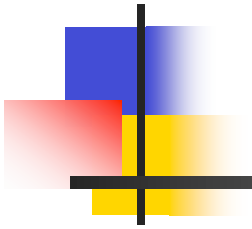
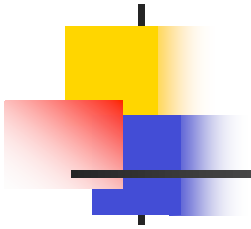


Fast Approximation of Matrix Coherence and Statistical Leverage

Michael W. Mahoney

Stanford University

(For more info, see:
http:// cs.stanford.edu/people/mmahoney/
or Google on "Michael Mahoney")



Statistical leverage

Def: Let A be n x d matrix, with n >> d, i.e., a *tall* matrix.

• The statistical leverage scores are the diagonal elements of the projection matrix onto the left singular vectors of A.

• The coherence of the rows of A is the largest score.

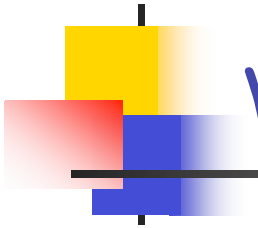
Basic idea: Statistical leverage measures:

• correlation b/w singular vectors of a matrix and the standard basis

• *how much* influence/leverage a row has on the the best LS fit

• *where* in the high-dimensional space the (singular value) information of A is being sent, independent of *what* that information is

• the extent to which a data point is an outlier



Who cares?

Statistical Data Analysis and Machine Learning

• historical measure of "outlierness" or error

• recently, "Nystrom method" and "matrix reconstruction" typically *assume* that the coherence is uniform/flat

Numerical Linear Algebra

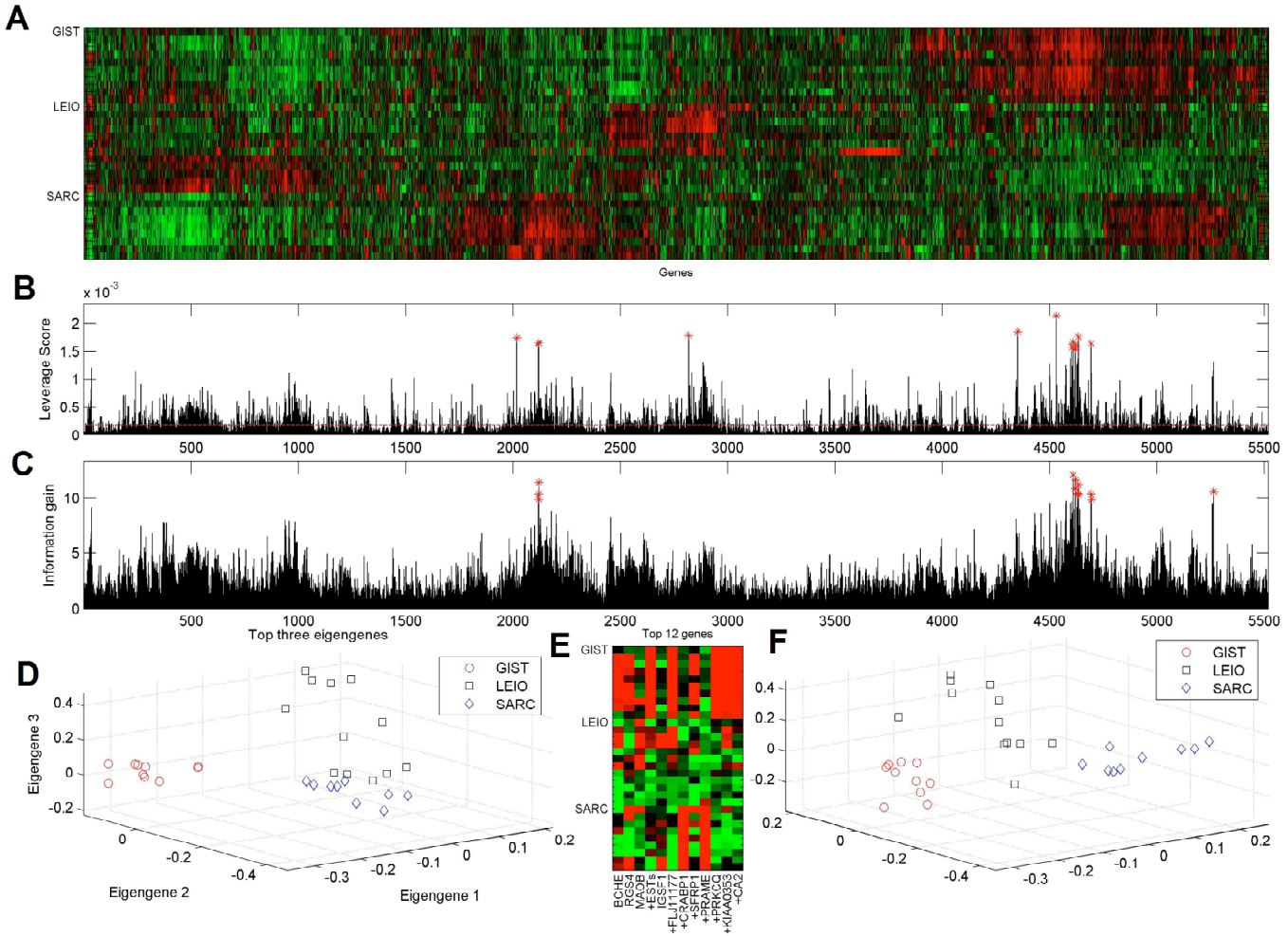
• key bottleneck to get high-quality numerical implementation of randomized matrix algorithms

Theoretical Computer Science

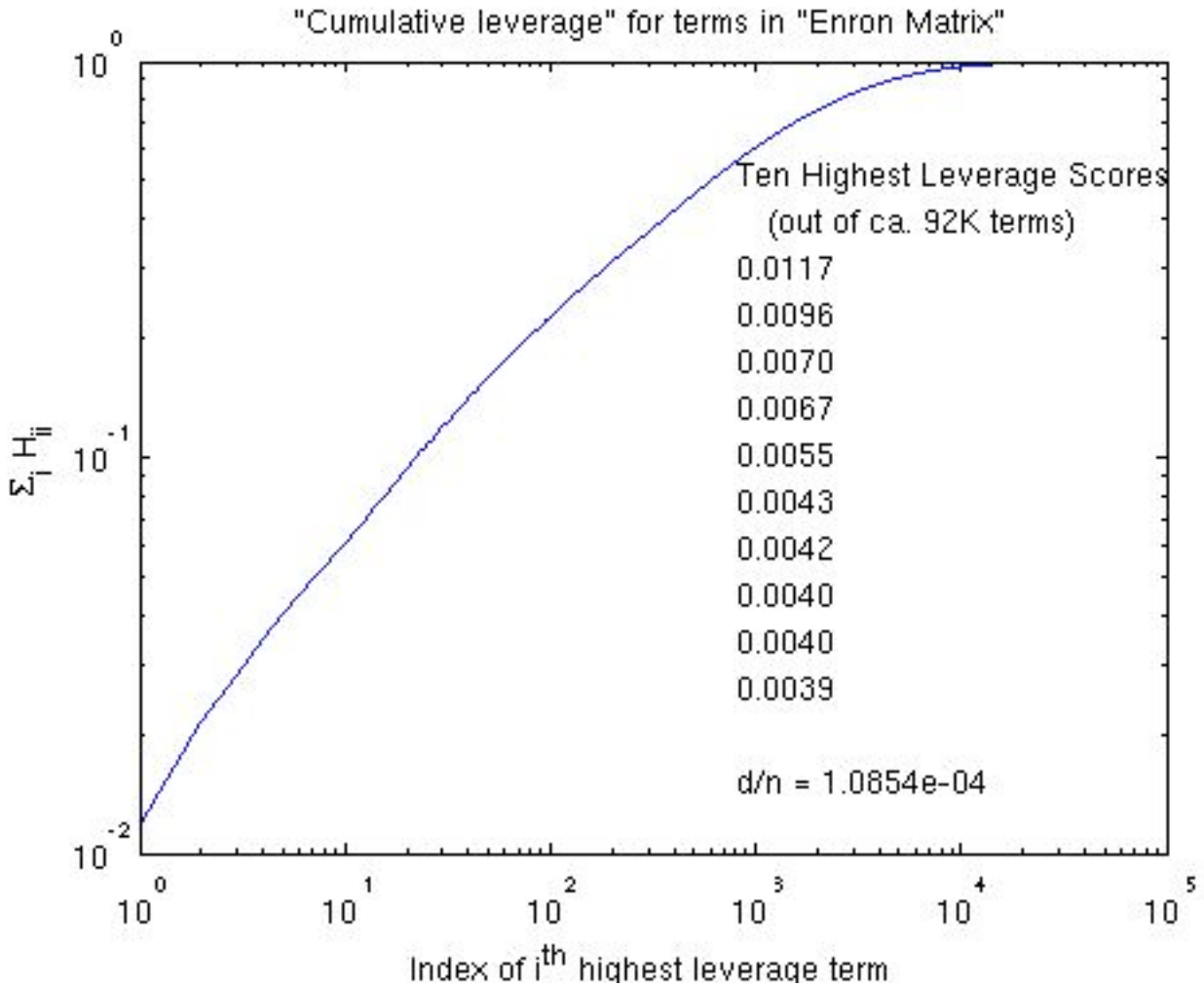
• key structural nonuniformity to deal with in worst-case analysis

• best random sampling algorithms use them as importance sampling distribution and best random projection algorithms uniformize them

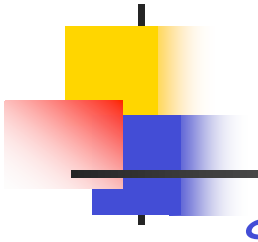
Statistical leverage and DNA SNP data



Statistical leverage and term-document data



Note: often the most "interesting" or "important" data points have highest leverage scores---especially when use L2-based models for computational (as opposed to statistical) reasons.



Computing statistical leverage scores

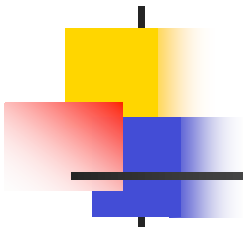
Simple (deterministic) algorithm:

- Compute a basis Q for the left singular subspace, with QR or SVD.
- Compute the Euclidean norms of the *rows* of Q.

Running time is $O(nd^2)$, if n >> d, O(on-basis) time otherwise.

We want *faster*!

- *$o(nd^2)$ or o(on-basis), with no assumptions on input matrix A.*
- Faster in terms of flops of clock time for not-obscenely-large input.
- OK to live with ε-error or to fail with overwhelmingly-small δ probability



Randomized algorithms for matrices (1)

General structure:

• "Preprocess" the matrix -- compute nonuniform importance sampling distribution, *or* perform a random projection/rotation to uniformize

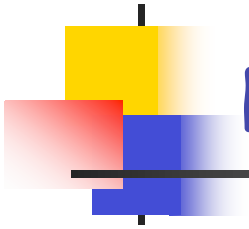
• Draw a random sample of columns/rows, in the original or randomly rotated basis

• "Postprocess" the sample with traditional deterministic NLA algorithm, the solution to which provides the approximation

Two connections to randomization today:

• statistical leverage is the key bottleneck to getting good randomized matrix algorithms (both in theory and in practice)

• the fast algorithm to approximate statistical leverage is a randomized algorithm



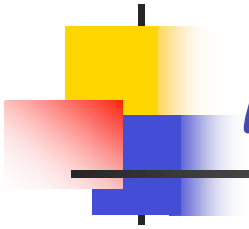
Randomized algorithms for matrices (2)

Maybe unfamiliar, but no need to be afraid:

- (unless you are afraid of flipping a fair coin heads 100 times in a row)
- Lots of work in TCS - but that tends to be *very* cavalier w.r.t. metrics that NLA and Sci Comp tends to care about.

Two recent reviews:

- *Randomized algorithms for matrices and data*, M. W. Mahoney, arXiv:1104.5557 - focuses on "what makes the algorithms work" and interdisciplinary "bridging the gap" issues
- *Finding structure with randomness*, Halko, Martinsson, and Tropp, arXiv:0909.4061 – focuses on connecting with traditional "NLA and scientific computing" issues



Main Theorem

Theorem: Given an n x d matrix A, with n >> d, let P_A be the projection matrix onto the column space of A. Then , there is a randomized algorithm that w.p. ≥ 0.999:

- computes *all* of the n diagonal elements of P_A (i.e., leverage scores) to within relative $(1\pm\varepsilon)$ error;
- computes *all the large* off-diagonal elements of P_A to within additive error;
- runs in *$o(nd^2)$* time*.

*Running time is basically $O(n\ d\ \log(n)/\varepsilon)$, i.e., same as DMMS fast randomized algorithm for over-constrained least squares.

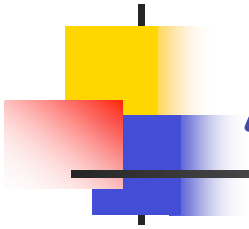
A "classic" randomized algorithm (1of3)

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

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

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)

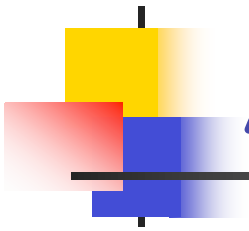
Theorem: Let $\gamma = ||U_A U_A^T b||_2 / ||b||_2$. Then:

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

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)

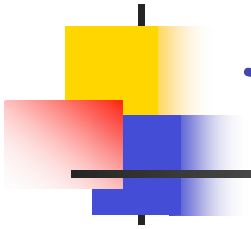
Sufficient condition for relative-error approximation.

For the "preprocessing" matrix X:

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

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

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



Two ways to speed up running time

Random Projection: uniformize leverage scores rapidly

- Apply a Structured Randomized Hadamard Transform to A and b
- Do uniform sampling to construct SHA and SHb
- Solve the induced subproblem
- I.e., call Main Algorithm on preprocessed problem

Random Sampling: approximate leverage scores rapidly

- Described below
- Rapidly approximate statistical leverage scores and call Main Algorithm
- Open problem for a long while

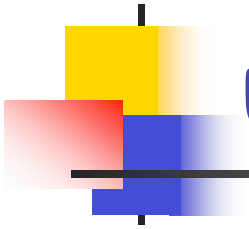
Under-constrained least squares (1of2)

Basic setup: n x d matrix A,n « d, and n-vector b

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

Algorithm:

- For all j ε {1,…,d}, compute $p_j = \frac{1}{d} ||V_{(j)}||_2^2$
- Randomly sample O(n log(n)/ ε) columns fro A, using $\{p_j\}$ *as importance sampling probabilities*.
- Return: $\tilde{x}_{opt} = \text{argmin}_x ||ASS^T x - b||_2 = A^T (AS)^{T\dagger} (AS)^{\dagger} b$



Under-constrained least squares (2of2)

Theorem:

• w.h.p. $||x_{opt} - \tilde{x}_{opt}||_2 \leq \epsilon ||x_{opt}||_2$

Notes:

• Can speed up this $O(nd^2)$ algorithm by doing random projection or by using fast leverage score algorithm below.

• Meng, Saunders, and Mahoney 2011 treat over-constrained and under-constrained, rank deficiencies, etc, in more general and uniform manner for numerical implementations.

Back to approximating leverage scores

View the computation of leverage scores i.t.o an under-constrained LS problem

Recall (A is n x d, n » d):

- $\min_{x \in R^n} ||x^T A - e_i A||_2^2 \quad \rightarrow \quad x^T = e_i A A^\dagger$

But:

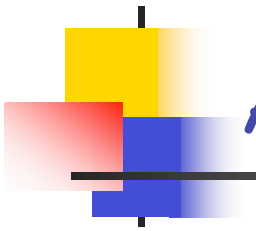
- $p_i = ||e_i U_A||_2^2 = ||e_i U_A U_A^T||_2^2 = ||e_i A A^\dagger||_2^2$

Leverage scores are the norm of a min-length solution of an under-constrained LS problem!

The key idea

$$
\begin{aligned}
p_i &= ||(AA^{\dagger})_{(i)}||_2^2 \\
&\approx ||(A(\Omega_1 A)^{\dagger})_{(i)}||_2^2 \quad \text{where } \Omega_1 \text{ is a fast SRHT} \\
&\approx ||(A(\Omega_1 A)^{\dagger}\Omega_2)_{(i)}||_2^2 \quad \text{where } \Omega_2 \text{ is Rand Proj}
\end{aligned}
$$

Note: this expression is simpler than that for the full under-constrained LS solution since we only need the norm of the solution.



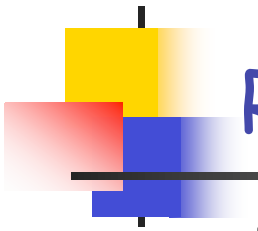
Algorithm and theorem

Algorithm:

- (Ω_1 is r_1 x n, with r_1=O(d log(n)/ε^2), SRHT matrix)
- (Ω_2 is r_1 x r_2, with r_2=O(log(n)/ε^2), RP matrix)
- Compute the n x log(n)/ε^2 matrix $X = A(\Omega_1 A)^+ \Omega_2$, and return the Euclidean norm of each *row* of X.

Theorem:

- $p_i \approx ||X_{(i)}||_2^2$, up to multiplicative $1\pm\varepsilon$, forall i.
- Runs is roughly O(nd log(n)/ε) time.

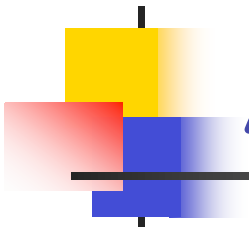


Running time analysis

Running time:

- Random rotation: $\Omega_1 A$ takes $O(nd \log(r_1))$ time
- Pseudoinverse: $(\Omega_1 A)^+$ takes $O(r_1 d^2)$ time
- Matrix multiplication: $A(\Omega_1 A)^+$ takes $O(ndr_1) \geq nd^2$ time - *too much!*
- Another projection: $(\Omega_1 A)^+ \Omega_2$ takes $O(dr_1 r_2)$ time
- Matrix Multiplication: $A(\Omega_1 A)^+ \Omega_2$ takes $O(ndr_2)$ time

Overall, takes $O(nd \log(n))$ time.



An almost equivalent approach

1. Preprocess A to $\Omega_1 A$ with SRHT
2. Find R s.t. $\Omega_1 A = QR$
3. Compute norms of rows of $AR^{-1}\Omega_2$ (a "sketch" which is an "approximately orthogonal" matrix)

- Same quality-of-approximation and running-time bounds
- Previous algorithm amounts to choosing a particular rotation
- Using R^{-1} as a preconditioner is how randomized algorithms for overconstrained LS were implemented numerically

Getting the large off-diagonal elements

Let $X = A(\Omega_1 A)^+ \Omega_2$ or $X = AR^{-1}\Omega_2$.

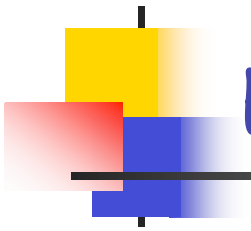
Also true that:

$$\langle X_{(i)}, X_{(j)} \rangle = \langle U_{(i)}, U_{(j)} \rangle \pm O(\epsilon) \|U_{(i)}\| \|U_{(j)}\|$$

So, can use hash function approaches to approximate all off-diagonal elements with

$$\langle U_{(i)}, U_{(j)} \rangle \geq 1/(n \log(nd))$$

to additive error of $\epsilon \|U_{(i)}\| \|U_{(j)}\|$



Extensions to "fat" matrices (1 of 2)

Question: Can we approximate leverage scores relative to best rank-k approximation to A? (Given an arbitrary n x d matrix A and rank parameter k)?

- Ill-posed: Consider $A = I_n$ and $k < n$. (Subpace not even unique, so leverage scores not even well-defined.)
- Unstable: Consider

(Subspace unique, but unstable w.r.t. $\gamma \to 0$.)

Extensions to "fat" matrices (2 of 2)

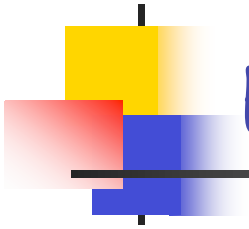
Define: S as set of matrices near the best rank-k approximation:

$$S = \{X \in R^{n \times d} : \text{rank}X = k; ||A - X|| \leq (1 + \epsilon)||A - A_k||\}$$

(Results different if norm is spectral or Frobenius.)

Algorithm:

- Construct compact sketch of A with random projection (several recent variants).
- Use left singular vectors of sketch to compute scores for some matrix in S.



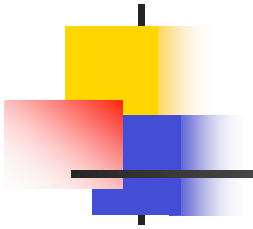
Extensions to streaming environments

Data "streams" by, and can only keep a "small" sketch

But still can compute:

- Rows with large leverage scores and their scores
- Entropy and number of nonzero leverage scores
- Sample of rows according to leverage score probs

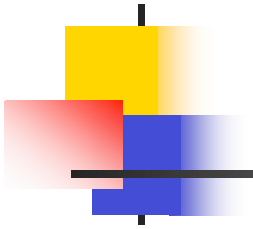
Use hash functions, linear sketching matrices, etc. from streaming literature.



Conclusions

Statistical leverage scores ...

- measure correlation of the dominant subspace with the canonical basis
- have a natural statistical interpretation in terms of outliers
- define "bad" examples for Nystrom method, matrix completion.
- define the key non-uniformity structure for improved worst-case randomized matrix algorithms, e.g., relative-error CUR algorithms, *and* for making TCS randomized matrix useful for NLA and scientific computing
- take $O(nd^2)$, i.e., O(orthonormal-basis), time to compute



Conclusions

... can be computed to $1\pm\varepsilon$ accuracy in $o(nd^2)$ time

- relates the leverage scores to an underdetermined LS problem
- running time comparable to DMMS-style fast approximation algorithms for over-determined LS problem
- also provides large off-diagonal elements in same time
- better numerical understanding of fast randomized matrix algorithms