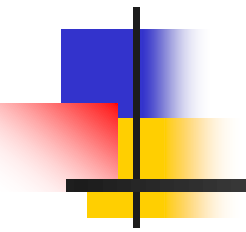


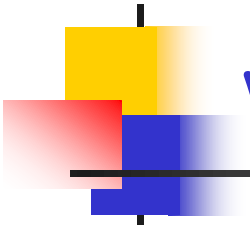
Randomized Algorithms in Data Mining: a Linear Algebraic Approach

Petros Drineas

Rensselaer Polytechnic Institute
Computer Science Department

To access my web page:

Google drineas



Why linear algebra?

Data are represented by matrices (or tensors)

Numerous modern datasets are in matrix form.

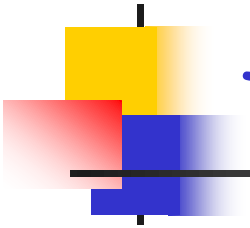
Data in the form of tensors (multi-mode arrays) have become very common in the data mining and information retrieval literature in the last few years.

Goal

Learn a model for the underlying "physical" system generating the data.

Toolbox

Linear algebra (and numerical analysis) provide the fundamental mathematical and algorithmic tools to deal with matrix and tensor computations.



Tool: matrix decompositions

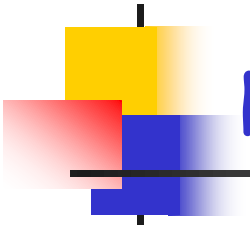
Matrix decompositions

(e.g., SVD, QR, SDD, CX and CUR, NMF, etc.)

• They use the relationships between the available data in order to identify components of the underlying physical system generating the data.

• Some assumptions on the relationships between the underlying components are necessary.

• Very active area of research; some matrix decompositions are more than one century old, whereas others are very recent.



Randomized algorithms

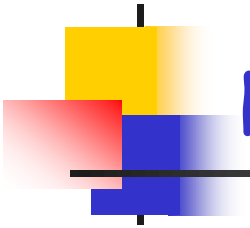
Randomization and sampling allow us to design provably accurate algorithms for problems that are:

- Massive

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

- Computationally expensive or NP-hard

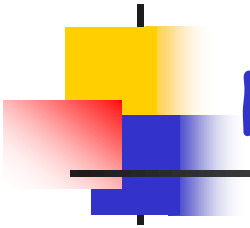
(e.g., combinatorial optimization problems such as the Column Subset Selection Problem and the related CX factorization)



Randomized algorithms & Linear Algebra

- **Randomized algorithms**
  - By (carefully) sampling rows/columns/entries of a matrix, we can construct new matrices (that have smaller dimensions or are sparse) and have bounded distance (in terms of some matrix norm) from the original matrix (with some failure probability).
  - By preprocessing the matrix using random projections (*), we can sample rows/columns/entries(?) much less carefully (uniformly at random) and still get nice bounds (with some failure probability).

(*) Alternatively, we can assume that the matrix is "well-behaved" and thus uniform sampling will work.



Randomized algorithms & Linear Algebra

- **Randomized algorithms**
  - By (carefully) sampling rows/columns/entries of a matrix, we can construct new matrices (that have smaller dimensions or are sparse) and have bounded distance (in terms of some matrix norm) from the original matrix (with some failure probability).
  - By preprocessing the matrix using random projections, we can sample rows/columns/entries(?) much less carefully (uniformly at random) and still get nice bounds (with some failure probability).
- **Matrix perturbation theory**
  - The resulting smaller/sparser matrices behave similarly (in terms of singular values and singular vectors) to the original matrices thanks to the norm bounds.

In this talk, I will illustrate some applications of the above ideas.

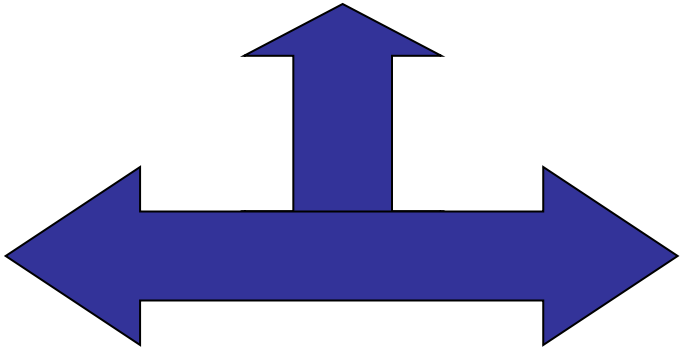
Interplay

(Data Mining) Applications

Biology & Medicine: **population genetics (coming up…)**

Electrical Engineering: testing of electronic circuits

Internet Data: recommendation systems, document-term data



Theoretical Computer Science

Randomized and approximation algorithms

Numerical Linear Algebra

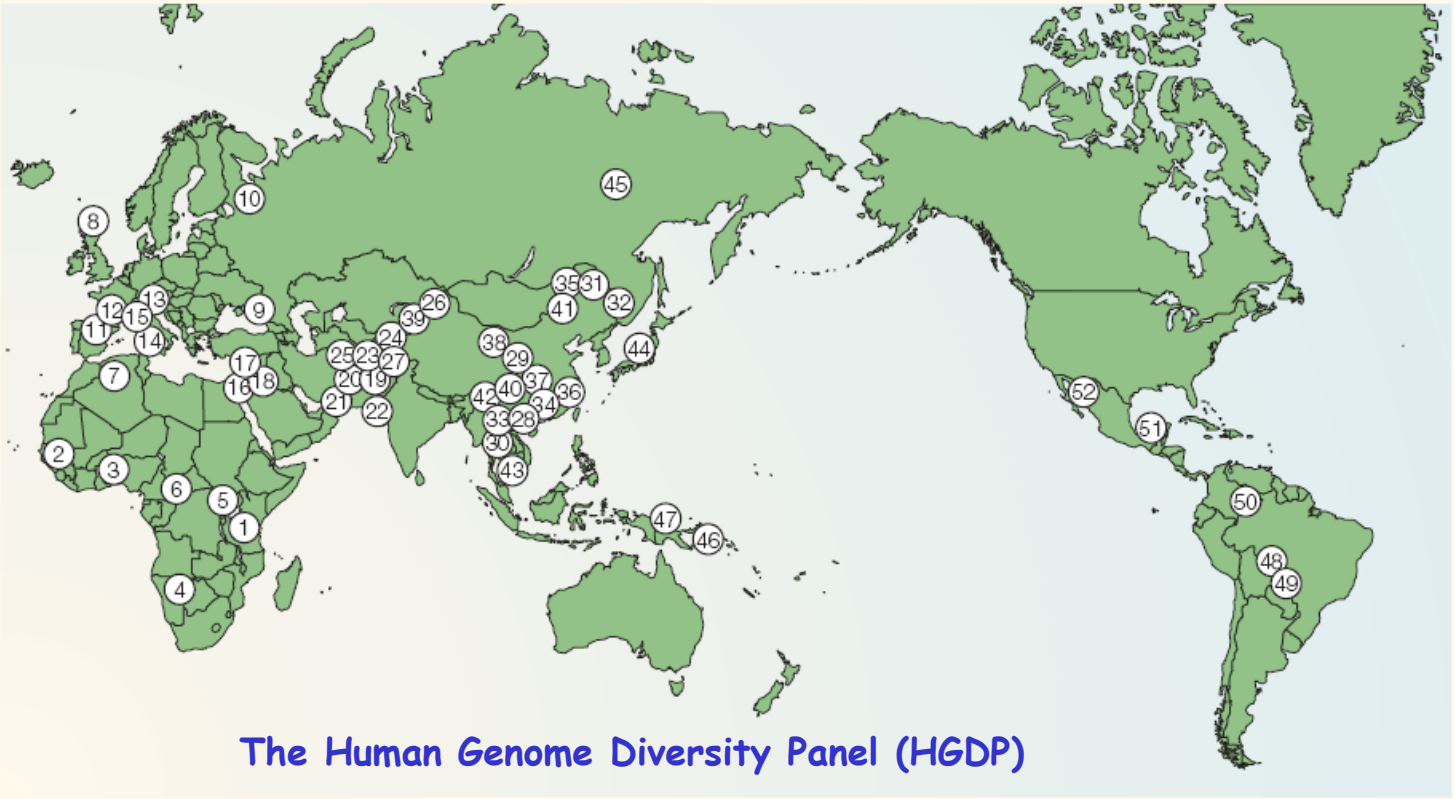
Matrix computations and Linear Algebra (ie., perturbation theory)

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).

Matrices including thousands of individuals and hundreds of thousands if 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

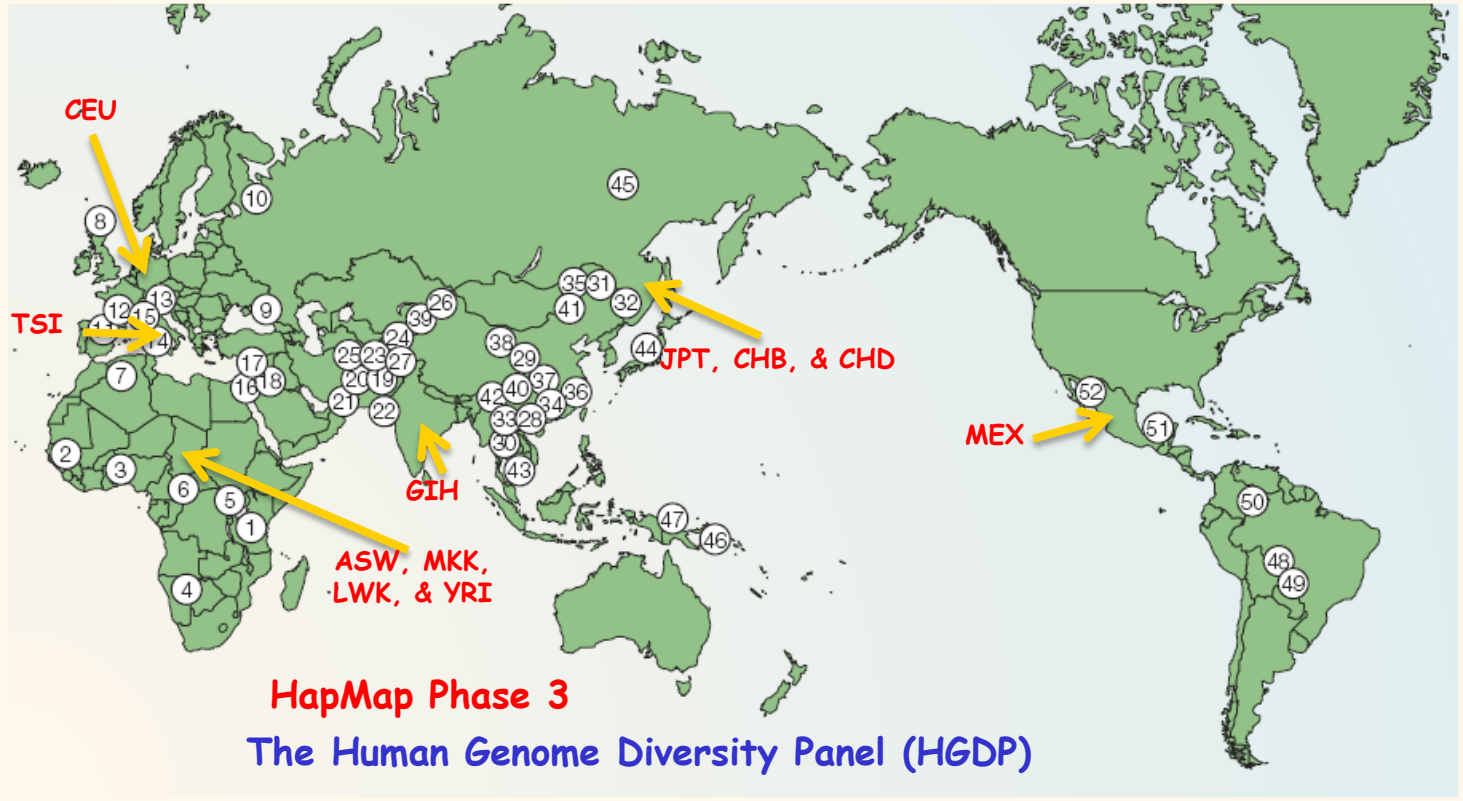
Cavalli-Sforza (2005) *Nat Genet Rev*

Rosenberg et al. (2002) *Science*

Li et al. (2008) *Science*

HGDP data

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



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*

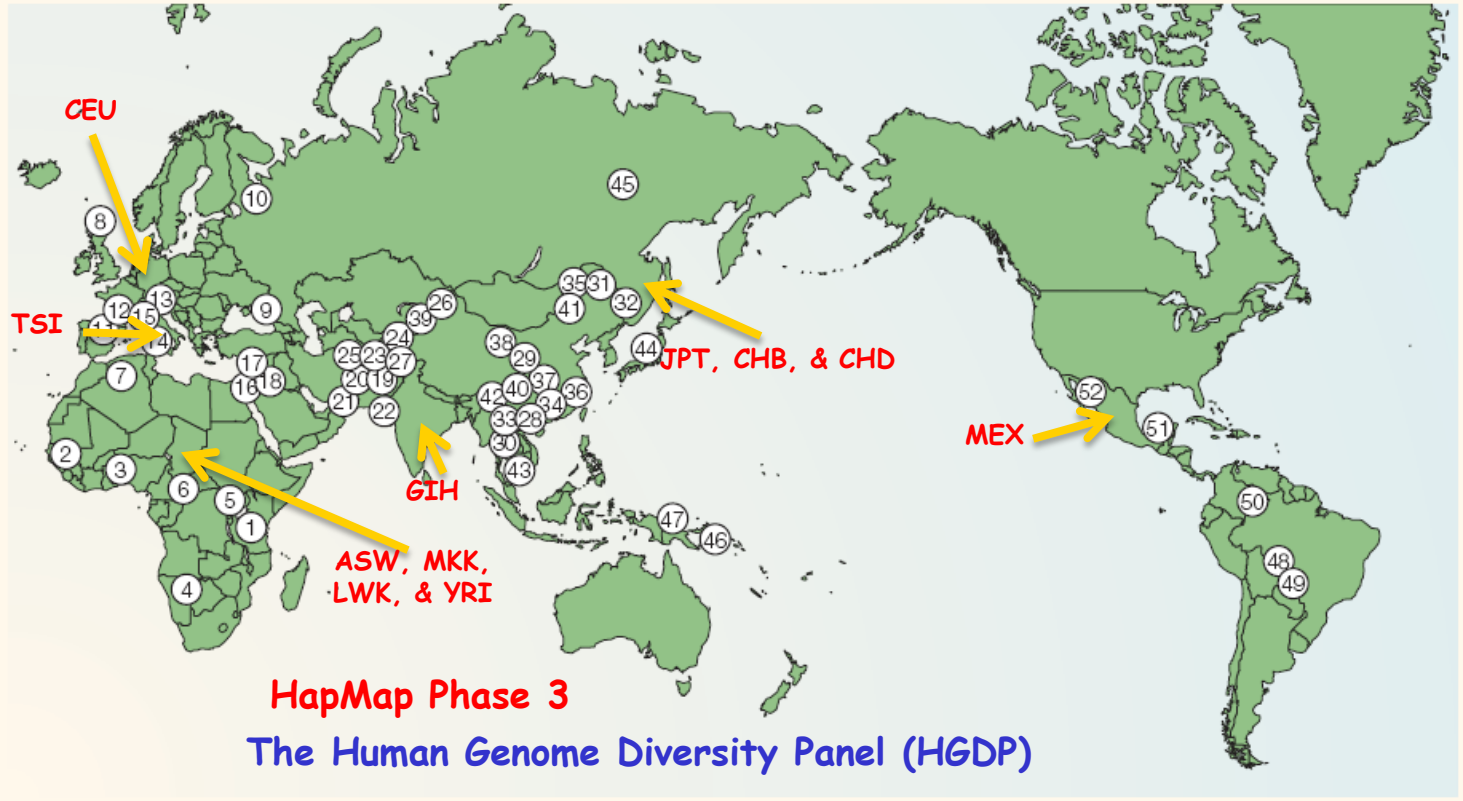
The International HapMap Consortium (2003, 2005, 2007) *Nature*

HGDP data

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

HapMap Phase 3 data

- 1,207 samples
- 11 populations



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*

HGDP data

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

HapMap Phase 3 data

- 1,207 samples
- 11 populations

We will apply SVD/PCA on the (joint) HGDP and HapMap Phase 3 data.

Matrix dimensions:

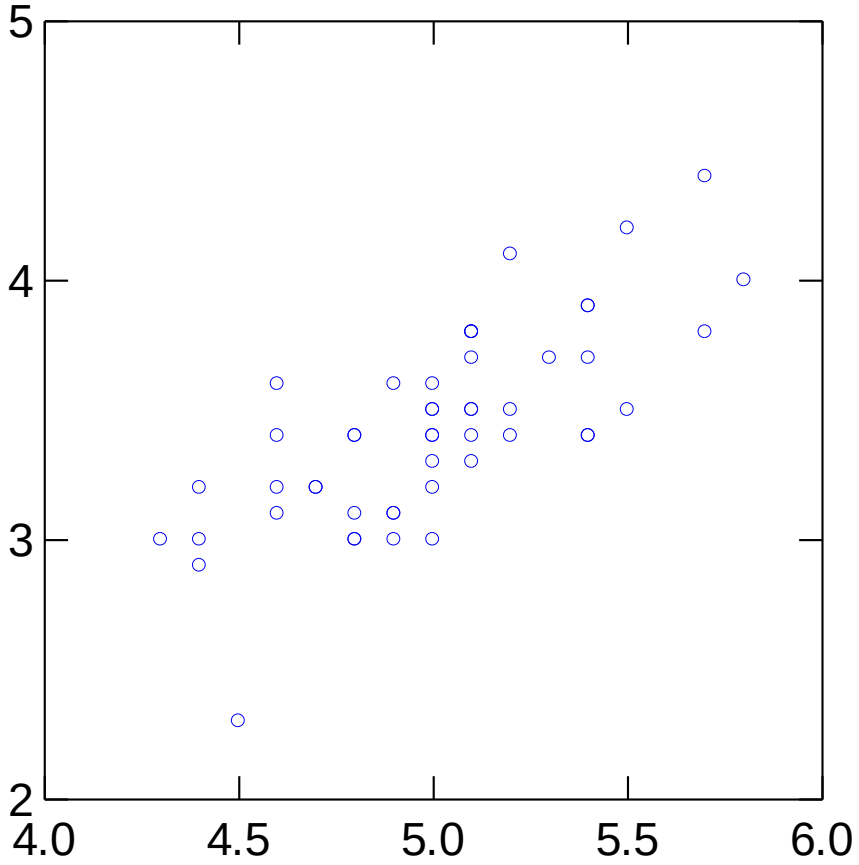
2,240 subjects (rows)

447,143 SNPs (columns)

Dense matrix:

over one billion entries

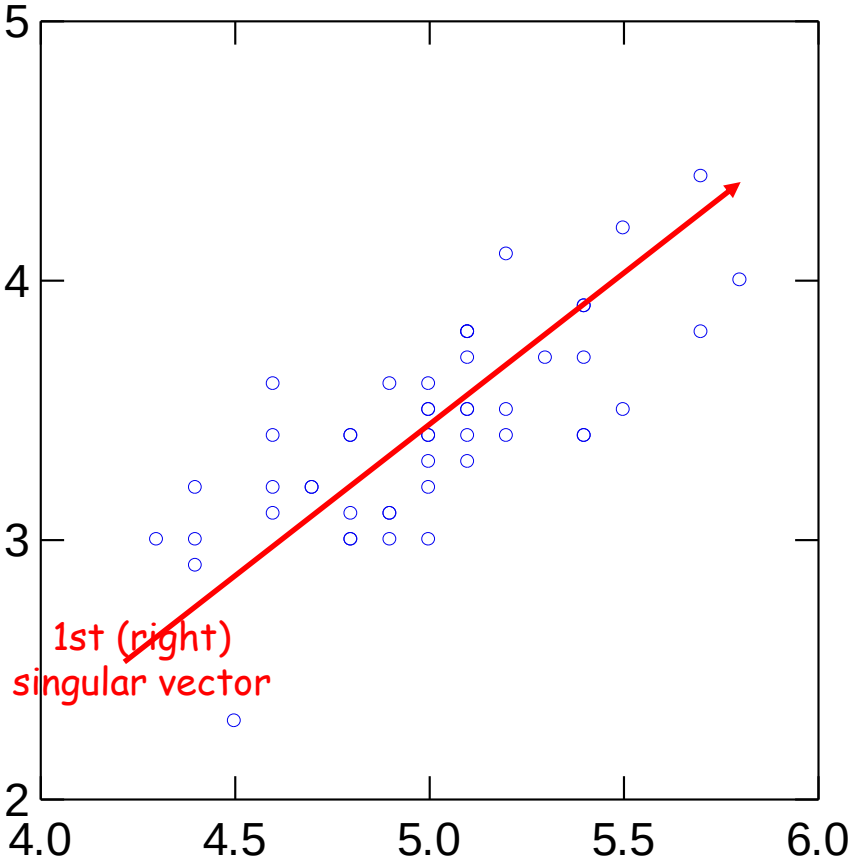
The Singular Value Decomposition (SVD)



Let the blue circles represent *m* data points in a 2-D Euclidean space.

Then, the SVD of the *m-by-2* matrix of the data will return …

The Singular Value Decomposition (SVD)



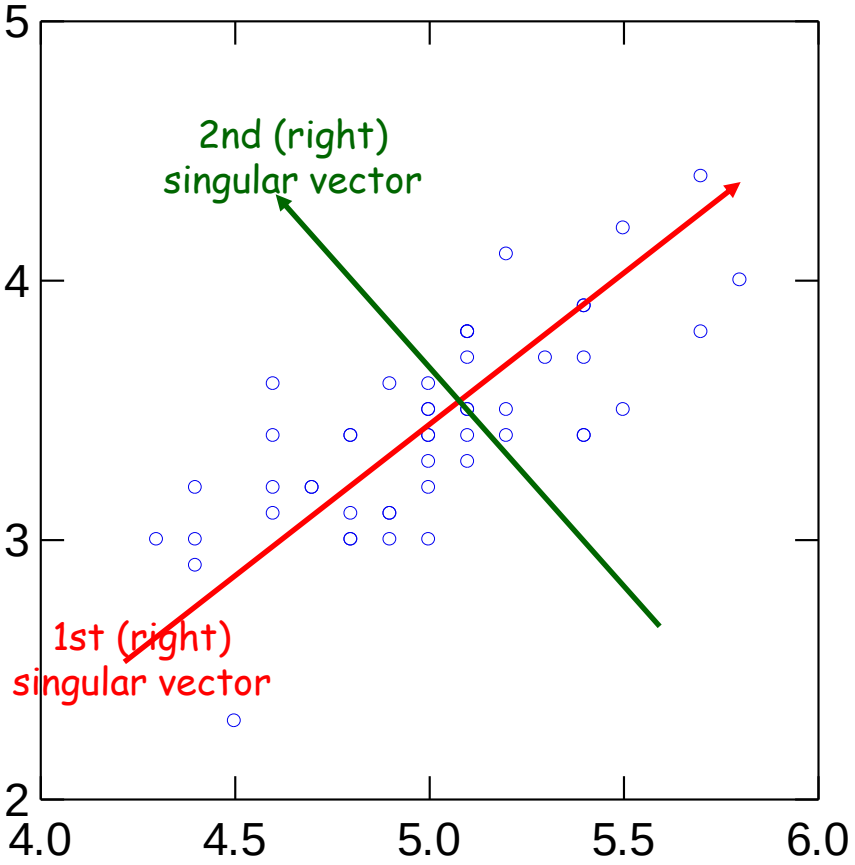
Let the blue circles represent *m* data points in a 2-D Euclidean space.

Then, the SVD of the *m-by-2* matrix of the data will return …

1st (right) singular vector:

direction of maximal variance,

The Singular Value Decomposition (SVD)



Let the blue circles represent *m* data points in a 2-D Euclidean space.

Then, the SVD of the *m-by-2* matrix of the data will return …

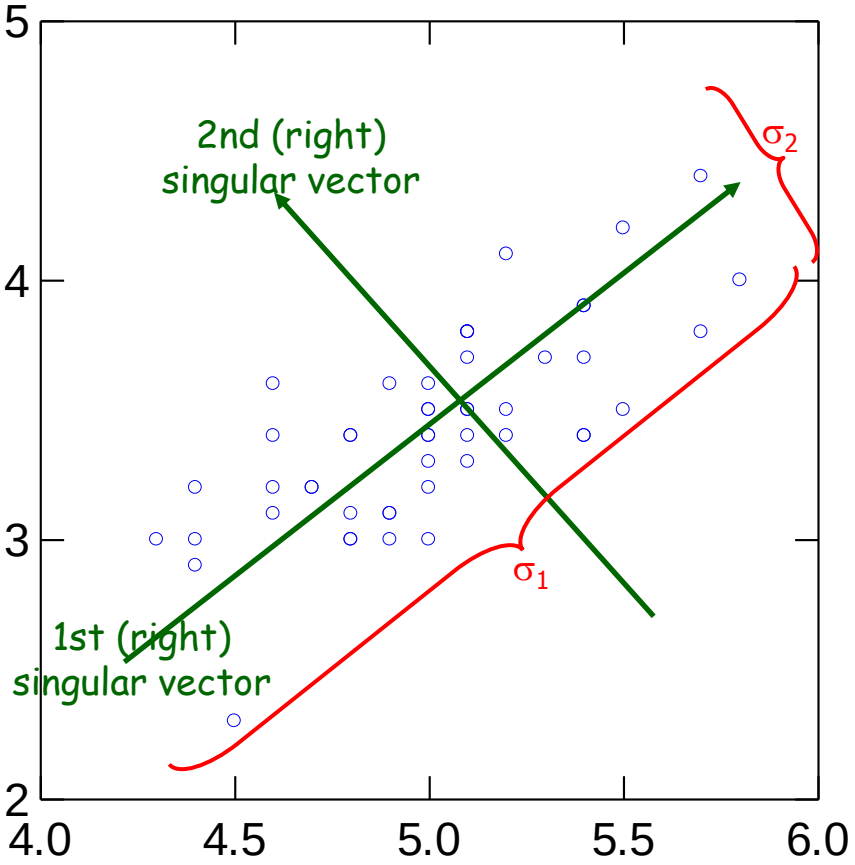
1st (right) singular vector:

direction of maximal variance,

2nd (right) singular vector:

direction of maximal variance, after removing the projection of the data along the first singular vector.

Singular values



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

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

Principal Components Analysis (PCA) is done via the computation of the Singular Value Decomposition (SVD) of a (mean-centered) covariance matrix.

Typically, a small constant number (say k) of the top singular vectors and values are kept.

SVD: formal definition

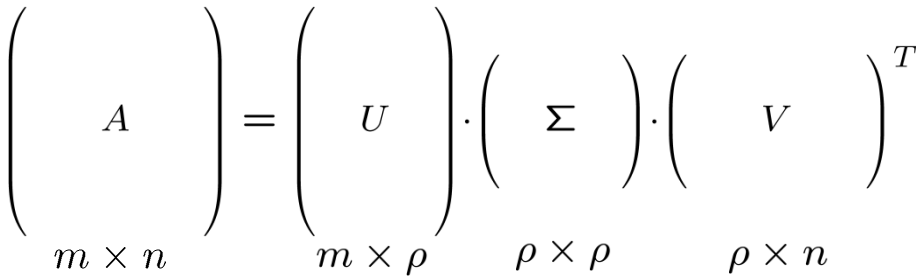
$$\underset{m \times n}{\begin{pmatrix} \\ A \\ \\ \end{pmatrix}} = \underset{m \times \rho}{\begin{pmatrix} \\ U \\ \\ \end{pmatrix}} \cdot \underset{\rho \times \rho}{\begin{pmatrix} \\ \Sigma \\ \\ \end{pmatrix}} \cdot \underset{\rho \times n}{\begin{pmatrix} \\ V \\ \\ \end{pmatrix}^T}$$

ρ: rank of A

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

Σ: diagonal matrix containing the singular values of A.

SVD: formal definition



ρ: rank of *A*

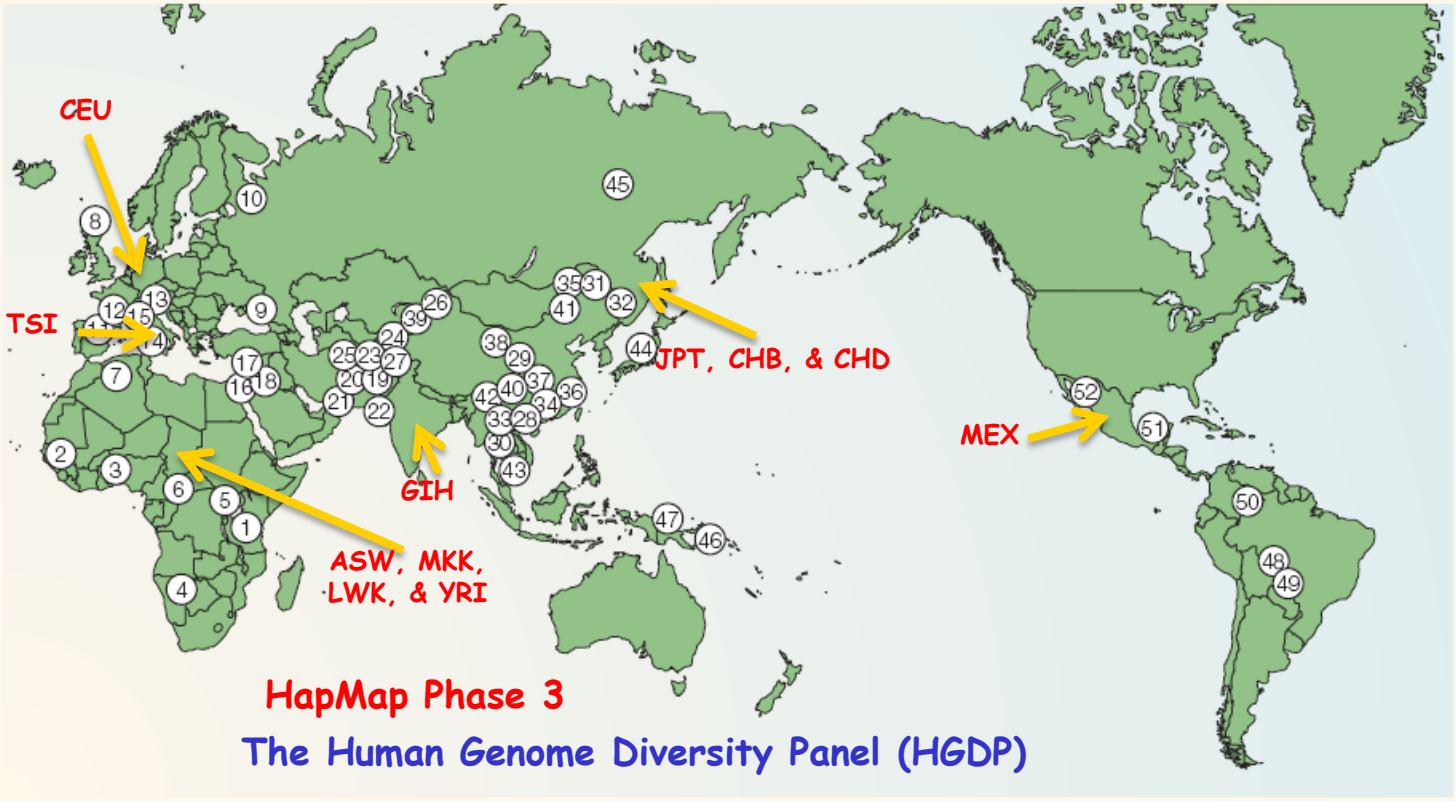
U (V): orthogonal matrix containing the left (right) singular vectors of *A*.

Σ: diagonal matrix containing the singular values of *A*.

Let σ_1 , σ_2 , … , σ_ρ be the entries of Σ.

Computing the SVD takes $O(\min\{mn^2 , m^2n\})$ time.

The top *k* left/right singular vectors/values can be computed faster using iterative methods.



HGDP data

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

HapMap Phase 3 data

- 1,207 samples
- 11 populations

Matrix dimensions:

2,240 subjects (rows)

447,143 SNPs (columns)

SVD/PCA returns…

Africans	Europeans	Western Asians	Eastern Asians	Oceanians
1 Bantu	8 Orcadian	16 Bedouin	28 Han (S. China)	46 Melanesian
2 Mandenka	9 Adygei	17 Druze	29 Han (N. China)	47 Papuan
3 Yoruba	10 Russian	18 Palestinian	30 Dai	
4 San	11 Basque		31 Daur	
5 Mbuti pygmy	12 French	Central and Southern Asians	32 Hezhen	Native Americans
6 Biaka	13 North Italian	19 Balochi	33 Lahu	48 Karitiana
7 Mozabite	14 Sardinian	20 Brahui	34 Miao	49 Surui
	15 Tuscan	21 Makrani	35 Oroqen	50 Colombian
		22 Sindhi	36 She	51 Maya
		23 Pathan	37 Tujia	52 Pima
		24 Burusho	38 Tu	
		25 Hazara	39 Xibo	
		26 Uygur	40 Yi	
		27 Kalash	41 Mongola	
			42 Naxi	
			43 Cambodian	
			44 Japanese	
			45 Yakut	

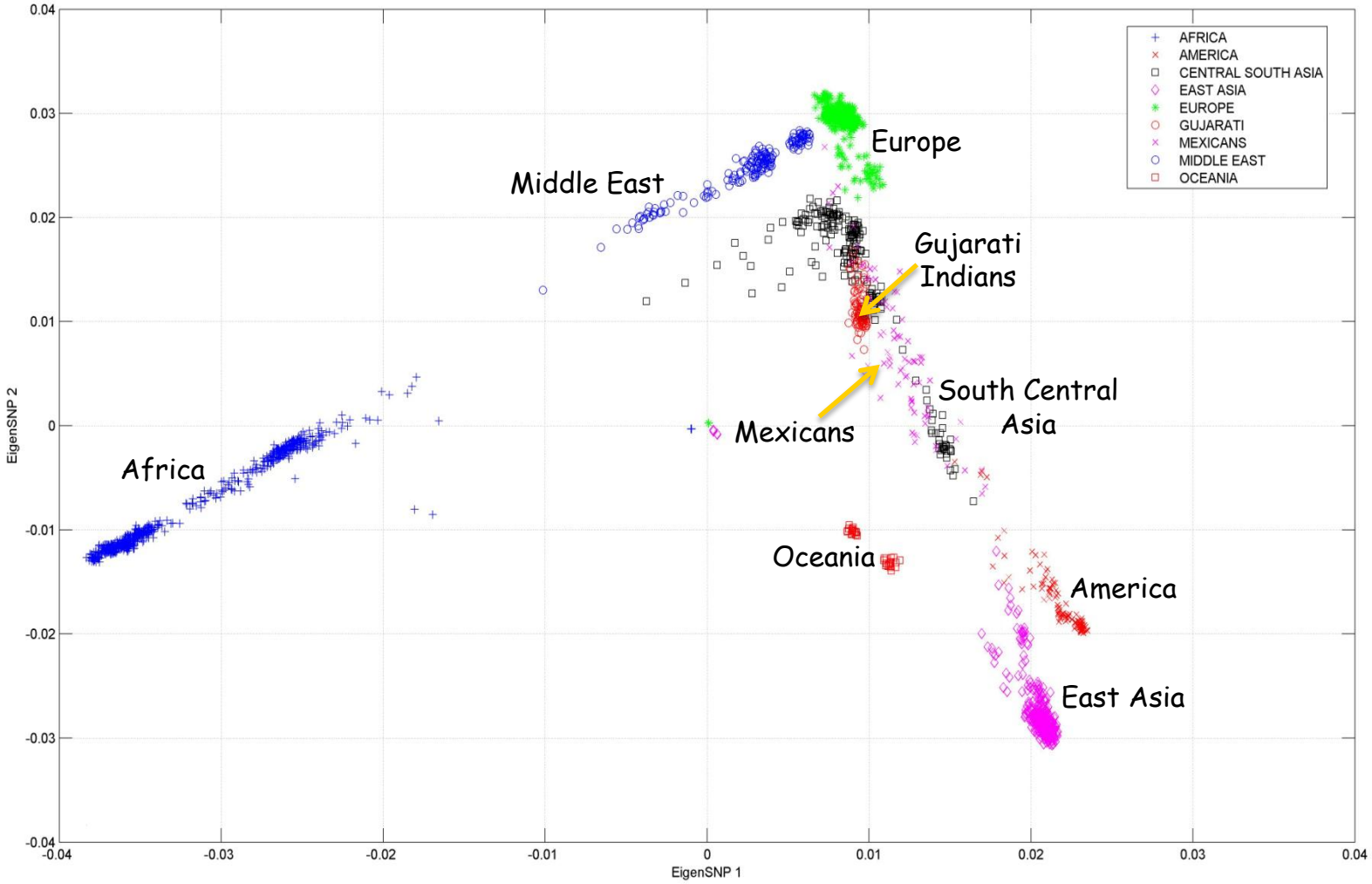
Cavalli-Sforza (2005) *Nat Genet Rev*

Rosenberg et al. (2002) *Science*

Li et al. (2008) *Science*

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

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

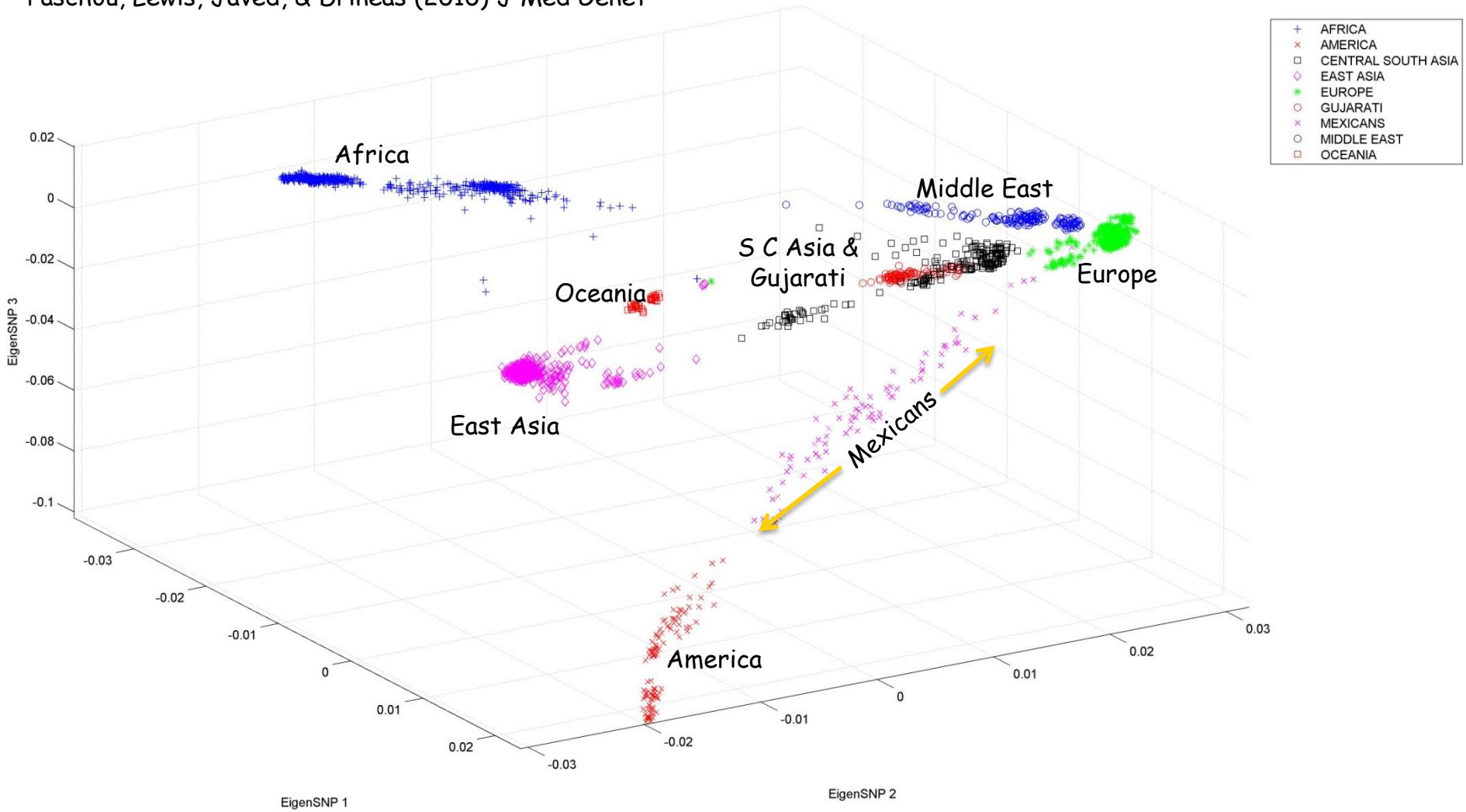


- Top two Principal Components (PCs or eigenSNPs)

(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.

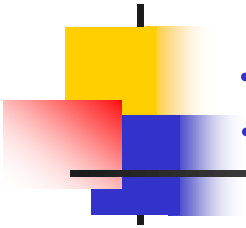
Paschou, Lewis, Javed, & Drineas (2010) J Med Genet



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?

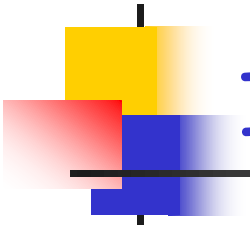
Formally: spanning the same subspace.



Issues

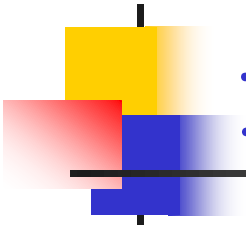
- **Computing large SVDs: computational time**
  - In commodity hardware (e.g., a 4GB RAM, dual-core laptop), using MatLab 7.0 (R14), the computation of the SVD of the dense 2,240-by-447,143 matrix A takes about 12 minutes.
  - Computing this SVD is not a one-liner, since we can not load the whole matrix in RAM (runs out-of-memory in MatLab).
  - We compute the eigendecomposition of AA^T.
  - In a similar experiment, we computed **1,200 SVDs** on matrices of dimensions (approx.) 1,200-by-450,000 (roughly speaking a full leave-one-out cross-validation experiment).

  (Drineas, Lewis, & Paschou (2010) PLoS ONE)
- **Obviously, running time is a concern.**
- **We need efficient, easy to implement, methods.**



Issues (cont'd)

- **Selecting good 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 columns even exist.



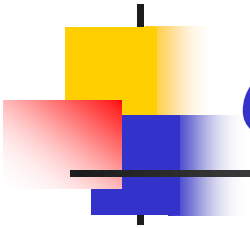
Issues (cont'd)

- **Selecting good 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 columns even exist.

Such datasets will only continue to increase in size:

In collaboration with K. Kidd's lab (Yale University, Department of Genetics) we are now analyzing:

- **4,000 samples** from over **100 populations**
- genotyped on over **500,000 SNPs**.

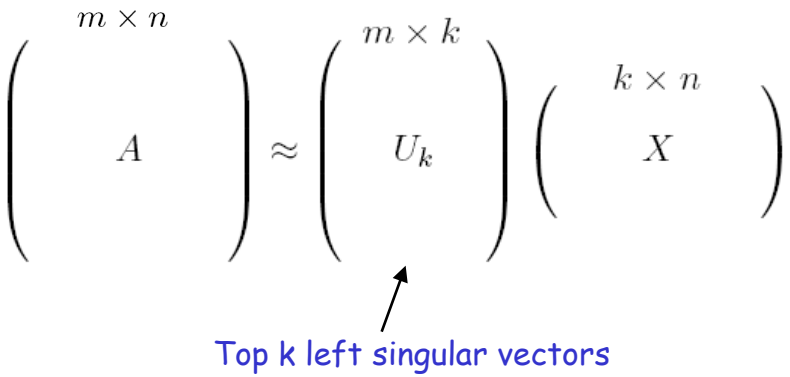


Our perspective

The 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!

SVD decomposes a matrix as…

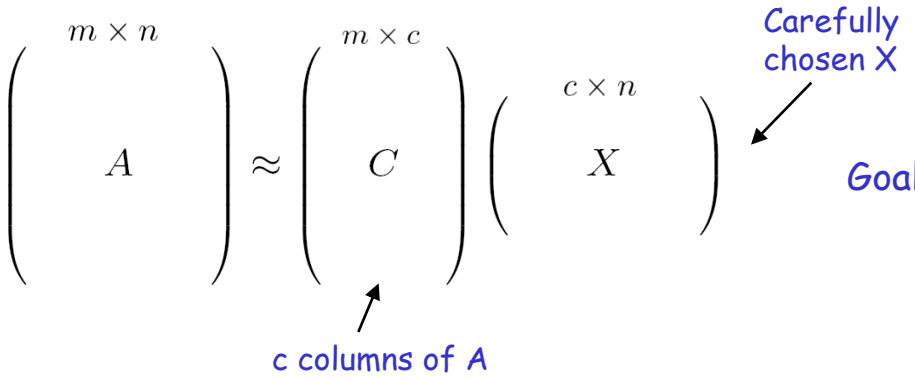


The SVD has strong optimality properties.

- It is easy to see that $X = U_k^T A$.
- SVD has strong optimality properties.
- The columns of U_k are linear combinations of up to all columns of A.

The CX decomposition

Drineas, Mahoney, & Muthukrishnan (2008) SIAM J Mat Anal Appl
Mahoney & Drineas (2009) PNAS



Goal: make (some norm) of A-CX small.

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.

We want c as small as possible!

CX decomposition

$$\underset{m \times n}{\begin{pmatrix} \\ A \\ \\ \end{pmatrix}} \approx \underset{m \times c}{\begin{pmatrix} \\ C \\ \\ \end{pmatrix}} \underset{c \times n}{\begin{pmatrix} \\ X \\ \\ \end{pmatrix}}$$

c columns of *A*

Easy to prove that optimal X = C^+A. (C^+ is the Moore-Penrose pseudoinverse of *C*.)

Thus, the challenging part is to find good columns (SNPs) of *A* to include in *C*.

From a mathematical perspective, this is a hard combinatorial problem, closely related to the so-called Column Subset Selection Problem (CSSP).

The CSSP has been heavily studied in Numerical Linear Algebra.

A much simpler statistic

(Frieze, Kannan, & Vempala FOCS 1998, Drineas, Frieze, Kannan, Vempala & Vinay SODA '99, Drineas, Kannan, & Mahoney SICOMP '06)

Algorithm: given an m-by-n matrix A, let $A^{(i)}$ be the i-th column of A.

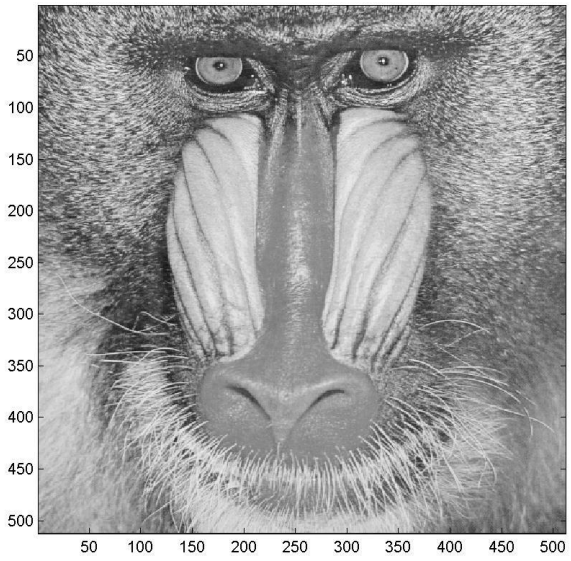
- Sample s columns of A in i.i.d. trials (with replacement), where in each trial

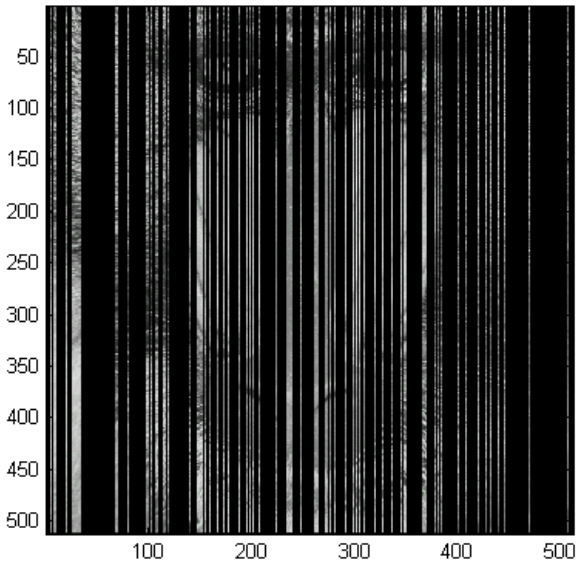
$$\mathbf{Pr}\left[\text{picking the } i\text{-th column}\right] = \frac{\left\|A^{(i)}\right\|_2^2}{\|A\|_F^2}$$

- Form the m-by-s matrix C by including $A^{(i)}$ as a column of C.

Error bound: $\mathbf{E}\left[\left\|A - CC^{+}A\right\|_F^2\right] \leq \|A - A_k\|_F^2 + \sqrt{\frac{4k}{s}}\,\|A\|_F^2$

Approximating singular vectors





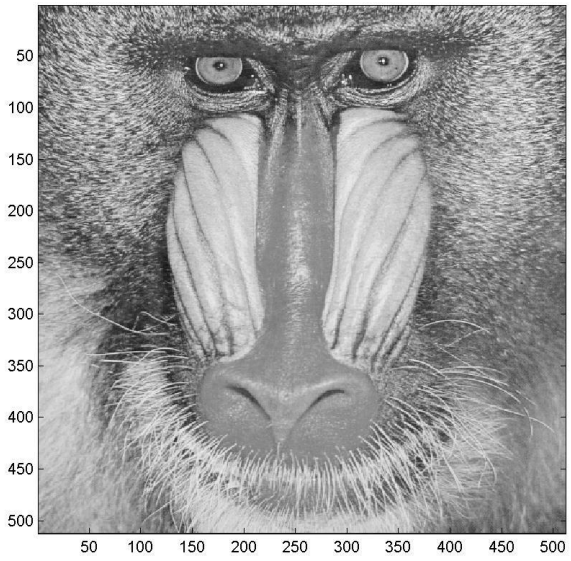
Original matrix

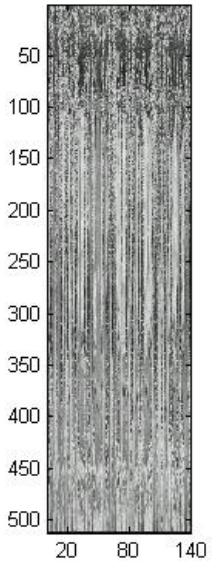
Sampling (s=140 columns)

1. Sample s (=140) columns of the original matrix A and form a 512-by-c matrix C.
2. Project A on CC^+ and show that $A-CC^+A$ is "small".

(C^+ is the pseudoinverse of C)

Approximating singular vectors





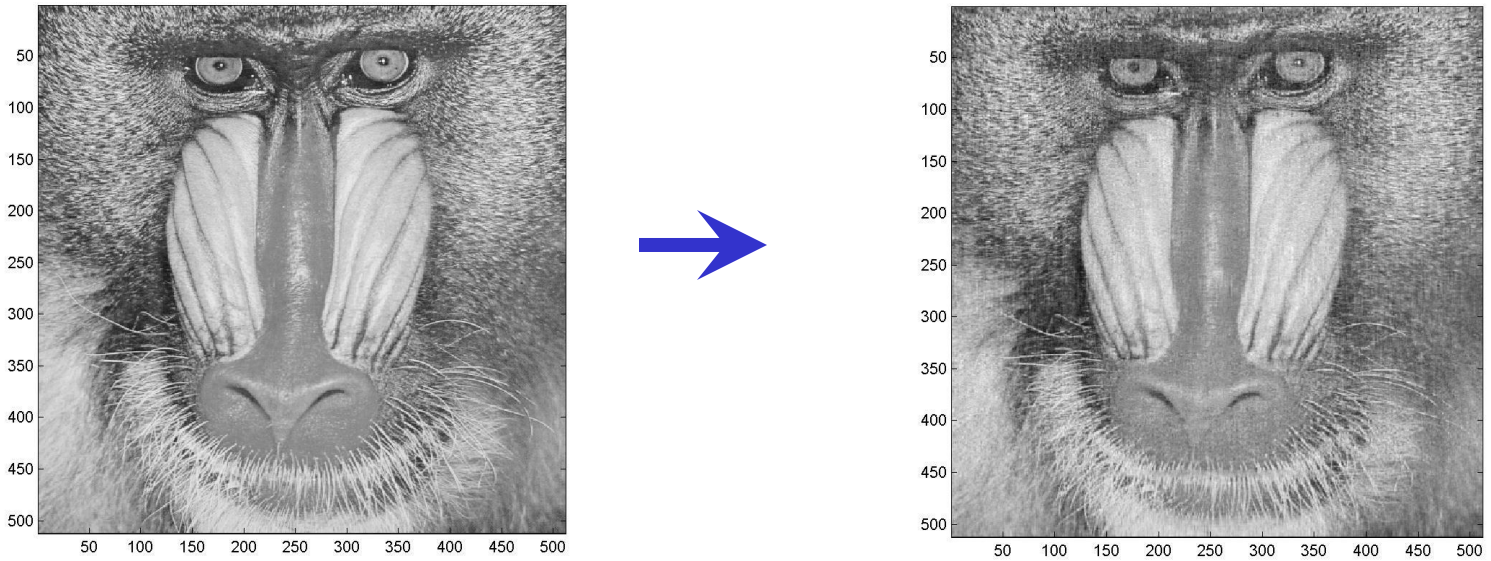
Original matrix

Sampling (s=140 columns)

1. Sample s (=140) columns of the original matrix A and form a 512-by-c matrix C.
2. Project A on CC^+ and show that $A-CC^+A$ is "small".

(C^+ is the pseudoinverse of C)

Approximating singular vectors (cont'd)



A CC^+A

Remark 1: Selecting the columns in this setting is trivial and can be implemented in a couple of (sequential) passes over the input matrix.

Remark 2: The proof is based on matrix perturbation theory and a probabilistic argument to bound $AA^T - \hat{C}\hat{C}^T$ (where $\hat{C}$ is a rescaled C).

Is this a good bound?

$$\mathbf{E}\left[\left\|A - CC^{+}A\right\|_F^2\right] \leq \|A - A_k\|_F^2 + \sqrt{\frac{4k}{s}}\,\|A\|_F^2$$

Problem 1: If s = n, we still do not get zero error.

That's because of sampling with replacement.

(We know how to analyze uniform sampling without replacement, but we have no bounds on non-uniform sampling without replacement.)

Problem 2: If A had rank exactly k, we would like a column selection procedure that drives the error down to zero when s=k.

This can be done deterministically simply by selecting k linearly independent columns.

Problem 3: If A had *numerical rank k*, we would like a bound that depends on the norm of $A-A_k$ and not on the norm of A.

A lot of prior work in the Numerical Linear Algebra community for the **spectral norm case** when s=k; the resulting bounds depend (roughly) on $(k(n-k))^{1/2}||A-A_k||_2$

Relative-error Frobenius norm bounds

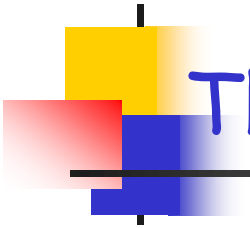
Drineas, Mahoney, & Muthukrishnan (2008) SIAM J Mat Anal Appl

Given an m-by-n matrix A, there exists an $O(mn^2)$ algorithm that picks

at most $O((k/\varepsilon^2) \log (k/\varepsilon))$ columns of A

such that with probability at least .9

$$\left\| A - CC^{\dagger}A \right\|_F \leq (1+\epsilon) \left\| A - A_k \right\|_F$$



The algorithm

Input: m-by-n matrix A,

$0 < \varepsilon < .5$, the desired accuracy

Output: C, the matrix consisting of the selected columns

Sampling algorithm

- Compute probabilities p_j summing to 1.
- Let $c = O((k/\varepsilon^2) \log (k/\varepsilon))$.
- In c i.i.d. trials pick columns of A, where in each trial the j-th column of A is picked with probability p_j.
- Let C be the matrix consisting of the chosen columns.

Subspace sampling (Frobenius norm)

$$\underset{m \times n}{\begin{pmatrix} \\ A_k \\ \\ \end{pmatrix}} = \underset{m \times k}{\begin{pmatrix} \\ U_k \\ \\ \end{pmatrix}} \cdot \underset{k \times k}{\begin{pmatrix} \\ \Sigma_k \\ \\ \end{pmatrix}} \cdot \underset{k \times n}{\begin{pmatrix} \\ V_k^T \\ \\ \end{pmatrix}}$$

V_k: orthogonal matrix containing the top k right singular vectors of A.

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

Remark: The rows of V_k^T are orthonormal vectors, but its columns $(V_k^T)^{(i)}$ are not.

Subspace sampling (Frobenius norm)

$$\underset{m \times n}{\begin{pmatrix} \\ A_k \\ \\ \end{pmatrix}} = \underset{m \times k}{\begin{pmatrix} \\ U_k \\ \\ \end{pmatrix}} \cdot \underset{k \times k}{\begin{pmatrix} \\ \Sigma_k \\ \\ \end{pmatrix}} \cdot \underset{k \times n}{\begin{pmatrix} \\ V_k^T \\ \\ \end{pmatrix}}$$

V_k: orthogonal matrix containing the top k right singular vectors of A.

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

Remark: The rows of V_k^T are orthonormal vectors, but its columns $(V_k^T)^{(i)}$ are not.

Subspace sampling in $O(mn^2)$ time

$$p_j = \frac{\left\| \left(V_k^T\right)^{(j)} \right\|_2^2}{k}$$

Normalization s.t. the p_j sum up to 1

Subspace sampling (Frobenius norm)

$$\underset{m \times n}{A_k} = \underset{m \times k}{U_k} \cdot \underset{k \times k}{\Sigma_k} \cdot \underset{k \times n}{V_k^T}$$

V_k: orthogonal matrix containing the top k right singular vectors of A.

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

Remark: The rows of V_k^T are orthonormal vectors, but its columns $(V_k^T)^{(i)}$ are not.

Subspace sampling in $O(mn^2)$ time

Leverage scores

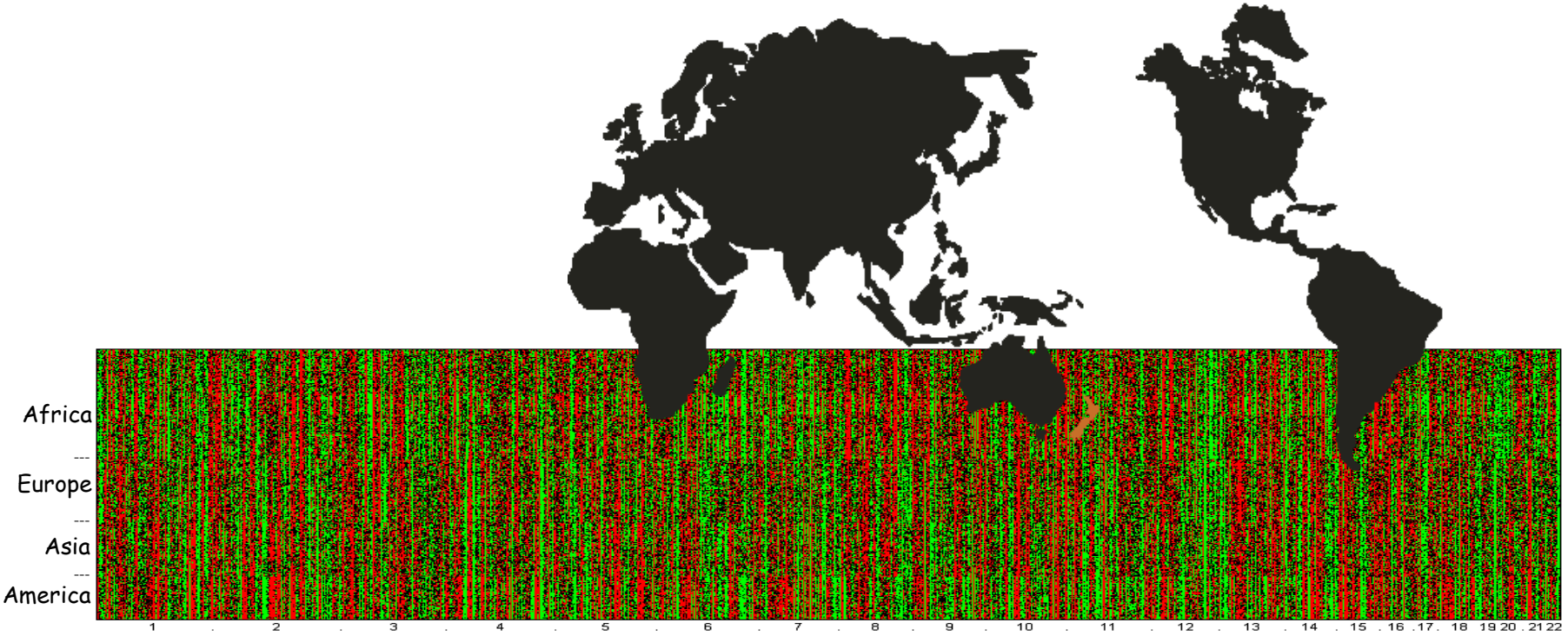
(useful in statistics for outlier detection)

$$p_j = \frac{\left\| (V_k^T)^{(j)} \right\|_2^2}{k}$$

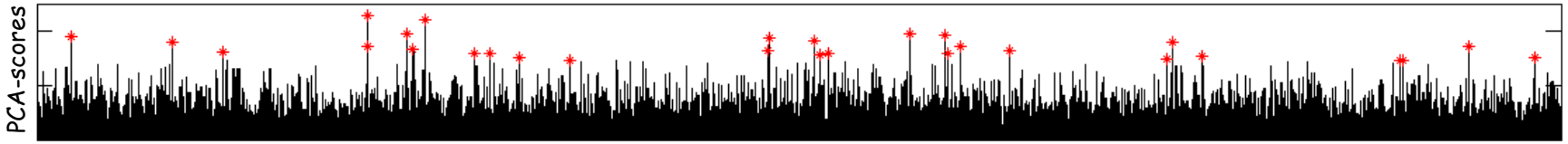
Normalization s.t. the p_j sum up to 1

BACK TO POPULATION GENETICS DATA

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



* top 30 PCA-correlated SNPs



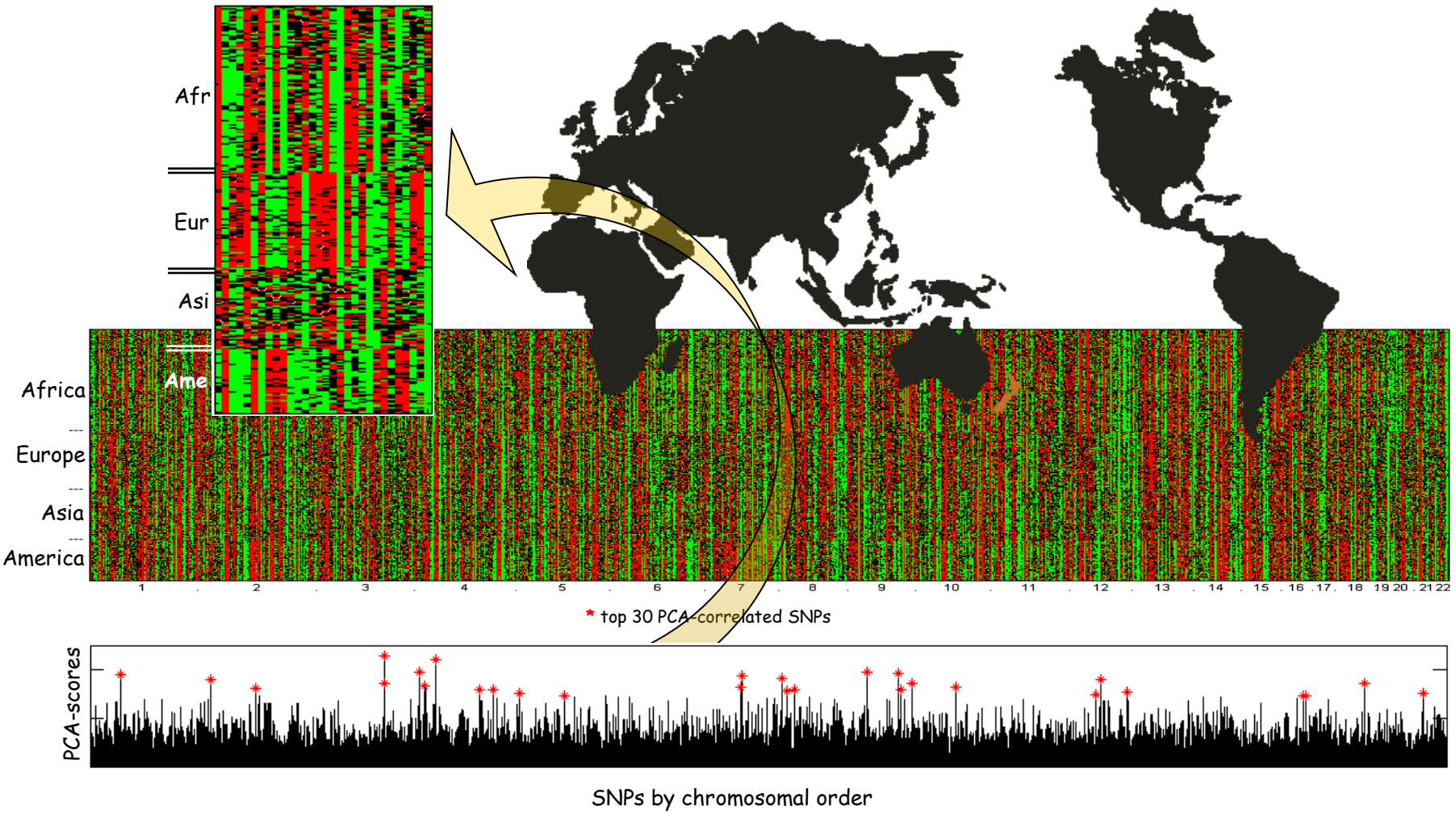
SNPs by chromosomal order

Paschou et al (2007; 2008) PLoS Genetics

Paschou et al (2010) J Med Genet

Drineas et al (2010) PLoS One

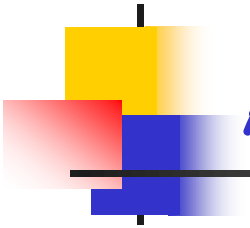
Selecting PCA SNPs for individual assignment to four continents (Africa, Europe, Asia, America)



Paschou et al (2007; 2008) PLoS Genetics

Paschou et al (2010) J Med Genet

Drineas et al (2010) PLoS One



Approximating leverage scores

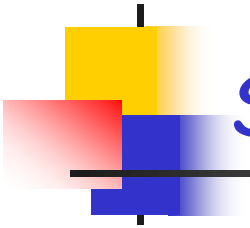
Can we approximate the leverage scores fast?

Theorem: Given any m-by-n matrix A with $m > n$, we can approximate its leverage scores with relative error accuracy in

$$O(mn \log m) \text{ time},$$

as opposed to the – trivial – $O(mn^2)$ time.

(Drineas, Mahoney, Magdon-Ismail, & Woodruff ICML '12)



Selecting fewer columns

Problem

How many columns do we need to include in the matrix C in order to get relative-error approximations ?

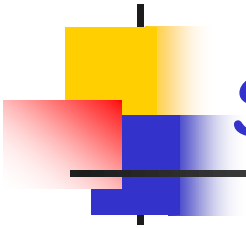
Recall: with $O((k/\varepsilon^2) \log (k/\varepsilon))$ columns, we get (subject to a failure probability)

$$\left\| A - CC^{\dagger}A \right\|_F \leq (1+\epsilon) \left\| A - A_k \right\|_F$$

Deshpande & Rademacher (FOCS '10): with exactly k columns, we get

$$\left\| A - CC^{\dagger}A \right\|_F \leq \sqrt{k} \left\| A - A_k \right\|_F$$

What about the range between k and O(k log(k))?



Selecting fewer columns (cont'd)

(Boutsidis, Drineas, & Magdon-Ismail, FOCS 2011)

Question:

What about the range between k and O(k log(k))?

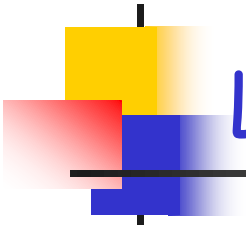
Answer:

A relative-error bound is possible by selecting **$s=3k/\varepsilon$** columns!

Technical breakthrough;

A combination of sampling strategies with a novel approach on column selection, inspired by the work of Batson, Spielman, & Srivastava (STOC '09) on graph sparsifiers.

- **The running time is $O((mnk+nk^3)\varepsilon^{-1})$.**
- **Simplicity is gone…**



Lower bounds and alternative approaches

Deshpande & Vempala, RANDOM 2006

A relative-error approximation necessitates at **least k/ε columns.**

Guruswami & Sinop, SODA 2012

Alternative approaches, based on volume sampling, guarantee

$(r+1)/(r+1-k)$ relative error bounds.

This bound is asymptotically optimal (up to lower order terms).

The proposed deterministic algorithm runs in $O(rnm^3 \log m)$ time, while the randomized algorithm runs in $O(rnm^2)$ time and achieves the bound in expectation.

Guruswami & Sinop, FOCS 2011

Applications of column-based reconstruction in Quadratic Integer Programming.

Random projections: the JL lemma

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

Random projections: the JL lemma

For every set S of m points in $\mathbb{R}^n$ and every $\epsilon > 0$, there exists a mapping $f : \mathbb{R}^n \to \mathbb{R}^s$, where $s = O\left(\log m/\epsilon^2\right)$, 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 R and computing

$$\tilde{A} = AR$$

(The original JL lemma was proven by projecting the points of S to a random k-dimensional subspace.)

Different constructions for R

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

Construct an n-by-s matrix R such that:

$$R_{ij} = \sqrt{3} \times \begin{cases} +1 & ,\text{w.p. } 1/6 \\ 0 & ,\text{w.p. } 2/3 \\ -1 & ,\text{w.p. } 1/6 \end{cases}$$

Return: $\tilde{A} = \frac{1}{\sqrt{s}} AR \in \mathbb{R}^{m \times s}$

$O(mns) = O(mn \log m / \varepsilon^2)$ time computation

Fast JL transform

Ailon & Chazelle (2006) FOCS, Matousek (2006)

$$P \in \mathbb{R}^{s \times n}$$

$$s = O\left(\log m/\epsilon^2\right)$$

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

$$q = O\left(\frac{\log^2 m}{n}\right)$$

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

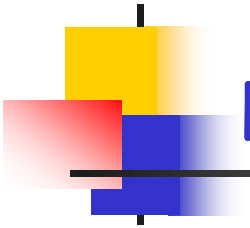
Normalized Hadamard-Walsh transform matrix

(if n is not a power of 2, add all-zero columns to A)

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

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

$$R = (PHD)^T \in \mathbb{R}^{n \times s} \longrightarrow \tilde{A} = \frac{1}{\sqrt{s}} AR$$



Fast JL transform, cont'd

Applying PHD on a vector u *in* R^n is fast, since:

- Du : $O(n)$, since D is diagonal,
- H(Du) : $O(n \log n)$, using the Hadamard-Walsh algorithm,
- P(H(Du)) : $O(\log^3 m/\varepsilon^2)$, since P has on average $O(\log^2 n)$ non-zeros per row (in expectation).

Back to approximating singular vectors

(Drineas, Mahoney, Muthukrishnan, & Sarlos Num Math 2011)

Let A by an m-by-n matrix whose SVD is:

$$A = U\Sigma V^T \in \mathbb{R}^{m\times n}$$

Apply the (HD) part of the (PHD) transform to A.

$$ADH = U\Sigma \underbrace{\left(V^T DH\right)}_{\text{orthogonal matrix}} \in \mathbb{R}^{m\times n}$$

Observations:

1. The left singular vectors of ADH span the same space as the left singular vectors of A.
2. The matrix ADH has (up to log n factors) uniform leverage scores .

   (Thanks to $V^T HD$ having bounded entries – the proof closely follows JL-type proofs.)
3. We can approximate the left singular vectors of ADH (and thus the left singular vectors of A) by uniformly sampling columns of ADH.

Back to approximating singular vectors

(Drineas, Mahoney, Muthukrishnan, & Sarlos Num Math 2011)

Let A by an m-by-n matrix whose SVD is:

$$A = U\Sigma V^T \in \mathbb{R}^{m\times n}$$

Apply the (HD) part of the (PHD) transform to A.

$$ADH = U\Sigma \underbrace{\left(V^T DH\right)}_{\text{orthogonal matrix}} \in \mathbb{R}^{m\times n}$$

Observations:

1. The left singular vectors of ADH span the same space as the left singular vectors of A.
2. The matrix ADH has (up to log n factors) uniform leverage scores .

   (Thanks to $V^T HD$ having bounded entries – the proof closely follows JL-type proofs.)
3. We can approximate the left singular vectors of ADH (and thus the left singular vectors of A) by uniformly sampling columns of ADH.
4. The orthonormality of HD and a version of our relative-error Frobenius norm bound (involving approximately optimal sampling probabilities) suffice to show that (w.h.p.)

$$\left\|A - \tilde{C}\tilde{C}^{\dagger}A\right\|_F \leq (1+\epsilon)\left\|A - A_k\right\|_F$$

Uniform sample of $s = O\left(\frac{k}{\epsilon^2}\log^{c_0}\frac{n}{\epsilon}\right)$ columns of ADH

Running time

Let A by an m-by-n matrix whose SVD is:

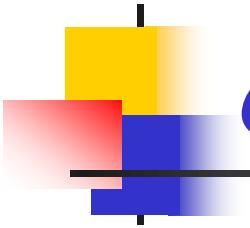
$$A = U\Sigma V^T \in \mathbb{R}^{m\times n}$$

Apply the (HD) part of the (PHD) transform to A.

$$ADH = U\Sigma \left(\underbrace{V^T DH}_{\text{orthogonal matrix}}\right) \in \mathbb{R}^{m\times n}$$

Running time:

1. Trivial analysis: first, uniformly sample s columns of DH and then compute their product with A.
   Takes O(mns) = O(mnk polylog(n)) time, already better than full SVD.
2. Less trivial analysis: take advantage of the fact that H is a Hadamard-Walsh matrix
   Improves the running time O(mn polylog(n) + mk^2polylog(n)).



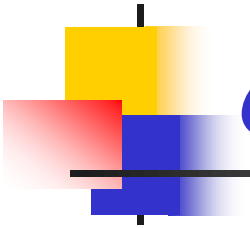
Conclusions

• Randomization and sampling can be used to solve problems that are massive and/or computationally expensive.

• By (carefully) sampling rows/columns/**entries** of a matrix, we can construct new sparse/smaller matrices that behave like the original matrix.

- Can entry-wise sampling be made competitive to column-sampling in terms of accuracy and speed?
  See Achlioptas and McSherry (2001) STOC, (2007) JACM.
- We improved/generalized/simplified it .
  See Nguyen, Drineas, & Tran (2011), Drineas & Zouzias (2010).
- Exact reconstruction possible using uniform sampling for constant-rank matrices that satisfy certain (strong) assumptions.
  See Candes & Recht (2008), Candes & Tao (2009), Recht (2009).

• By preprocessing the matrix using random projections, we can sample rows/ columns much less carefully (even uniformly at random) and still get nice "behavior".



Conclusions

• Randomization and sampling can be used to solve problems that are massive and/or computationally expensive.

• By (carefully) sampling rows/columns/**entries** of a matrix, we can construct new sparse/smaller matrices that behave like the original matrix.

  • Can entry-wise sampling be made competitive to column-sampling in terms of accuracy and speed?
  See Achlioptas and McSherry (2001) STOC, (2007) JACM.
  • We improved/generalized/simplified it .
  See Nguyen, Drineas, & Tran (2011), Drineas & Zouzias (2010).
  • Exact reconstruction possible using uniform sampling for constant-rank matrices that satisfy certain (strong) assumptions.
  See Candes & Recht (2008), Candes & Tao (2009), Recht (2009).

• By preprocessing the matrix using random projections, we can sample rows/ columns much less carefully (even uniformly at random) and still get nice "behavior".