}^{+}\mathbf{C}^{T}\|_2 &\leq \|\mathbf{A}-\mathbf{A}_k\|_2 + \frac{2}{\delta} \cdot \|\mathbf{A}-\mathbf{A}_k\|_{Tr}
\end{aligned}$$

So weak that these results aren't even a qualitative guide to practice

Weakness of previous theory (2 of 2)

source, sketch	pred./obs. spectral error	pred./obs. Frobenius error	pred./obs. trace error
Protein, $k = 10$			
DM05 nonuniform Nyström	119.2	18.6	–
BW09 uniform Nyström	–	–	3.6
KMT12 uniform Nyström	33.4	20.5	–
GM13 Leverage-based Lemma	42.5	6.9	2.0
GM13 Fourier-based Lemma	297.5	21.7	3.1
GM13 Gaussian-based Lemma	3.8	3.3	1.8
GM13 uniform Nyström Lemma	86.3	91.3	8
AbaloneD, $\sigma = .15, k = 20$			
DM05 nonuniform Nyström	349.9	42.5	–
BW09 uniform Nyström	–	–	2.0
KMT12 uniform Nyström	62.9	46.7	–
GM13 Leverage-based Lemma	235.3	14.6	1.3
GM13 Fourier-based Lemma	139.4	36.9	1.7
GM13 Gaussian-based Lemma	5.2	4.7	1.1
GM13 uniform Nyström Lemma	12.9	228.3	5.1
WineS, $\sigma = 1, k = 20$			
DM05 nonuniform Nyström	422.5	41.0	–
BW09 uniform Nyström	–	–	2.1
KMT12 uniform Nyström	72.8	44.2	–
GM13 Leverage-based Lemma	244.9	13.4	1.2
GM13 Fourier-based Lemma	186.7	36.8	1.7
GM13 Gaussian-based Lemma	6.6	4.7	1.2
GM13 uniform Nyström Lemma	13.7	222.6	5.1

Approximating the leverage scores (for very rectangular matrices)

Drineas, Magdon-Ismail, Mahoney, and Woodruff (2012)

Input: $\mathbf{A} \in R^{n \times d}$ (with $n \gg d$ and SVD $\mathbf{A} = \mathbf{U\Sigma V}^T$), and $\epsilon \in (0, 1/2]$.

- Let $\mathbf{\Pi}_1 \in R^{r_1 \times n}$ be an SRFT with $r_1 = \Omega(\epsilon^{-2}(\sqrt{d} + \sqrt{\ln n})^2 \ln d)$.
- Compute $\mathbf{\Pi}_1\mathbf{A} \in R^{r_1 \times d}$ and its QR factorization $\mathbf{\Pi}_1\mathbf{A} = \mathbf{QR}$.
- Let $\mathbf{\Pi}_2 \in R^{d \times r_2}$ be a matrix of i.i.d. standard Gaussian random variables, where $r_2 = \Omega\left(\epsilon^{-2} \ln n\right)$.
- Construct the product $\mathbf{\Omega} = \mathbf{AR}^{-1}\mathbf{\Pi}_2$.
- For $i = 1, \ldots, n$ compute $\tilde{\ell}_i = ||\Omega_{(i)}||_2^2$.

Output: $\tilde{\ell}_i, i = 1, \ldots, n$, approximations to the leverage scores of $\mathbf{A}$.

• This algorithm returns relative-error (1±ε) approximations to all the leverage scores of an arbitrary tall matrix in o(nd2) time, i.e., in time

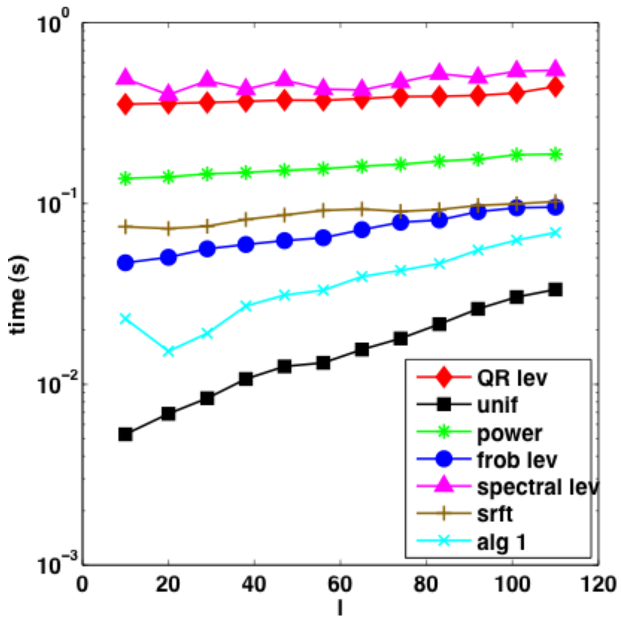
$$O(nd\ln(\sqrt{d} + \sqrt{\ln n}) + nd\epsilon^{-2}\ln n + d^2\epsilon^{-2}(\sqrt{d} + \sqrt{\ln n})^2 \ln d).$$

An aside: Timing for fast approximating leverage scores of rectangular matrices

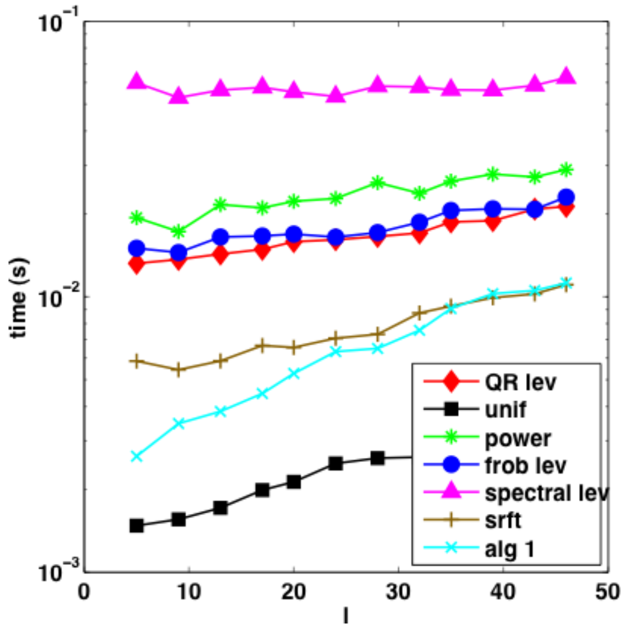
Gittens and Mahoney (2013)

Running time is comparable to underlying random projection

- (Can solve the subproblem directly; or, as with Blendenpik, use it to precondition to solve LS problems of size ≥ thousands-by-hundreds faster than LAPACK.)



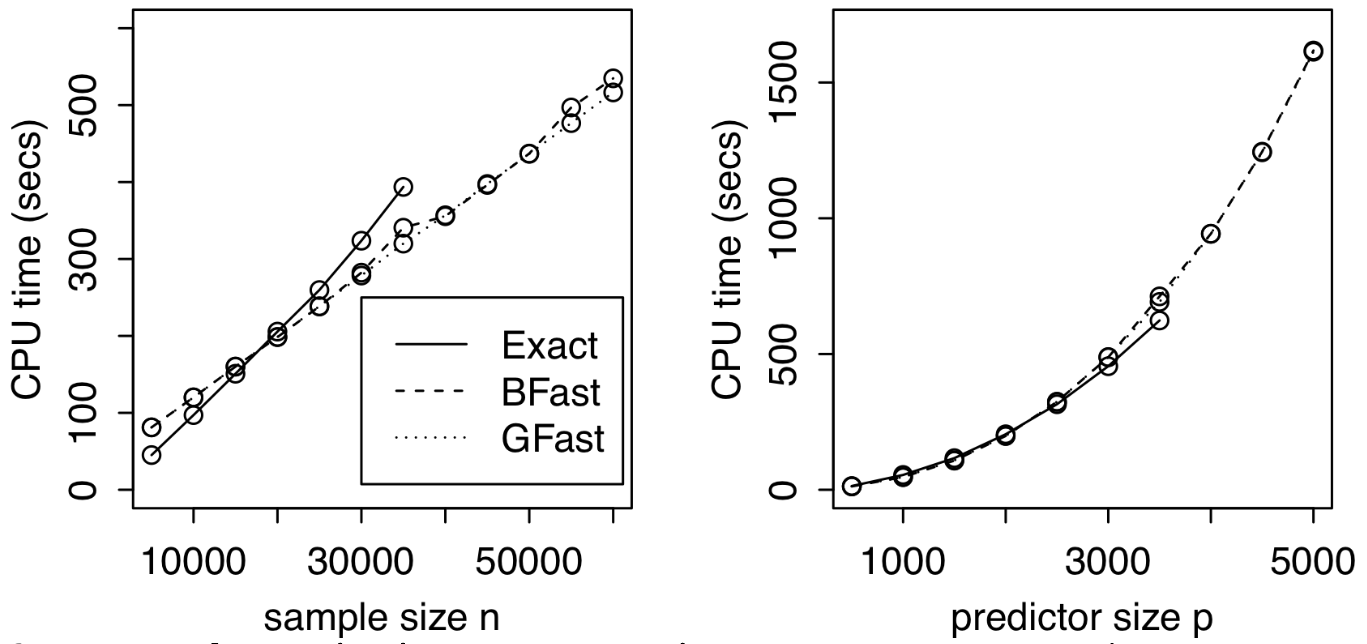
Protein k=10;



SNPs(k=5)

Running time results (for a vanilla implementation in R)

"A statistical perspective on algorithmic leveraging," Ma, Mahoney, and Yu 2013



CPU time for calculating exact leverage scores and approximate leverage scores using the Bfast (Binary Projections) and Gfast (Gaussian Projections), i.e., "slow" versions of the "fast" algorithm of DMMW12.

Left panel is for varying sample size n for fixed predictor size p=500.

Right panel is for varying predictor size p for fixed sample size n=20000.

Summary of running time issues

Running time of exact leverage scores:

- worse than uniform sampling, SRFT-based, & Gaussian-based projections

Running time of approximate leverage scores:

- can be much faster than exact computation
- with q=0 iterations, time comparable to SRFT or Gaussian projection time
- with q>0 iterations, time depends on details of stopping condition

The leverage scores:

- with q=0 iterations, the actual leverage scores are poorly approximated
- with q>0 iterations, the actual leverage scores are better approximated
- reconstruction quality is often no worse, and is often *better*, when using approximate leverage scores

On "tall" matrices:

- running time is comparable to underlying random projection
- can use the coordinate-biased sketch thereby obtained as preconditioner for overconstrained L2 regression, as with Blendenpik or LSRN

Roadmap of the tutorial

Focus: sketching matrices by (i) sampling rows/columns and (ii) via "random projections."

Machinery: (i) Approximating matrix multiplication and (ii) Decoupling "randomization" from "matrix perturbation."

Overview of the tutorial:

(i) Motivation (computational efficiency, interpretability)

(ii) Approximating matrix multiplication

(iii) From matrix multiplication to CX/CUR factorizations and the SVD

(iv) Improvements and recent progress

(v) Algorithmic approaches to least-squares problems

(vi) Statistical perspectives on least-squares algorithms

(vii) Theory and practice of: extending these ideas to kernels and SPSD matrices

(viii) Theory and practice of: implementing these ideas in large-scale settings

Parallel environments and how they scale

Shared memory

- cores: [10, 10^3]*
- memory: [100GB, 100TB]

Message passing

- cores: [200, 10^5]**
- memory: [1TB, 1000TB]
- CUDA cores: [5×10^4, 3×10^6]***
- GPU memory: [500GB, 20TB]

MapReduce

- cores: [40, 10^5]****
- memory: [240GB, 100TB]
- storage: [100TB, 100PB]*****

Distributed computing

- cores: [-, 3×10^5]******

“Traditional” matrix algorithms

For L2 regression:

- *direct methods*: QR, SVD, and normal equation ($O(mn^2 + n^2)$ time)
  - Pros: high precision & implemented in LAPACK
  - Cons: hard to take advantage of sparsity & hard to implement in parallel environments

- *iterative methods*: CGLS, LSQR, etc.
  - Pros: low cost per iteration, easy to implement in some parallel environments, & capable of computing approximate solutions
  - Cons: hard to predict the number of iterations needed

For L1 regression:

- linear programming
- interior-point methods (or simplex, ellipsoid? methods)
- re-weighted least squares
- first-order methods

Two important notions: leverage and condition

Statistical leverage. (Think: eigenvectors & low-precision solutions.)

- The *statistical leverage scores* of A (assume m>>n) are the diagonal elements of the projection matrix onto the column span of A.
- They equal the L2-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 & high-precision solutions.)

- The *L2-norm condition number* of A is (A) = $\sigma_{max}(A)/\sigma_{min}(A)$.
- $\kappa(A)$ bounds the number of iterations
    - for ill-conditioned problems (e.g., $\kappa(A) \cong 10^6 >> 1$), convergence speed is slow.
- *Computing $\kappa(A)$ is generally as hard as solving the LS problem.*

These are for the L2-norm. Generalizations exist for the L1-norm.

Meta-algorithm for L2 regression

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

1: Using the L2 statistical leverage scores of A, construct an importance sampling distribution $\{p_i\}_{i=1,\ldots,m}$

2: Randomly sample a small number of constraints according to $\{p_i\}_{i,\ldots,m}$ to construct a subproblem.

3: Solve the L2-regression problem on the subproblem.

Naïve implementation: $1 + \varepsilon$ approximation in $O(mn^2/\varepsilon)$ time. (Ugh.)

"Fast" $O(mn \log(n)/\varepsilon)$ in RAM if

- Hadamard-based projection and sample uniformly
- Quickly compute approximate leverage scores

"High precision" $O(mn \log(n)\log(1/\varepsilon))$ in RAM if:

- use the random projection/sampling basis to construct a preconditioner

Question: can we extend these ideas to parallel-distributed environments?

Meta-algorithm for L1 (& Lp) regression

(Clakson 2005, DDHKM 2008, Sohler and Woodruff 2011, CDMMMW 2012, Meng and Mahoney 2012.)

1: Using the L1 statistical leverage scores of A, construct an importance sampling distribution $\{p_i\}_{i=1,\ldots,m}$

2: Randomly sample a small number of constraints according to $\{p_i\}_{i,\ldots,m}$ to construct a subproblem.

3: Solve the L1-regression problem on the subproblem.

Naïve implementation: $1 + \varepsilon$ approximation in $O(mn^5/\varepsilon)$ time. (Ugh.)

"Fast" in RAM if

• we perform a fast "L1 projection" to uniformize them approximately

• we approximate the L1 leverage scores quickly

"High precision" in RAM if:

• we use the random projection/sampling basis to construct an L1 preconditioner

Question: can we extend these ideas to parallel-distributed environments?

LARGE versus large versus large: extending to parallel/distributed environments

Can we extend these ideas to parallel & distributed environments?

- Yes!!!
- Roughly, use the same meta-algorithm, but minimize communication rather than minimize flops

In the remainder, focus on L2 regression.

- Technical issues, especially for iterations, are very different for L2 regression versus L1/quantile regression
- Talk with me later if you care about L1 regression

LSRN: a fast parallel implementation

Meng, Saunders, and Mahoney (2011, arXiv)

A parallel iterative solver based on normal random projections

- computes unique min-length solution to $\min_x ||Ax-b||_2$
- very over-constrained or very under-constrained *A*
- full-rank or rank-deficient *A*
- *A* can be dense, sparse, or a linear operator
- easy to implement using threads or with MPI, and scales well in parallel environments

LSRN: a fast parallel implementation

Meng, Saunders, and Mahoney (2011, arXiv)

Algorithm:

• Generate a γn x m matrix with i.i.d. Gaussian entries G

• Let N be R^{-1} or $V \Sigma^{-1}$ from QR or SVD of GA

• Use LSQR or Chebyshev Semi-Iterative (CSI) method to solve the preconditioned problem $\min_y ||ANy-b||_2$

Things to note:

• Normal random projection: embarassingly parallel

• Bound $\kappa(A)$: strong control on number of iterations

• CSI particularly good for parallel environments: doesn't have vector inner products that need synchronization b/w nodes

LSRN: Solving real-world problems

Meng, Saunders, and Mahoney (2011, arXiv)

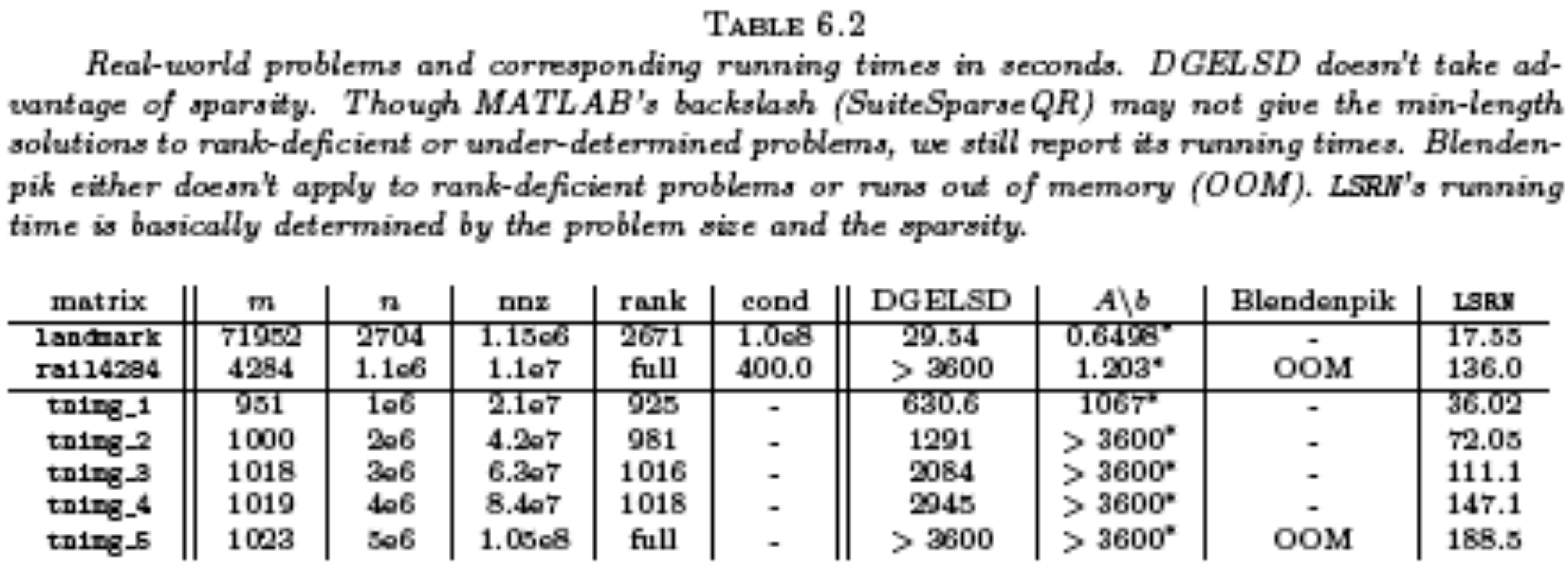


TABLE 6.2

Real-world problems and corresponding running times in seconds. DGELSD doesn't take advantage of sparsity. Though MATLAB's backslash (SuiteSparseQR) 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 basically determined by the problem size and the sparsity.

matrix	m	n	nnz	rank	cond	DGELSD	$A\backslash b$	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	1.05e8	full	-	> 3600	> 3600*	OOM	188.5

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

Chebyshev semi-iterative (CSI)

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

LSRN: on Amazon EC2 cluster

Meng, Saunders, and Mahoney (2011, arXiv)

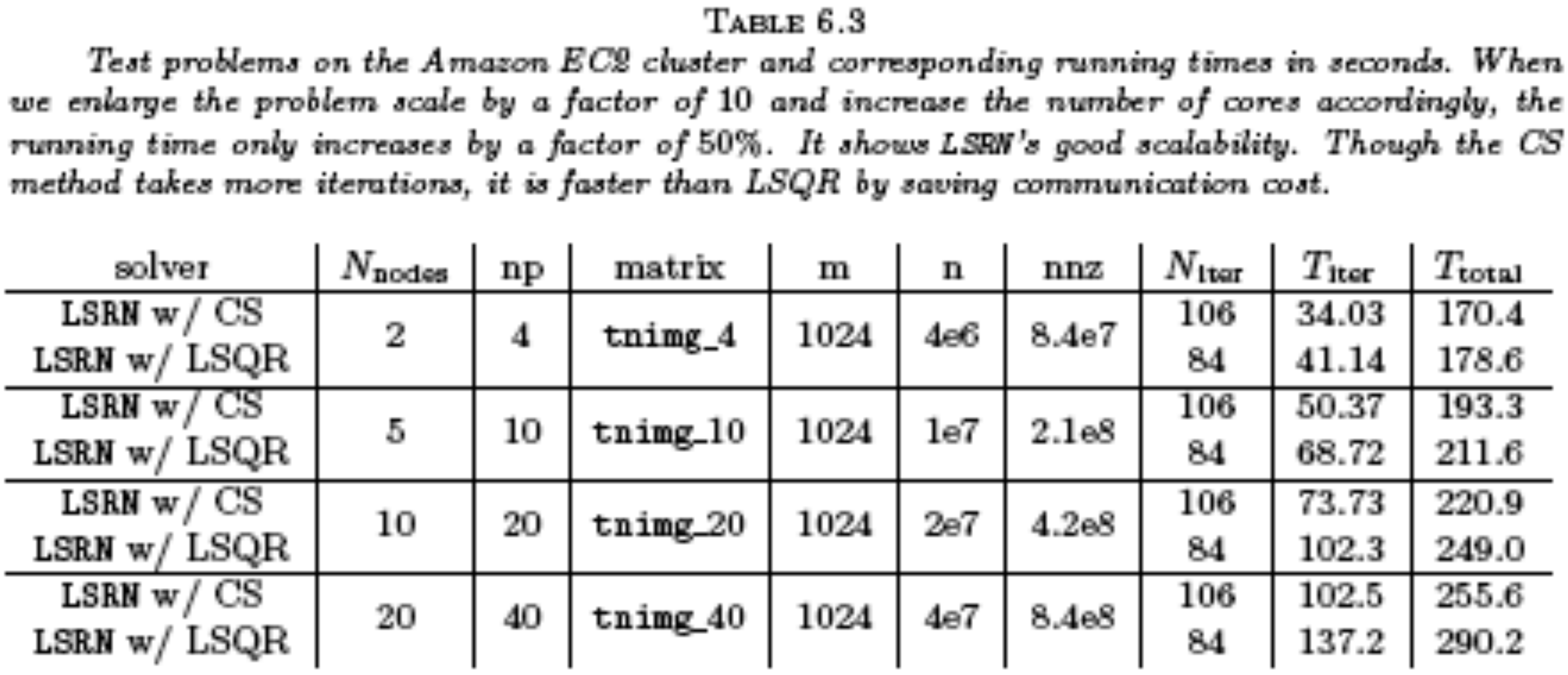


TABLE 6.3

Test problems on the Amazon EC2 cluster and corresponding running times in seconds. When we enlarge the problem scale by a factor of 10 and increase the number of cores accordingly, the running time only increases by a factor of 50%. It shows LSRN's good scalability. Though the CS method takes more iterations, it is faster than LSQR by saving communication cost.

solver	N_{nodes}	np	matrix	m	n	nnz	N_{iter}	T_{iter}	T_{total}
LSRN w/ CS	2	4	tnimg_4	1024	4e6	8.4e7	106	34.03	170.4
LSRN w/ LSQR							84	41.14	178.6
LSRN w/ CS	5	10	tnimg_10	1024	1e7	2.1e8	106	50.37	193.3
LSRN w/ LSQR							84	68.72	211.6
LSRN w/ CS	10	20	tnimg_20	1024	2e7	4.2e8	106	73.73	220.9
LSRN w/ LSQR							84	102.3	249.0
LSRN w/ CS	20	40	tnimg_40	1024	4e7	8.4e8	106	102.5	255.6
LSRN w/ LSQR							84	137.2	290.2

Additional topics not covered ...

Theory/practice of L1/quantile regression:

• Cauchy transform, ellipsoidal rounding, etc. to get low-precision soln

• couple with randomized interior point cutting plane method to get moderate-precision solutions on a terabyte of data in Hadoop

Theory/practice of “input-sparsity” regression algorithms:

• input-sparsity time matrix multiplication result -> input-sparsity time L2 regression, low-rank approximation, leverage score algorithms

• nearly-input-sparsity time Lp regression algorithms via input-sparsity time low-distortion embeddings

Conclusions to Part II

Least-squares regression:

- faster sampling/projection in theory and implementation
- importance of decoupling randomness from vector space structure

Statistical perspective:

- better practical results without sacrificing worst-case quality

Revisiting the Nystrom method:

- the devil is in the details, if we want to make these algorithms useful in real large-scale systems

Implementing in parallel/distributed environments:

- the same meta-algorithms work, but highlights the limits of theoretically-useful models, and suggests future directions

All of these suggest future directions ...

Conclusions on "RandNLA"

Many many modern massive data sets are well-modeled by matrices:

- but existing algorithms were not designed for them

Randomization is a powerful tool for:

- the design of algorithms with better worst-case guarantees
- the design of algorithms with better statistical properties
- the design of algorithms for large-scale architectures

Great model/proof-of-principle for "bridging the gap":

- between TCS and NLA and ML
- useful theory and theoretically-fruitful practice arises