



Randomized algorithms for matrices and data

Michael W. Mahoney

Stanford University

November 2011

(For more info, see: <http://cs.stanford.edu/people/mmahoney>)



Matrix computations

Eigendecompositions, QR, SVD, least-squares, etc.

Traditional algorithms:

- compute “exact” answers to, say, 10 digits as a black box
- assume the matrix is in RAM and minimize flops

But they are NOT well-suited for:

- with missing or noisy entries
- problems that are very large
- distributed or parallel computation
- when communication is a bottleneck
- when the data must be accessed via “passes”



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

*Today, focus on **low-rank matrix approximation** and **least-squares approximation**: ubiquitous, fundamental, and at the center of recent developments*



The general idea ...

- **Randomly sample** columns/rows/entries of the matrix, with carefully-constructed *importance sampling probabilities*, to form a randomized sketch
- Preprocess the matrix with **random projections**, to form a randomized sketch by sampling columns/rows *uniformly*
- Use the sketch to compute an **approximate solution** to the original problem **w.h.p.**
- **Resulting sketches** are “similar” to the original matrix in terms of singular value and singular vector structure, e.g., w.h.p. are bounded distance from the original matrix



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!



History of NLA

≤ 1940s: Prehistory

- Close connections to data analysis, noise, statistics, randomization

1950s: Computers

- Banish randomness & downgrade data (except scientific computing)

1980s: NLA comes of age - high-quality codes

- QR, SVD, spectral graph partitioning, etc. (written for HPC)

1990s: Lots of new DATA

- LSI, PageRank, NCuts, etc., etc., etc. used in ML and Data Analysis

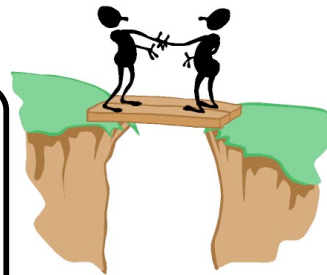
2000s: New data problems force new approaches ...



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!



Statistical leverage, coherence, etc.

Definition: Given a “tall” $n \times d$ matrix A , i.e., with $n > d$, let U be the $n \times d$ matrix of left singular vectors, and let the d -vector $U_{(i)}$ be the i^{th} 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 $U_{(i)}^\top U_{(j)} = \langle U_{(i)}, U_{(j)} \rangle$

Note: There are extension of this to:

- “**fat**” **matrices** A , with n, d are large and low-rank parameter k
- L_1 and **other p-norms**



Algorithmic vs. Statistical Perspectives

Lambert (2000)

Computer Scientists

- *Data*: are a **record of everything** that happened.
- *Goal*: process the data to **find interesting patterns** and associations.
- *Methodology*: Develop approximation algorithms under different models of data access since the goal is typically **computationally hard**.

Statisticians (and Natural Scientists)

- *Data*: are a **particular random instantiation** of an underlying process describing unobserved patterns in the world.
- *Goal*: is to **extract information** about the world from noisy data.
- *Methodology*: Make inferences (perhaps about unseen events) by **positing a model** that describes the random variability of the data around the deterministic model.

Human genetics

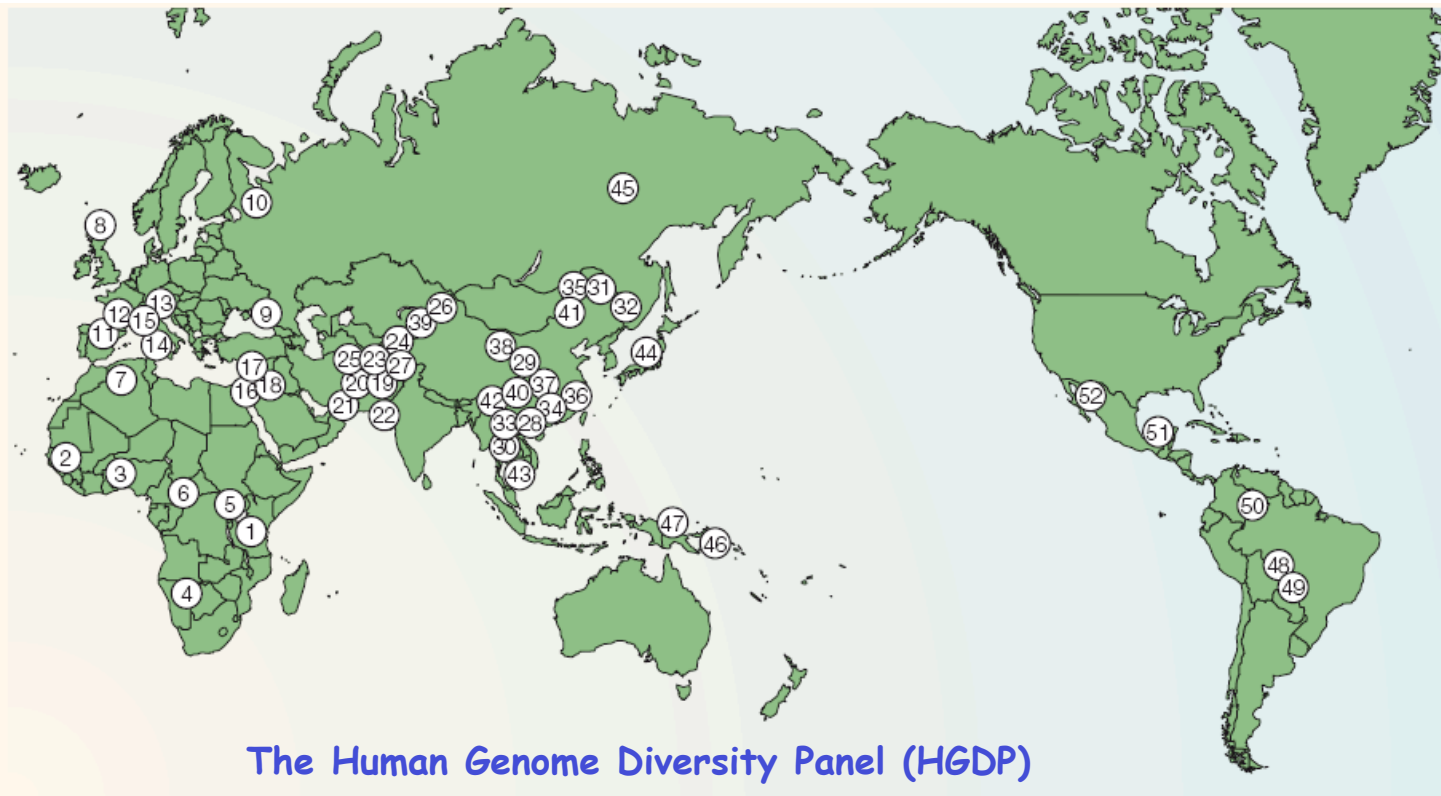
Single Nucleotide Polymorphisms: the most common type of genetic variation in the genome across different individuals.

They are **known** locations at the human genome where **two** alternate nucleotide bases (**alleles**) are observed (out of A, C, G, T).

SNPs

individuals	... AG CT GT GG CT CC CC CC CC AG AG AG AG AG AA CT AA GG GG CC GG AG CG AC CC AA CC AA GG TT AG CT CG CG CG AT CT CT AG CT AG GG GT GA AG ...
	... GG TT TT GG TT CC CC CC CC GG AA AG AG AG AA CT AA GG GG CC GG AA GG AA CC AA CC AA GG TT AA TT GG GG GG TT TT CC GG TT GG GG TT GG AA ...
	... GG TT TT GG TT CC CC CC CC GG AA AG AG AG AA AG CT AA GG GG CC AG AG CG AC CC AA CC AA GG TT AG CT CG CG CG AT CT CT AG CT AG GG GT GA AG ...
	... GG TT TT GG TT CC CC CC CC GG AA AG AG AG AA CC GG AA CC CC AG GG CC AC CC AA CG AA GG TT AG CT CG CG CG AT CT CT AG CT AG GT GT GA AG ...
	... GG TT TT GG TT CC CC CC CC GG AA GG GG GG AA CT AA GG GG CT GG AA CC AC CG AA CC AA GG TT GG CC CG CG CG AT CT CT AG CT AG GG TT GG AA ...
	... GG TT TT GG TT CC CC CG CC AG AG AG AG AG AA CT AA GG GG CT GG AG CC CC CG AA CC AA GT TT AG CT CG CG CG AT CT CT AG CT AG GG TT GG AA ...
	... GG TT TT GG TT CC CC CC CC GG AA AG AG AG AA TT AA GG GG CC AG AG CG AA CC AA CG AA GG TT AA TT GG GG GG TT TT CC GG TT GG GT TT GG AA ...

Matrices including thousands of individuals and hundreds of thousands of SNPs are available.



The Human Genome Diversity Panel (HGDP)

Africans

- 1 Bantu
- 2 Mandenka
- 3 Yoruba
- 4 San
- 5 Mbuti pygmy
- 6 Biaka
- 7 Mozabite

Europeans

- 8 Orcadian
- 9 Adygei
- 10 Russian
- 11 Basque
- 12 French
- 13 North Italian
- 14 Sardinian
- 15 Tuscan

Western Asians

- 16 Bedouin
- 17 Druze
- 18 Palestinian

Central and Southern Asians

- 19 Balochi
- 20 Brahui
- 21 Makrani
- 22 Sindhi
- 23 Pathan
- 24 Burusho
- 25 Hazara
- 26 Uygur
- 27 Kalash

Eastern Asians

- 28 Han (S. China)
- 29 Han (N. China)
- 30 Dai
- 31 Daur
- 32 Hezhen
- 33 Lahu
- 34 Miao
- 35 Oroqen
- 36 She
- 37 Tujia
- 38 Tu
- 39 Xibo
- 40 Yi
- 41 Mongola
- 42 Naxi
- 43 Cambodian
- 44 Japanese
- 45 Yakut

Oceanians

- 46 Melanesian
- 47 Papuan

Native Americans

- 48 Karitiana
- 49 Surui
- 50 Colombian
- 51 Maya
- 52 Pima

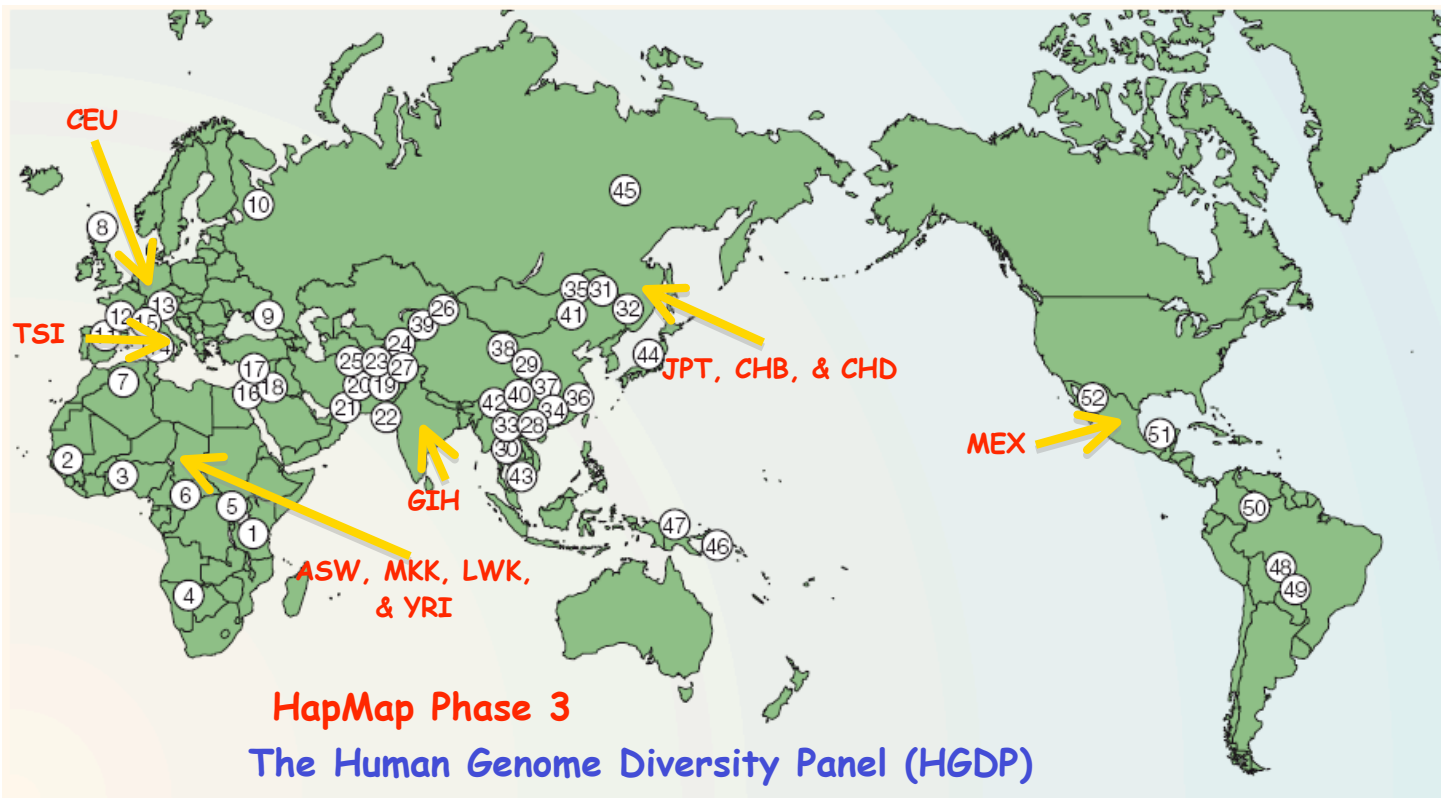
Cavalli-Sforza (2005) *Nat Genet Rev*

Rosenberg et al. (2002) *Science*

Li et al. (2008) *Science*

HGDP data

- 1,033 samples
- 7 geographic regions
- 52 populations



HGDP data

- 1,033 samples
- 7 geographic regions
- 52 populations

HapMap Phase 3 data

- 1,207 samples
- 11 populations

Apply SVD/PCA on the
(joint) HGDP and HapMap
Phase 3 data.

Africans

- 1 Bantu
- 2 Mandenka
- 3 Yoruba
- 4 San
- 5 Mbuti pygmy
- 6 Biaka
- 7 Mozabite

Europeans

- 8 Orcadian
- 9 Adygei
- 10 Russian
- 11 Basque
- 12 French
- 13 North Italian
- 14 Sardinian
- 15 Tuscan

Western Asians

- 16 Bedouin
- 17 Druze
- 18 Palestinian

Central and Southern Asians

- 19 Balochi
- 20 Brahui
- 21 Makrani
- 22 Sindhi
- 23 Pathan
- 24 Burusho
- 25 Hazara
- 26 Uygur
- 27 Kalash

Eastern Asians

- 28 Han (S. China)
- 29 Han (N. China)
- 30 Dai
- 31 Daur
- 32 Hezhen
- 33 Lahu
- 34 Miao
- 35 Oroqen
- 36 She
- 37 Tujia
- 38 Tu
- 39 Xibo
- 40 Yi
- 41 Mongola
- 42 Naxi
- 43 Cambodian
- 44 Japanese
- 45 Yakut

Oceanians

- 46 Melanesian
- 47 Papuan

Native Americans

- 48 Karitiana
- 49 Surui
- 50 Colombian
- 51 Maya
- 52 Pima

Cavalli-Sforza (2005) *Nat Genet Rev*

Rosenberg et al. (2002) *Science*

Li et al. (2008) *Science*

The International HapMap Consortium
(2003, 2005, 2007) *Nature*

Matrix dimensions:

2,240 subjects (rows)

447,143 SNPs (columns)

Dense matrix:

over one billion entries



The Singular Value Decomposition (SVD)

The formal definition:

Given any $m \times n$ matrix A , one can decompose it as:

$$\begin{pmatrix} A \end{pmatrix} = \begin{pmatrix} U \\ m \times \rho \end{pmatrix} \cdot \begin{pmatrix} \Sigma \\ \rho \times \rho \end{pmatrix} \cdot \begin{pmatrix} V \\ \rho \times n \end{pmatrix}^T$$

ρ : rank of A

U (V): **orthogonal** matrix containing the left (right) singular vectors of A .

Σ : diagonal matrix containing $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_\rho$, the singular values of A .

SVD is the "the Rolls-Royce and the Swiss Army Knife of Numerical Linear Algebra."*

*Dianne O'Leary, MADS 2006



Rank- k approximations (A_k)

Truncate the SVD at the top- k terms:

$$\begin{pmatrix} A_k \end{pmatrix} = \begin{pmatrix} U_k \end{pmatrix} \cdot \begin{pmatrix} \Sigma_k \end{pmatrix} \cdot \begin{pmatrix} V_k^T \end{pmatrix}$$

Keep the "most important" k -dim subspace.

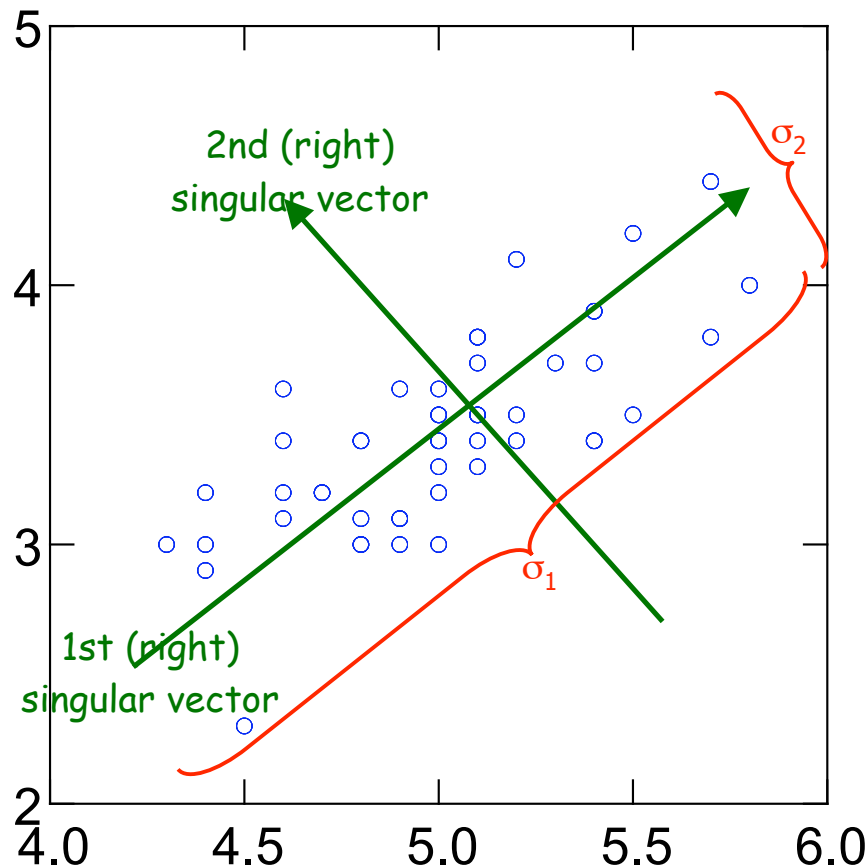
U_k (V_k): orthogonal matrix containing the top k left (right) singular vectors of A .

Σ_k : diagonal matrix containing the top k singular values of A .

Important: Keeping top k singular vectors provides "best" rank- k approximation to A (w.r.t. Frobenius norm, spectral norm, etc.):

$$A_k = \operatorname{argmin}\{ \|A - X\|_{2,F} : \operatorname{rank}(X) \leq k \}.$$

Singular values, intuition



Blue circles are m data points in a 2-D space.

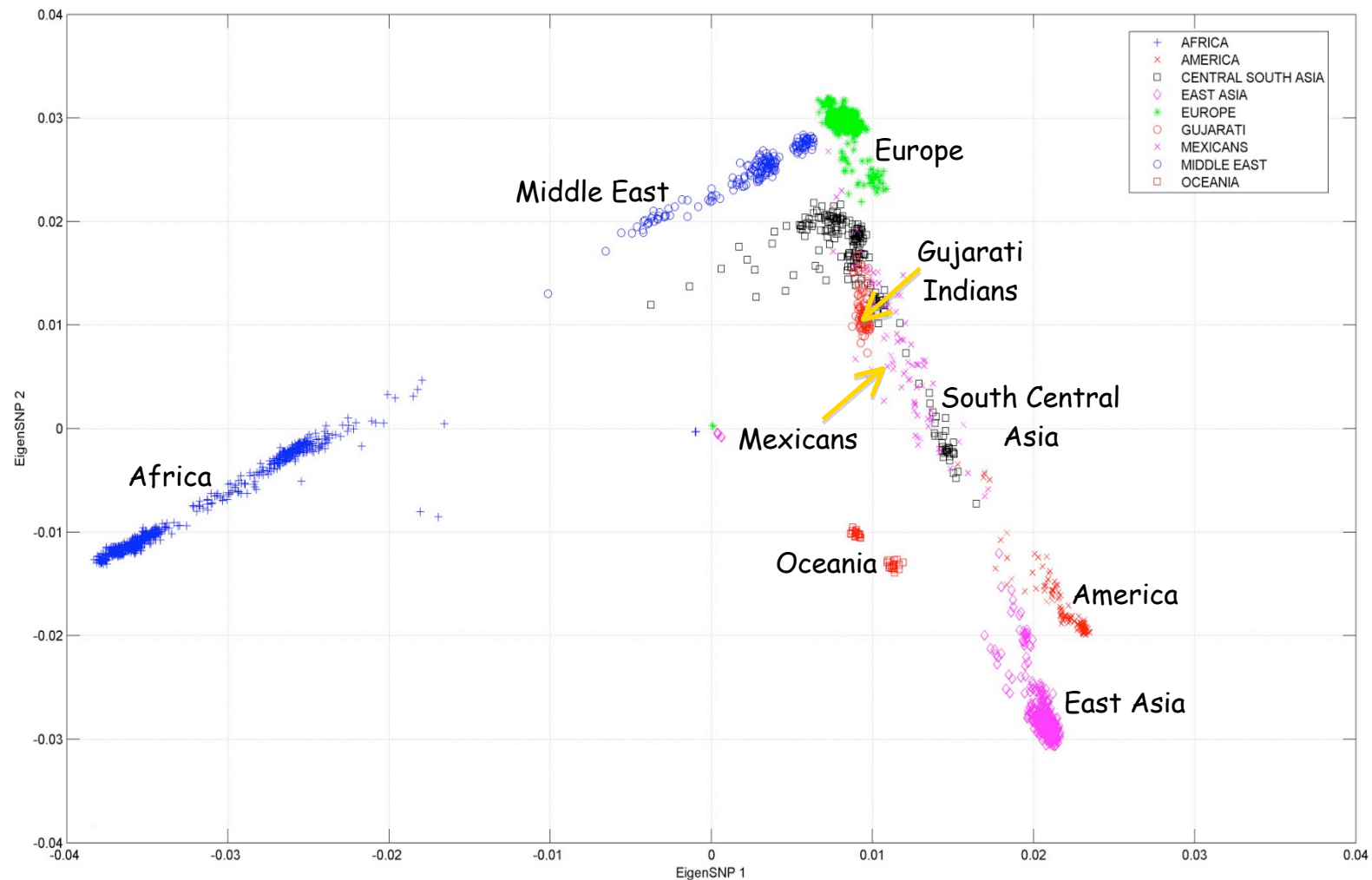
The SVD of the m -by-2 matrix of the data will return ...

$V^{(1)}$: 1st (right) singular vector: direction of maximal variance,

σ_1 : how much of data variance is explained by the first singular vector.

$V^{(2)}$: 2nd (right) singular vector: direction of maximal variance, after removing projection of the data along first singular vector.

σ_2 : measures how much of the data variance is explained by the second singular vector.

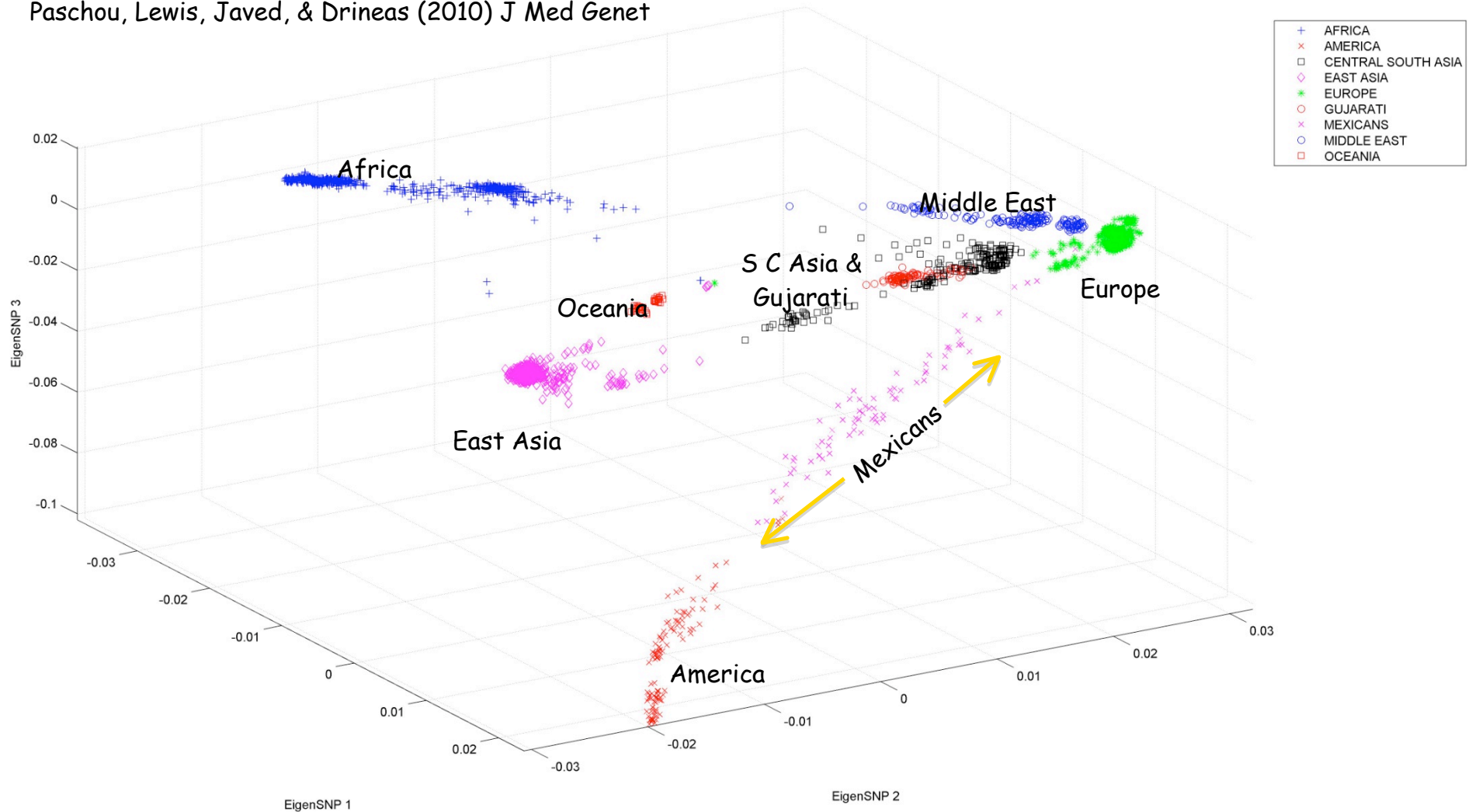


- Top two Principal Components (PCs or eigenSNPs)

(Lin and Altman (2005) *Am J Hum Genet*)

- The figure renders visual support to the “out-of-Africa” hypothesis.
- Mexican population seems out of place: we move to the top three PCs.

Paschou, Lewis, Javed, & Drineas (2010) J Med Genet



Not altogether satisfactory: the principal components are linear combinations of all SNPs, and - of course - can not be assayed!

Can we find **actual SNPs** that capture the information in the singular vectors?

Formally: **spanning the same subspace.**



Issues with eigen-analysis

- **Computing large SVDs: computational time**


- **In commodity hardware** (e.g., a 4GB RAM, dual-core laptop), using MatLab 7.0 (R14), the computation of the SVD of the dense 2,240-by-447,143 matrix *A* takes about 20 minutes.
- Computing this SVD is not a one-liner, since we can not load the whole matrix in RAM (runs out-of-memory in MatLab).
- Instead, compute the SVD of AA^T .
- In a similar experiment, compute **1,200 SVDs** on matrices of dimensions (approx.) 1,200-by-450,000 (roughly, a full leave-one-out cross-validation experiment).
(e.g., Drineas, Lewis, & Paschou (2010) PLoS ONE)

- **Selecting *actual columns* that “capture the structure” of the top PCs**

- Combinatorial optimization problem; hard even for small matrices.
- Often called the Column Subset Selection Problem (CSSP).
- Not clear that such “good” columns even exist.
- Avoid “reification” problem of “interpreting” singular vectors!



SVD decomposes a matrix as...

$$\begin{pmatrix} A \\ m \times n \end{pmatrix} = \begin{pmatrix} U \\ m \times \rho \end{pmatrix} \cdot \begin{pmatrix} \Sigma \\ \rho \times \rho \end{pmatrix} \cdot \begin{pmatrix} V \\ \rho \times n \end{pmatrix}^T$$


Top k left singular vectors

The SVD has very strong optimality properties., e.g. the matrix U_k is the "best" in many ways.

- Note that, given U_k , the best $X = U_k^T A = \Sigma V^T$.
- SVD can be computed fairly quickly.
- The columns of U_k are linear combinations of up to all columns of A .




CX (and CUR) matrix decompositions

Mahoney and Drineas (2009, PNAS); Drineas, Mahoney, and Muthukrishnan (2008, SIMAX)

$$\begin{pmatrix} A \end{pmatrix} \approx \begin{pmatrix} C \end{pmatrix} \begin{pmatrix} X \end{pmatrix}$$

Carefully
chosen X



Goal: choose actual columns C to make
(some norm) of $A-CX$ small.



c columns of A

Why?

If A is an subject-SNP matrix, then selecting representative columns is equivalent to **selecting representative SNPs to capture the same structure as the top eigenSNPs.**

Note: To make C small, **we want c as small as possible!**



CX (and CUR) matrix decompositions

Mahoney and Drineas (2009, PNAS); Drineas, Mahoney, and Muthukrishnan (2008, SIMAX)

$$\begin{pmatrix} A \end{pmatrix} \approx \begin{pmatrix} C \end{pmatrix} \begin{pmatrix} X \end{pmatrix}$$

↑
c columns of A

Easy to see optimal $X = C^+A$.

Hard to find **good columns** (e.g., SNPs) of A to include in C.

This **Column Subset Selection Problem (CSSP)**, heavily studied in N LA, is a hard combinatorial problem.

Two issues are connected

- There exist “good” columns in any matrix that contain information about the top principal components.
- We can identify such columns via a simple statistic: **the leverage scores**.
- This does not immediately imply faster algorithms for the SVD, but, **combined with random projections**, it does!
- Analysis (almost!) boils down to **understanding least-squares approximation** ...



Least Squares (LS) Approximation

$$\begin{pmatrix} A \\ n \times d, \quad n \gg d \end{pmatrix} \begin{pmatrix} \hat{x} \end{pmatrix} \approx \begin{pmatrix} b \end{pmatrix} \quad \rightarrow \quad \begin{aligned} Z_2 &= \min_{x \in \mathbb{R}^d} \|b - Ax\|_2 \\ &= \|b - A\hat{x}\|_2 \end{aligned}$$

We are interested in **over-constrained Lp regression problems**, $n \gg d$.

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

Want to find the "best" x such that $Ax \approx b$.

Ubiquitous in applications & central to theory:

Statistical interpretation: best linear unbiased estimator.

Geometric interpretation: orthogonally project b onto $\text{span}(A)$.



Exact solution to LS Approximation

Cholesky Decomposition:

If A is full rank and well-conditioned,
decompose $A^T A = R^T R$, where R is upper triangular, and
solve the normal equations: $R^T R x = A^T b$.

QR Decomposition:

Slower but numerically stable, esp. if A is rank-deficient.
Write $A = QR$, and solve $Rx = Q^T b$.

Singular Value Decomposition:

Most expensive, but best if A is very ill-conditioned.
Write $A = U \Sigma V^T$, in which case: $x_{\text{OPT}} = A^+ b = V \Sigma^{-1}_k U^T b$.

Complexity is $O(nd^2)$ for all of these, but
constant factors differ.

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

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 $\text{Var}(e)=\sigma^2 I$), uncorrelated, normally distributed

$x_{\text{opt}} = (A^T A)^{-1} A^T b$ (what we computed before)

$b' = Hb$ $H = A(A^T A)^{-1} A^T$ = **"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: $E(e')=0$, $\text{Var}(e')=\sigma^2(I-H)$

$\text{Trace}(H)=d$ **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 $\text{span}(A)$

$H_{ii} = \|U^{(i)}\|_2^2$ **leverage scores** = row **"lengths"** of spanning orthogonal matrix

Hat Matrix and Regression Diagnostics

See: "The Hat Matrix in Regression and ANOVA," Hoaglin and Welsch (1978)

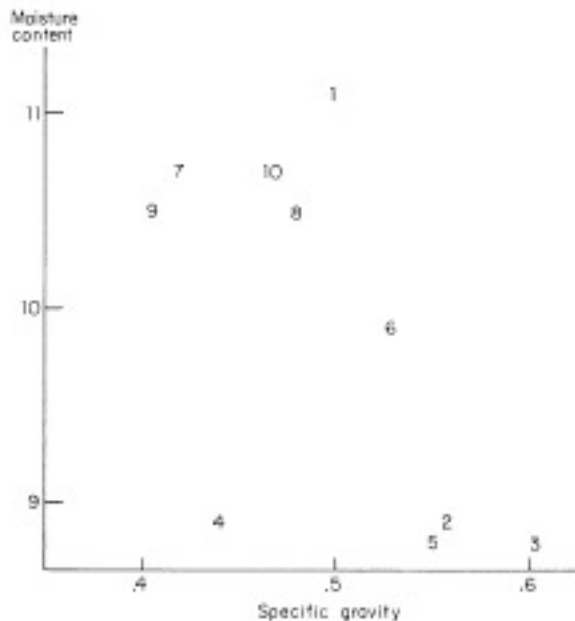


Figure A. The Two Carriers for the Wood Beam Data (Plotting symbol is beam number.).

2. The Hat Matrix for the Wood Beam Data (lower triangle omitted by symmetry)

i	j									
	1	2	3	4	5	6	7	8	9	10
1	.418	-.002	.079	-.274	-.046	.181	.128	.222	.050	.242
2		.242	.292	.136	.243	.128	-.041	.033	-.035	.004
3			.417	-.019	.273	.187	-.126	.044	-.153	.004
4				.504	.197	-.038	.168	-.022	.275	-.028
5					.252	.111	-.030	.019	-.010	-.010
6						.148	.042	.117	.012	.111
7							.262	.145	.277	.174
8								.154	.120	.168
9									.315	.148
10										.187

Examples of things to note:

- Point 4 is a bivariate outlier - and $H_{4,4}$ is largest, just exceeds $2p/n=6/10$.
- Points 1 and 3 have relatively high leverage - extremes in the scatter of points.
- $H_{1,4}$ is moderately negative - opposite sides of the data band.
- $H_{1,8}$ and $H_{1,10}$ moderately positive - those points mutually reinforce.
- $H_{6,6}$ is fairly low - point 6 is in central position.



A "classic" randomized algorithm (1 of 3)

Over-constrained least squares ($n \times d$ matrix $A, n \gg d$)

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

Randomized Algorithm:

- For all $i \in \{1, \dots, n\}$, compute $p_i = \frac{1}{d} \|U_{(i)}\|_2^2$
- Randomly sample $O(d \log(d)/\epsilon)$ rows/elements from 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 :

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

- *Important: this condition decouples the randomness from the linear algebra.*
- Random sampling algorithms with leverage score probabilities and random projections satisfy it!



Random projections: the JL lemma

For every set S of m points in \mathbb{R}^n and every $\epsilon > 0$, there exists a mapping $f : \mathbb{R}^n \rightarrow \mathbb{R}^s$, where $s = O(\log m / \epsilon^2)$, such that for all points $u \in S$,

$$(1 - \epsilon) \|u\|_2 \leq \|f(u)\|_2 \leq (1 + \epsilon) \|u\|_2$$

holds with probability at least $1 - 1/m^2$.

Johnson & Lindenstrauss (1984)

- We can represent S by an m -by- n matrix A , whose rows correspond to points.
- We can represent all $f(u)$ by an m -by- s \tilde{A} .
- The “mapping” corresponds to the construction of an n -by- s matrix Ω and computing

$$\tilde{A} = A \Omega$$



Different constructions for Ω matrix

“Slow” Random Projections ($\geq O(nd^2)$ time to implement in RAM model):

- JL (1984): random k -dimensional space
- Frankl & Maehara (1988): random orthogonal matrix
- DasGupta & Gupta (1999): random matrix with entries from $N(0,1)$, normalized
- Indyk & Motwani (1998): random matrix with entries from $N(0,1)$, normalized
- Achlioptas (2003): random matrix with entries in $\{-1,0,+1\}$, normalized
- Alon (2003): optimal dependency on n , and almost optimal dependency on ε

“Fast” Random Projections ($o(nd^2)$ time to implement in RAM model):

- Ailon and Chazelle (2006,2009); Matousek (2008); and many variants more recently.



Fast Johnson-Lindenstrauss Transform

Facts implicit or explicit in: Ailon & Chazelle (2006), Ailon and Liberty (2008), and Matousek(2008).

$$P \in \mathbb{R}^{s \times n}$$
$$s = O(\log m / \epsilon^2)$$

$$P_{ij} = \sqrt{q} \times \begin{cases} +1 & , \text{w.p. } q/2 \\ 0 & , \text{w.p. } 1-q \\ -1 & , \text{w.p. } q/2 \end{cases}$$
$$q = O(\log^2 m)$$

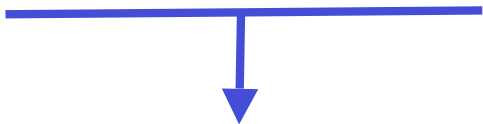
$$H \in \mathbb{R}^{n \times n}$$

Normalized Hadamard-Walsh transform matrix

(if n is not a power of 2, add all-zero columns to A ; or use other related Hadamard-based methods)

$$D \in \mathbb{R}^{n \times n}$$

Diagonal matrix with D_{ii} set to +1 or -1 w.p. 1/2.



$$R = (PHD)^T \in \mathbb{R}^{n \times s}$$

$$\longrightarrow \tilde{A} = \frac{1}{\sqrt{s}} AR$$

- P can also be a matrix representing the “uniform sampling” operation.
- In both cases, the $O(n \log(n))$ running time is computational bottleneck.



Randomized Hadamard preprocessing

Facts implicit or explicit in: Ailon & Chazelle (2006), Ailon and Liberty (2008), and Matousek(2008).

Let H_n be an n -by- n deterministic Hadamard matrix, and

Let D_n be an n -by- n random diagonal matrix with $+1/-1$ chosen u.a.r. on the diagonal.

Fact 1: Multiplication by $H_n D_n$ doesn't change the solution:

$$\|Ax - b\|_2 = \|H_n D_n Ax - H_n D_n b\|_2 = \|\mathcal{H}Ax - \mathcal{H}b\|_2$$

(since H_n and D_n are orthogonal matrices).

Fact 2: Multiplication by $H_n D_n$ is fast - only $O(n \log(r))$ time, where r is the number of elements of the output vector we need to "touch".

Fact 3: Multiplication by $H_n D_n$ approximately uniformizes all leverage scores:

$$\|U_{(i)\mathcal{H}A}\|_2 = \|(\mathcal{H}U_A)_{(i)}\|_2 \leq O\left(\sqrt{\frac{d \log n}{n}}\right)$$



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

Main theorem: For both of these randomized algorithms, we get:

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



Fast approximation of statistical leverage and matrix coherence (1 of 4)

Drineas, Magdon-Ismail, Mahoney, and Woodruff (2011, arXiv)

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 $n \gg d$, $O(\text{on-basis})$ time otherwise.

We want faster!

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



Fast approximation of statistical leverage and matrix coherence (2 of 4)

Drineas, Magdon-Ismail, Mahoney, and Woodruff (2011, arXiv)

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

Recall (A is $n \times d$, $n \gg d$):

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

But:

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

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



Fast approximation of statistical leverage and matrix coherence (3 of 4)

Drineas, Magdon-Ismail, Mahoney, and Woodruff (2011, arXiv)

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

- This is simpler than for the full under-constrained LS solution since only need the norm of the solution.
- This is essentially using R^{-1} from QR of subproblem as preconditioner for original problem.
- I.e., $\Omega_1 A$ is a **randomized "sketch"** of A ; $QR = \Omega_1 A$ is QR decomposition of this sketch; and evaluate row norms of $X \approx A R^{-1}$, but need Ω_2 , a second projection, to make it "fast."



Fast approximation of statistical leverage and matrix coherence (4 of 4)

Drineas, Magdon-Ismail, Mahoney, and Woodruff (2011, arXiv)

Theorem: Given an $n \times d$ matrix A , with $n \gg d$, 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 $(1 \pm \epsilon)$ 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 $O(n d \log(n)/\epsilon)$, i.e., same as DMMS fast randomized algorithm for over-constrained least squares.



Practically “fast” implementations

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!



LSRN: a fast parallel implementation (1 of 4)

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 (2 of 4)

Meng, Saunders, and Mahoney (2011, arXiv)

Algorithm:

- Generate a $\gamma n \times m$ 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: embarrassingly 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: a fast parallel implementation (3 of 4)

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	$A \setminus b$	Blendenpik	LSRN
landmark	71952	2704	1.15e6	2671	1.0e8	29.54	0.6498*	-	17.55
ra114284	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

LSRN: a fast parallel implementation (4 of 4)

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	tning_4	1024	4e6	8.4e7	106	34.03	170.4
LSRN w/ LSQR							84	41.14	178.6
LSRN w/ CS	5	10	tning_10	1024	1e7	2.1e8	106	50.37	193.3
LSRN w/ LSQR							84	68.72	211.6
LSRN w/ CS	10	20	tning_20	1024	2e7	4.2e8	106	73.73	220.9
LSRN w/ LSQR							84	102.3	249.0
LSRN w/ CS	20	40	tning_40	1024	4e7	8.4e8	106	102.5	255.6
LSRN w/ LSQR							84	137.2	290.2

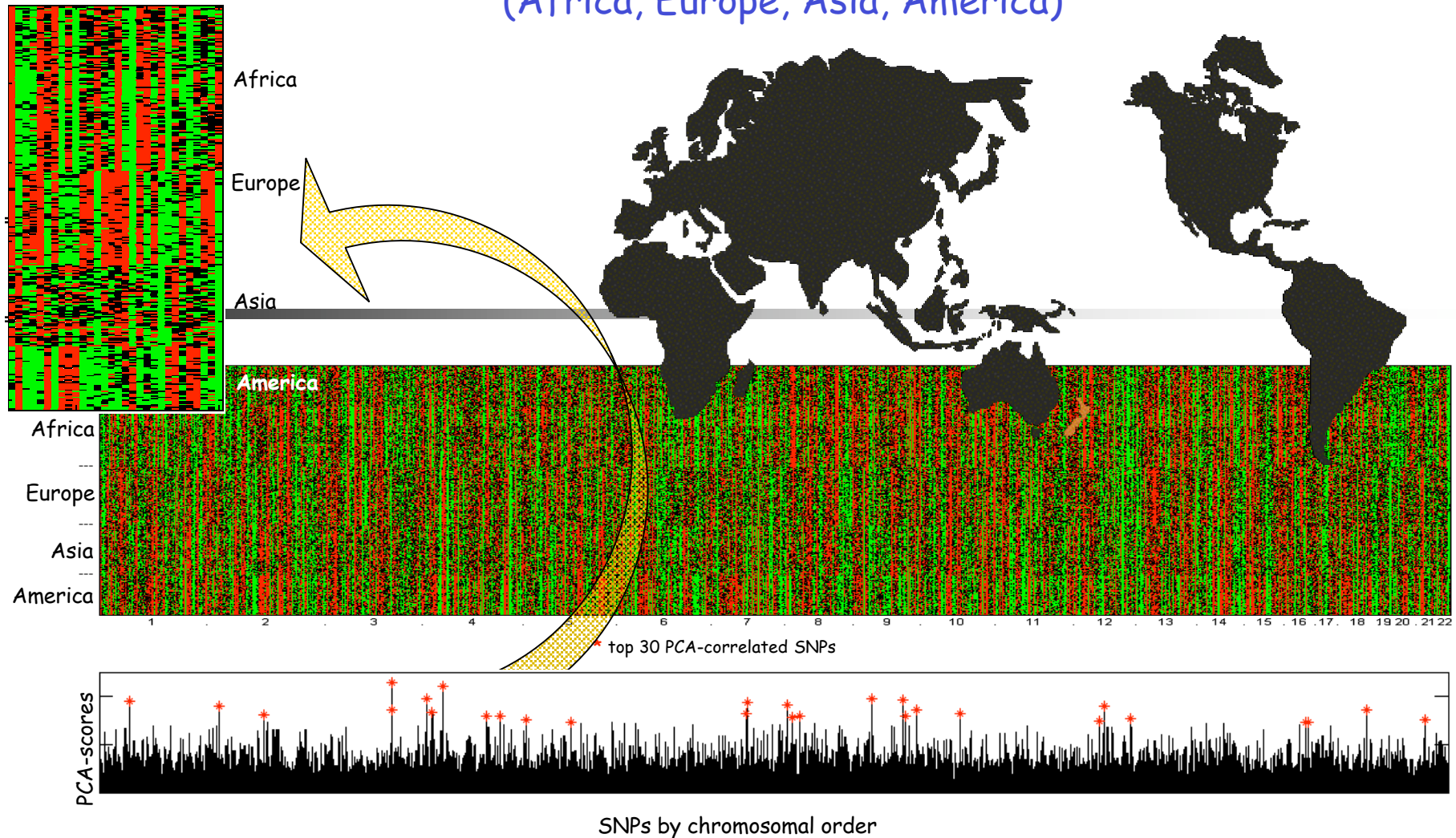


Low-rank approximation algorithms

Many randomized algorithms for **low-rank matrix approximation** use extensions of these basic least-squares ideas:

- Relative-error random sampling CX/CUR algorithms (DMM07)
- Relative-error random projection algorithms (S08)
- Column subset selection problem (exactly k columns) (BMD09)
- Numerical implementations, with connections to interpolative decomposition (LWMRT07, WLRT08, MRT11)
- Numerical implementations for slower spectral decay (RST09)

Selecting PCA SNPs for individual assignment to four continents (Africa, Europe, Asia, America)



Paschou et al (2007; 2008) PLoS Genetics

Paschou et al (2010) J Med Genet

Drineas et al (2010) PLoS One

Javed et al (2011) Annals Hum Genet



An interesting observation

Sampling w.r.t. to leverage scores results in redundant columns being selected.

(Almost) identical columns have (almost) the same leverage scores and thus might be all selected, even though they do not really add new “information.”

First Solution:

Apply a “redundancy removal” step, e.g., a deterministic CSSP algorithm on the sampled columns.

Very good empirically, even with “naïve” CSSP algorithms (such as the pivoted QR factorization).

Conjecture:

The “leverage scores” filter out relevant columns, so deterministic methods do a better job later.

Paschou et al. (2007,2008) for population genetics applications; and Boutsidis et al. (2009, 2010) for theory.

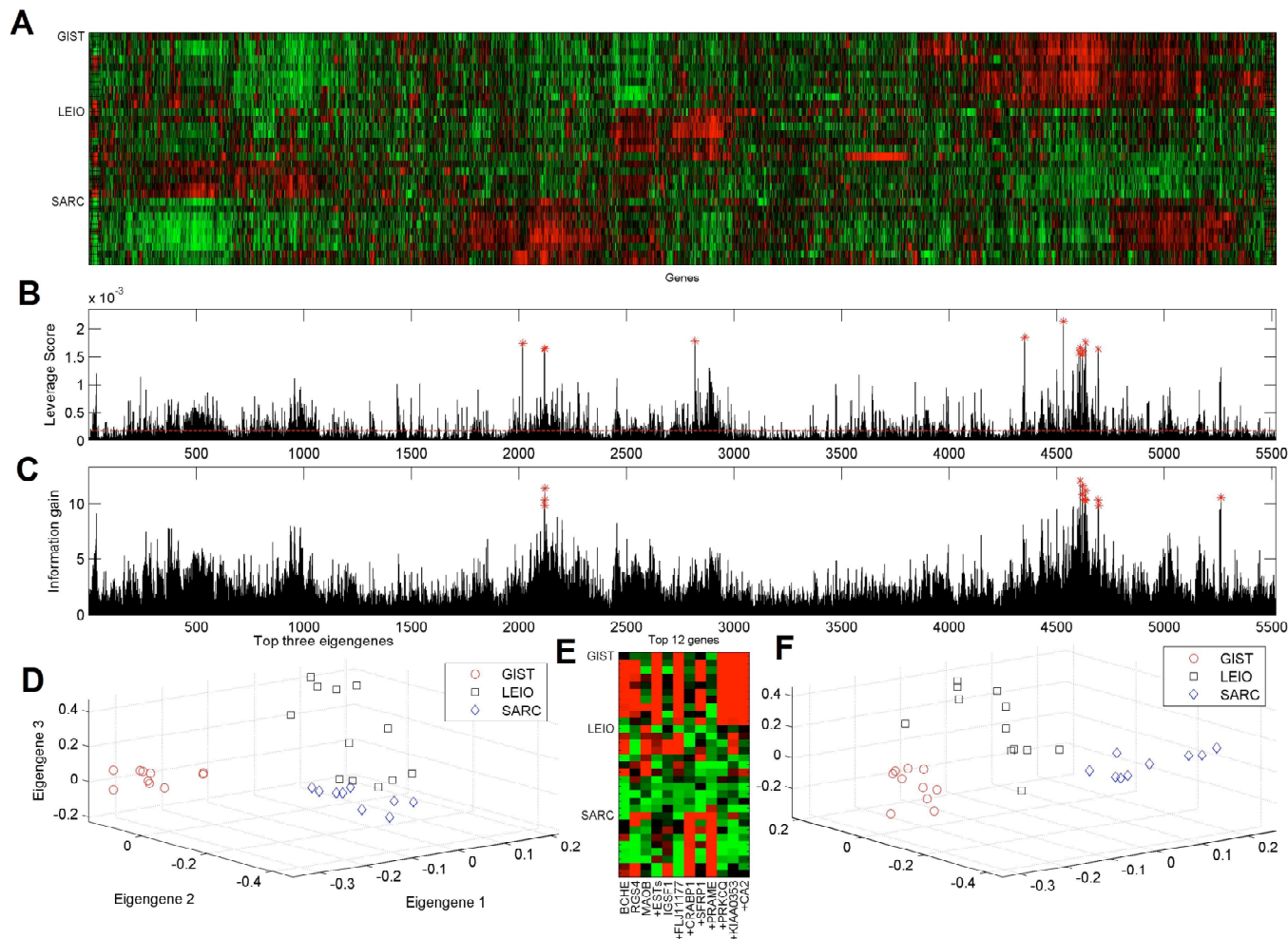
Second Solution:

Apply clustering to the sampled columns and then return a representative column from each cluster.

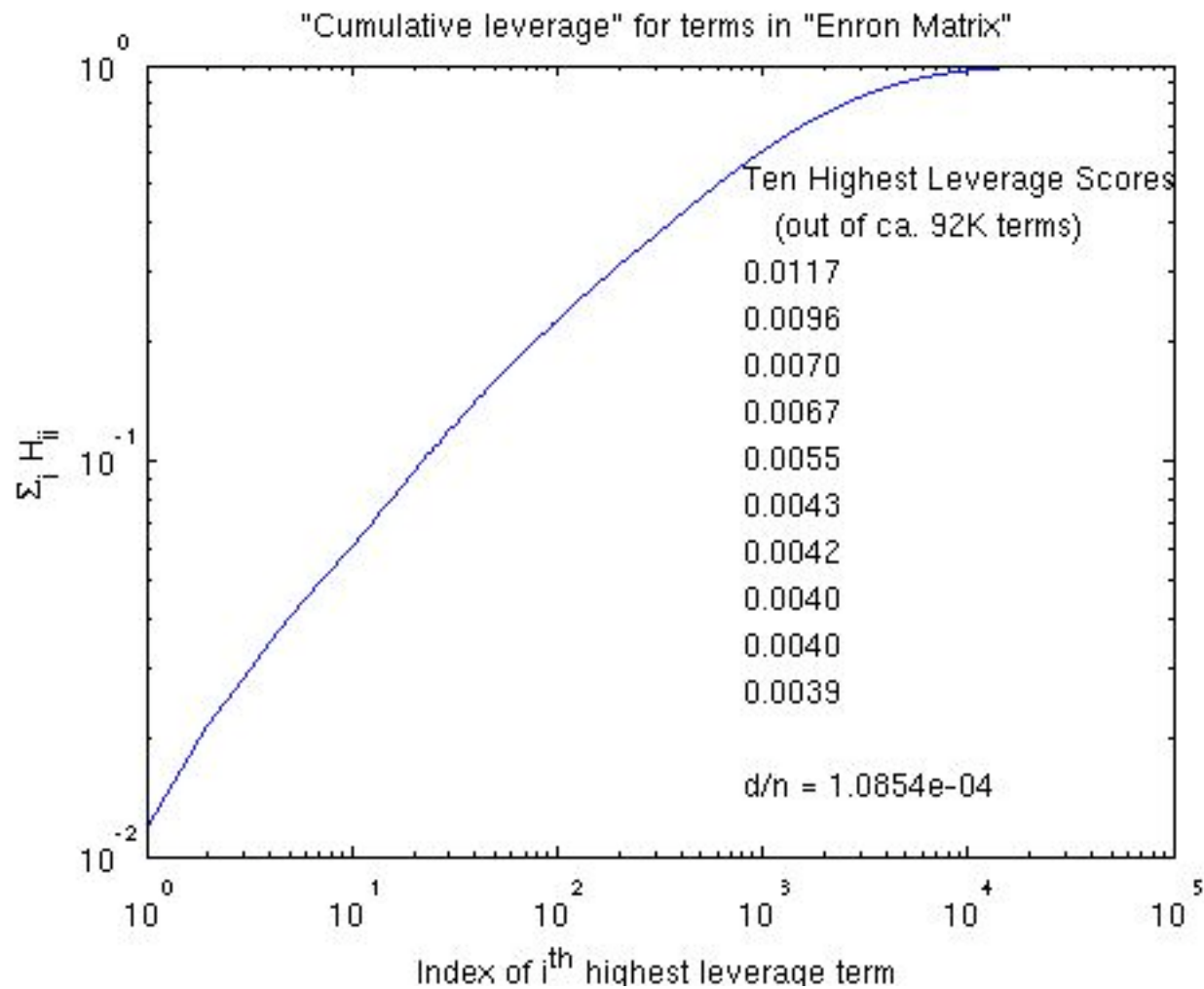
Very good empirically, since it permits clustering of SNPs that have similar functionalities and thus allows better understanding of the proposed ancestry-informative panels.

Statistical Leverage and DNA Microarray data

Mahoney and Drineas, PNAS (2009)



Statistical Leverage and Large Internet Data





Future directions?

Lots of them:

- Other traditional NLA and large-scale optimization problems
- Parallel and distributed computational environments
- Sparse graphs, sparse matrices, and sparse projections
- Laplacian matrices and large informatics graphs
- Randomized algorithms and implicit regularization
- ...

"New data and new problems are forcing us to reconsider the algorithmic and statistical basis of large-scale data analysis."



For more info ...

Two very good recent reviews:

- "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," by N. Halko, P. G. Martinsson, J. Tropp, *SIAM Review*, 53(2), 2011. (Also available at [arXiv:0909.4061](https://arxiv.org/abs/0909.4061)).
- "Randomized Algorithms for Matrices and Data," M. W. Mahoney, In press in NOW Publishers' Foundations and Trends in Machine Learning series. (Also available at [arXiv:1104.5557](https://arxiv.org/abs/1104.5557)).

And no doubt more to come ...