# RandNLA:
Randomization in Numerical Linear Algebra

Michael W. Mahoney

University of California at Berkeley

These lecture slides are substantially the same as used by P. Drineas at the
2016 PMCI Summer School on "Mathematics of Data"

Visit http://www.drineas.org/RandNLA-PCMI-2016/ for slides, TA sessions, etc.

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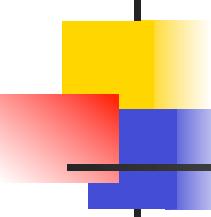
Randomization and sampling allow us to design provably accurate algorithms for problems that are:

➤ Massive

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

➤ Computationally expensive or NP-hard

(combinatorial optimization problems, such as the Column Subset Selection Problem)

RandNLA in a slide

Randomized algorithms

• By (carefully) sampling rows/columns of a matrix, we can construct new, smaller matrices that are close to the original matrix (w.r.t. matrix norms) with high probability.
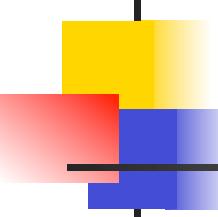
Example:
Randomized
Matrix
Multiplication

$$ A \cdot B \approx C \cdot R $$

• By preprocessing the matrix using "random projection" matrices , we can sample rows/columns much less carefully (uniformly at random) and still get nice bounds with high probability.

Matrix perturbation theory

• The resulting smaller matrices behave similarly (e.g., in terms of singular values and singular vectors) to the original matrices thanks to the norm bounds.

Interplay

Applications in BIG DATA
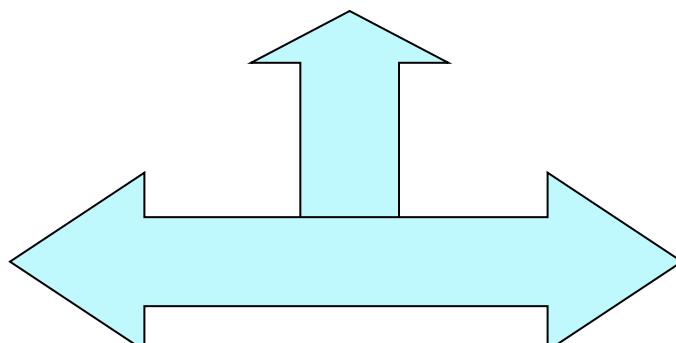
(Data Mining, Information Retrieval, Machine Learning, Bioinformatics, etc.)
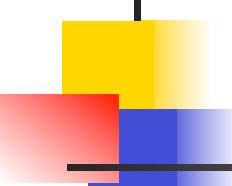
Theoretical Computer Science

Randomized and approximation algorithms

Numerical Linear Algebra

Matrix computations and linear algebra (ie., perturbation theory)
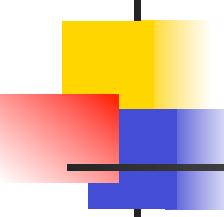
Roadmap of my lectures

- **Approximating matrix multiplication** (first lecture)

- **Leverage scores and their applications** (second and third lectures)

  1. Over- (or under-) constrained least squares problems

  2. Feature selection and the CX decomposition

- **Solving systems of linear equations with Laplacian matrices** (fourth lecture)

- **Element-wise sampling** (fourth lecture)

Approximating Matrix Multiplication

<u>Problem Statement</u>

Given an m-by-n matrix A and an n-by-p matrix B, approximate the product A·B,

OR, equivalently,
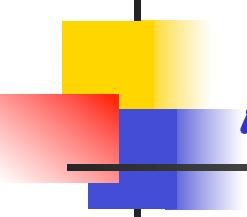
Approximate the sum of n rank-one matrices.

$$A \cdot B = \sum_{i=1}^{n} \left(A_{*i} \right) \cdot \left(B_{i*} \right)$$

$\in \mathbb{R}^{m \times p}$

i-th column of A

i-th row of B

<u>Each term in the summation is a rank-one matrix</u>

A sampling approach

$$A \cdot B = \sum_{i=1}^{n} \underbrace{\left(A_{*i} \right) \cdot \left(B_{i*} \right)}_{\in \mathbb{R}^{m \times p}}$$

i-th row of B

i-th column of A

Algorithm

1. Fix a set of probabilities p_i, i =1...n, summing up to one.

2. For t = 1...c,

   set j_t = i, where $Pr(j_t = i) = p_i$.

   (Pick c terms of the sum, with replacement, with respect to the p_i.)

3. Approximate the product AB by summing the c terms, after scaling.

Sampling (cont'd)

$$A \cdot B = \sum_{i=1}^{n} \begin{pmatrix} A_{*i} \end{pmatrix} \cdot \begin{pmatrix} B_{i*} \end{pmatrix}$$

i-th row of B

$\underbrace{\qquad\qquad}_{\in \mathbb{R}^{m \times p}}$

i-th column of A

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

$\underbrace{\qquad\qquad}_{\in \mathbb{R}^{m \times p}}$

Keeping the terms
$j_1, j_2, \cdots j_c$.

Sampling (alternative formulation)

$$A \cdot B = \sum_{i=1}^{n} \left(A_{*i} \right) \cdot \left(\quad B_{i*} \quad \right)$$

$\in \mathbb{R}^{m \times p}$

i-th row of B

i-th column of A

$$\approx \frac{1}{c} \sum_{i=1}^{n} \frac{c_i}{p_i} \left(A_{*i} \right) \cdot \left(\quad B_{i*} \quad \right)$$

$\in \mathbb{R}^{m \times p}$

The c_i (for all i=1...n) are random variables indicating how many times the i-th column-row pair was sampled.

9

Sampling (alternative formulation)

$$A \cdot B = \sum_{i=1}^{n} \begin{pmatrix} A_{*i} \end{pmatrix} \cdot \begin{pmatrix} B_{i*} \end{pmatrix}$$

$$\underbrace{\phantom{\begin{pmatrix} A_{*i} \end{pmatrix} \cdot \begin{pmatrix} B_{i*} \end{pmatrix}}}_{\in \mathbb{R}^{m \times p}}$$

i-th row of B

i-th column of A

$$\approx \frac{1}{c} \sum_{i=1}^{n} \frac{c_i}{p_i} \begin{pmatrix} A_{*i} \end{pmatrix} \cdot \begin{pmatrix} B_{i*} \end{pmatrix}$$

$$\underbrace{\phantom{\begin{pmatrix} A_{*i} \end{pmatrix} \cdot \begin{pmatrix} B_{i*} \end{pmatrix}}}_{\in \mathbb{R}^{m \times p}}$$

For most i, c_i will be zero; notice that all possible values for c_i are 0, 1, 2, ... c.

10

Sampling (alternative formulation)

$$A \cdot B = \sum_{i=1}^{n} \left(A_{*i} \right) \cdot \left(B_{i*} \right)$$

$\in \mathbb{R}^{m \times p}$

i-th row of B

i-th column of A

$$\approx \frac{1}{c} \sum_{i=1}^{n} \frac{c_i}{p_i} \left(A_{*i} \right) \cdot \left(B_{i*} \right)$$

$\in \mathbb{R}^{m \times p}$

Prove: the expectation of c_i is equal to cp_i.

This is the simplest way to see why (algebraically) we need to rescale by $1/cp_i$.

The algorithm (matrix notation)

$$
\left(\begin{array}{c} A \end{array} \right) \cdot \left(\begin{array}{c} B \end{array} \right) \approx \left(\begin{array}{c} C \end{array} \right) \cdot \left(\begin{array}{c} R \end{array} \right)
$$

$$
\underset{m \times n}{} \qquad \underset{n \times p}{} \qquad \underset{m \times c}{} \qquad \underset{c \times p}{}
$$

Algorithm

1. Pick c columns of A to form an m-by-c matrix C and the corresponding c rows of B to form a c-by-p matrix R.

2. Approximate $A \cdot B$ by $C \cdot R$.

Notes

3. We pick the columns and rows with non-uniform probabilities.

4. We scale the columns (rows) prior to including them in C (R).
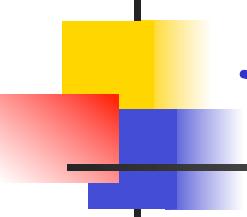
The algorithm (matrix notation, cont'd)

$$\left(\begin{array}{c} A \\ m \times n \end{array} \right) \cdot \left(\begin{array}{c} B \\ n \times p \end{array} \right) \approx \left(\begin{array}{c} C \\ m \times c \end{array} \right) \cdot \left(\begin{array}{c} R \\ c \times p \end{array} \right)$$

- Create C and R by performing c i.i.d. trials, with replacement.

- For t = 1...c, pick a column $A^{(j_t)}$ and a row $B_{(j_t)}$ with probability

$$\Pr\left(j_t = i\right) = \frac{\|A_{*i}\|_2 \, \|B_{i*}\|_2}{\sum_{j=1}^{n} \|A_{*j}\|_2 \, \|B_{j*}\|_2}$$

- Include $A_{*j_t}/(cp_{j_t})^{1/2}$ as a column of C, and $B_{j_t*}/(cp_{j_t})^{1/2}$ as a row of R.

The algorithm (matrix notation, cont'd)
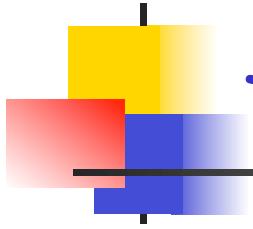
We can also use the sampling matrix notation:

Let S be an n-by-c matrix whose t-th column (for t = 1...c) has a single non-zero entry, namely

$$S_{jtt} = \frac{1}{\sqrt{cp_{jt}}}$$

Clearly:

$$A \cdot B \approx C \cdot R = (AS) \cdot (S^T B)$$
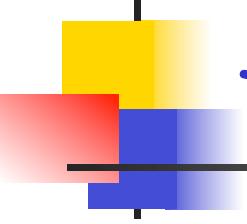
The algorithm (matrix notation, cont'd)

We can also use the sampling matrix notation:

Let S be an n-by-c matrix whose t-th column (for t = 1...c) has a single non-zero entry, namely

$$S_{jtt} = \frac{1}{\sqrt{cp_{jt}}}$$

rescaling factor
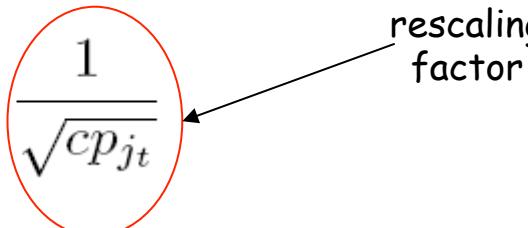
Clearly:

$$A \cdot B \approx C \cdot R = (AS) \cdot (S^T B)$$

The algorithm (matrix notation, cont'd)

We can also use the sampling matrix notation:

Let S be an n-by-c matrix whose t-th column (for t = 1...c) has a single non-zero entry, namely

$$S_{jtt} = \frac{1}{\sqrt{cp_{jt}}}$$

rescaling factor

Clearly:

$$A \cdot B \approx C \cdot R = (AS) \cdot (S^T B)$$

Remark 1: S is sparse (has exactly c non-zero elements, one per column).

Remark 2: In some cases, we express S as a product of a sampling matrix (exactly as described above, but with the non-zero entries being equal to one) and a rescaling c-by-c diagonal matrix whose entries are equal to the rescaling factors.

16

Simple Lemmas
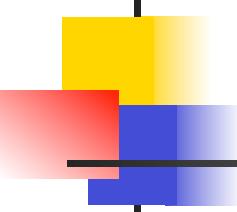
- It is easy to implement this particular sampling in two passes, when the input matrices are given as a sequence of triplets (i, j, A_{ij}).

- The expectation of CR (element-wise) is AB (unbiased estimator), regardless of the sampling probabilities, i.e.,

$$\mathbf{E}\left[(CR)_{ij}\right] = (AB)_{ij}$$

- We can also bound the variance of the (i, j)-th entry of CR:

$$\mathbf{Var}\left[(CR)_{ij}\right] = \frac{1}{c}\sum_{k=1}^{n}\frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{c}(AB)_{ij}^2$$

Simple Lemmas
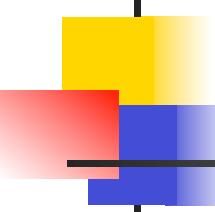
$$\mathbf{E}\left[(CR)_{ij}\right] = (AB)_{ij}$$

$$\mathbf{Var}\left[(CR)_{ij}\right] = \frac{1}{c}\sum_{k=1}^{n}\frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{c}(AB)_{ij}^2$$

The above two bounds can now be used to bound

$$\mathbf{E}\left[\|AB - CR\|_F^2\right]$$

Our particular choice of sampling probabilities **minimizes the above expectation.**
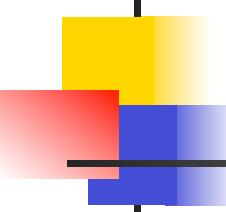
A bound for the Frobenius norm

For the above algorithm,

$$\mathbf{E}\,\|AB - CR\|_F = \mathbf{E}\,\|AB - ASS^T B\|_F \leq \frac{1}{\sqrt{c}}\,\|A\|_F\,\|B\|_F$$

- We can now use Markov's inequality to directly get a "constant probability" bound.

- "High probability" follows from a martingale argument (we will skip this discussion).

- The above bound immediately implies an upper bound for the spectral norm of the error AB-CR (but better bounds can be derived).

A note on the sampling probabilities
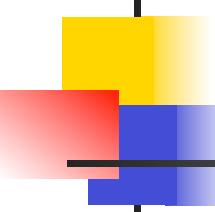
Recall the sampling probabilities:

$$\Pr\left(j_t = i\right) = \frac{\|A_{*i}\|_2 \, \|B_{i*}\|_2}{\sum_{j=1}^{n} \|A_{*j}\|_2 \, \|B_{j*}\|_2}$$

- It turns out that the aforementioned bound (in expectation) holds even if the sampling probabilities have the following form:

$$\Pr\left(j_t = i\right) \geq \frac{\beta \, \|A_{*i}\|_2 \, \|B_{i*}\|_2}{\sum_{j=1}^{n} \|A_{*j}\|_2 \, \|B_{j*}\|_2}$$

- The probabilities still need to sum up to one.

- In the above, β is a constant in the interval (0,1].

- In the upper bound of the expectation, c is replaced by βc.

A note on the sampling probabilities
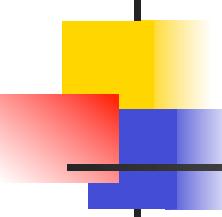
Recall the sampling probabilities:

$$\Pr\left(j_t = i\right) = \frac{\|A_{*i}\|_2 \, \|B_{i*}\|_2}{\sum_{j=1}^{n} \|A_{*j}\|_2 \, \|B_{j*}\|_2}$$

- The aforementioned bound (in expectation) holds even if the sampling probabilities depend only on **A** or only on **B**:

$$\Pr\left(j_t = i\right) \geq \frac{\beta \, \|A_{*i}\|_2^2}{\|A\|_F^2} \qquad \text{or} \qquad \Pr\left(j_t = i\right) \geq \frac{\beta \, \|B_{i*}\|_2^2}{\|B\|_F^2}$$

- In the above, β is a constant in the interval (0,1].

- In the upper bound of the expectation, c is replaced by βc.

- The martingale argument though does not work any more...

21
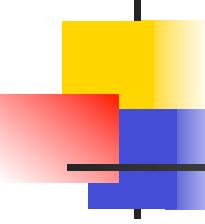
Special case: $B = A^\top$

If $B = A^\top$, then the sampling probabilities are

$$\Pr\left(j_t = i\right) = \frac{\|A_{*i}\|_2^2}{\sum_{j=1}^n \|A_{*j}\|_2^2} = \frac{\|A_{*i}\|_2^2}{\|A\|_F^2}$$

Also, $R = C^\top$, and the error bounds are:

$$\mathbf{E}\left\|AA^T - CC^T\right\|_F = \mathbf{E}\left\|AA^T - ASS^T A^T\right\|_F \leq \frac{1}{\sqrt{c}}\|A\|_F^2$$

Special case: B = A$^\top$ (cont'd)

(Drineas et al. Num Math 2011, Theorem 4)

A better **spectral norm** bound via matrix Chernoff/Bernstein inequalities:
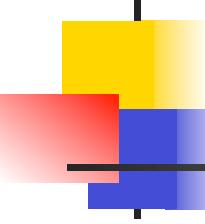
Assumptions:

- Spectral norm of A is at most one (not important, just normalization)

- Frobenius norm of A is at least 0.2 (not important, simplifies bounds).

- **Important**: Set

$$c = \Omega \left(\frac{\|A\|_F^2}{\epsilon^2} \ln \left(\frac{\|A\|_F^2}{\epsilon^2 \sqrt{\delta}} \right) \right)$$

Then: for any $0 < \varepsilon < 1$, with probability at least $1 - \delta$,

$$\left\| AA^T - CC^T \right\|_2 = \left\| AA^T - ASS^T A^T \right\|_2 \leq \varepsilon$$

Proof outline

Define the random vector y such that
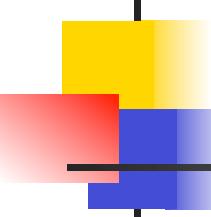
$$\Pr\left(y = \frac{1}{\sqrt{p_i}} A_{*i}\right) = p_i$$

Now, it is easy to see that the matrix C has columns

$$\frac{1}{\sqrt{c}} y^1, \frac{1}{\sqrt{c}} y^2, \ldots, \frac{1}{\sqrt{c}} y^c$$

where $y^1, y^2, \ldots, y^c$ are independent copies of y. It is easy to prove that

$$\mathbf{E}\left[yy^T\right] = AA^T \quad \text{and} \quad CC^T = ASS^T A^T = \frac{1}{c}\sum_{t=1}^{c} y^t y^{tT}.$$

Proof outline

We can now upper bound the norm of the vector y:

$$M = \|y\|_2 = \frac{1}{\sqrt{p_i}}\|A_{*i}\|_2$$

Also, given our assumption that the spectral norm of A is at most one, we get:

$$\left\|\mathbf{E}\left[yy^T\right]\right\|_2 = \left\|AA^T\right\|_2 \leq \|A\|_2 \left\|A^T\right\|_2 \leq 1.$$

We can now apply a matrix-Bernstein inequality (Lemma 1 of Oliveira 2010) to get the desired bound.

An inequality by Oliveira

Let $y^1, y^2, ..., y^c$ be independent identically distributed copies of the m-dimensional random vector y with
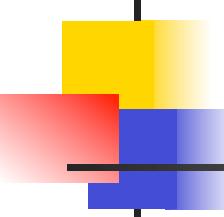
$$\|y\|_2 \leq M \quad \text{and} \quad \left\|\mathbf{E}\left(yy^T\right)\right\|_2 \leq 1$$

Then, for any $a > 0$,

$$\left\|\frac{1}{c}\sum_{i=1}^{c} y^i y^{iT} - \mathbf{E}\left(yy^T\right)\right\|_2 \leq a$$

holds with probability at least
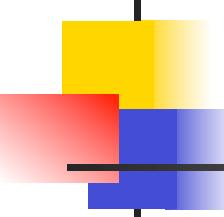
$$1 - (2c)^2 \exp\left(-\frac{ca^2}{16M^2 + 8M^2 a}\right)$$

Special case: B = $A^\top$ (cont'd)

<u>Remarks:</u>

- The proof is relatively simple, given the matrix-Bernstein inequality.

- Even better bounds (in terms of constants) can be derived using tighter inequalities; see Ipsen & Wentworth (2014) SIMAX.

- We will now take a look at alternative mathematical tools that could be used to prove such spectral norm bounds.
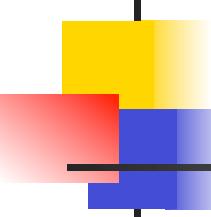
Bounding the spectral norm of random matrices

➢ The simplest approach (often very useful) is to bound the Frobenius norm.

➢ Today, the most common alternative is to apply matrix-Bernstein inequalities.

  Check the review by J. Tropp (2015) "An introduction to matrix concentration inequalities".

➢ A tougher approach is to use the moments method, dating back to Wigner (1967) SIREV: for symmetric matrices A, compute the trace of the matrix A^k for large (even) values of k and bound its expectation. Then,

$$\text{Trace}\left(A^k\right) = \sum_{i=1}^{n} \lambda_i^k \geq \lambda_1^k = \|A\|_2^k$$

- The "problem" with this approach is that it typically boils down to tough path-counting combinatorial arguments on graphs.

- But, it can sometimes result in tighter bounds than matrix-Bernstein inequalities for certain problems.

  Check Furedi & Komlos (1981) Combinatorica, Nelson & Huy FOCS 2013.
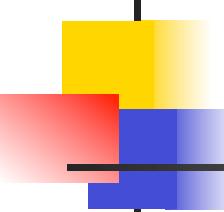
Bounding the spectral norm of random matrices

➢ An even more complicated alternative is the entropy-concentration method, developed by M. Rudelson & R. Virshynin.

  - Given a random matrix A, in order to bound its spectral norm, we could bound

$$\max_{x:\|x\|_2=1,\,y:\|y\|_2=1} \left| y^T A x \right|$$

Bounding the spectral norm of random matrices

➢ An even more complicated alternative is the entropy-concentration method, developed by M. Rudelson & R. Virshynin.

- Given a random matrix A, in order to bound its spectral norm, we could bound

$$\max_{x:\|x\|_2=1,\, y:\|y\|_2=1} \left| y^T A x \right|$$

- **<u>Theoretical Computer Science approach</u>:** use a so-called ε-net argument on the unit sphere (e.g., discretize the unit sphere and compute bounds over all possible vectors). However, this simple technique could fail, mainly because one cannot treat all vectors in the (discretized) unit sphere the same.

Bounding the spectral norm of random matrices

➢ An even more complicated alternative is the entropy-concentration method, developed by M. Rudelson & R. Virshynin.

- Given a random matrix *A*, in order to bound its spectral norm, we could bound

$$\max_{x:\|x\|_2=1,\, y:\|y\|_2=1} \left| y^T A x \right|$$

- **Theoretical Computer Science approach:** use a so-called ε-net argument on the unit sphere (e.g., discretize the unit sphere and compute bounds over all possible vectors). However, this simple technique could fail, mainly because one cannot treat all vectors in the (discretized) unit sphere the same.

- Rudelson and Virshynin proposed to express all vectors in the unit sphere as **a sum of two vectors:** a **sparse vector** with a **bounded number of non-zeros** (but arbitrarily large entries) and a **spread vector**, whose entries have **restricted magnitudes**.

- Now the unit sphere can be split in two sets and we can build two ε-net arguments.
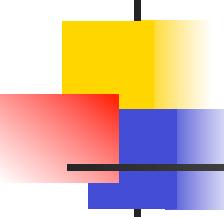
31

Bounding the spectral norm of random matrices

➢ An even more complicated alternative is the entropy-concentration method, developed by M. Rudelson & R. Virshynin.

- Given a random matrix **A**, in order to bound its spectral norm, we could bound

$$\max_{x:\|x\|_2=1,\, y:\|y\|_2=1} \left| y^T A x \right|$$

- Their method gave state-of-the-art bounds for the smallest singular value of rectangular matrices (Rudelson & Virshynin (2009) Comm Pure App Math).

- I also used this method in a tensor sparsification paper (with Nam Nguyen); was not fun…

- The original proof of the famous input sparsity time random projection method of Clarkson and Woodruff STOC 2013 is (at least in my eyes) reminiscent of this approach.
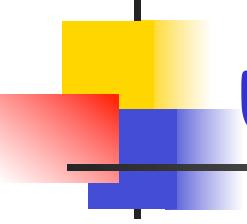
Bounding the spectral norm of random matrices

➢ Finally, some of the original proofs for the spectral norm bound used the Khintchine inequality and so-called symmetrization arguments.

- • At least within the context of RandNLA, I am not aware of applications of such inequalities that improve current state-of-the-art.

- • They are harder to apply and they (typically) involve unknown constants.

For more details on measure concentration techniques for random matrices (and their applications) follow Roman Vershyinin's mini-course.

Using a dense S (instead of a sampling matrix...)
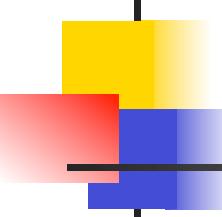
We approximated the product AB as follows:

$$A \cdot B \approx C \cdot R = (AS) \cdot (S^T B)$$

Recall that S is an n-by-c sparse matrix (one non-zero entry per column).

Let's replace S by a dense matrix, the random sign matrix:

$$S_{ij} = \begin{cases} +1/\sqrt{c} & \text{,w.p. } 1/2 \\ -1/\sqrt{c} & \text{,w.p. } 1/2 \end{cases}$$

Using a dense S (instead of a sampling matrix...)

We approximated the product AB as follows:

$$A \cdot B \approx C \cdot R = (AS) \cdot (S^T B)$$

Recall that S is an n-by-c sparse matrix (one non-zero entry per column).

Let's replace S by a dense matrix, the random sign matrix:

$$S_{ij} = \begin{cases} +1/\sqrt{c} & ,\text{w.p. } 1/2 \\ -1/\sqrt{c} & ,\text{w.p. } 1/2 \end{cases}$$
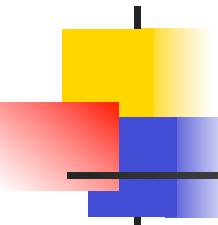
st(A): stable rank of A

$$\text{st}(A) = \|A\|_F^2 / \|A\|_2^2$$

If

$$c = \Omega\left(\max\{\text{st}(A), \text{st}(B)\} \ln(m+p) / \epsilon^2\right)$$

then, with high probability (see Theorem 3.1 in Magen & Zouzias SODA 2012)

$$\|AB - CR\|_2 = \left\|AB - ASS^T B\right\|_2 \leq \varepsilon \|A\|_2 \|B\|_2$$

35

Using a dense S (instead of a sampling matrix…)
(and focusing on B = $A^\top$, normalized)

Approximate the product AA^T (assuming that the spectral norm of **A** is one):

$$A \cdot A^T \approx C \cdot C^T = (AS) \cdot (S^T A^T)$$

Let S by a dense matrix, the random sign matrix:

$$S_{ij} = \begin{cases} +1/\sqrt{c} & \text{,w.p. } 1/2 \\ -1/\sqrt{c} & \text{,w.p. } 1/2 \end{cases}$$

If

$$c = \Omega \left(\frac{\|A\|_F^2}{\epsilon^2} \ln m \right)$$

then, with high probability:

$$\left\| AA^T - CC^T \right\|_2 = \left\| AA^T - ASS^T A^T \right\|_2 \leq \varepsilon$$

Using a dense S (instead of a sampling matrix...)
(and focusing on B = A^T, normalized)

Approximate the product AA^T (assuming that the spectral norm of **A** is one):

$$A \cdot A^T \approx C \cdot C^T = (AS) \cdot (S^T A^T)$$

Let S by a dense matrix, the random sign matrix:

$$S_{ij} = \begin{cases} +1/\sqrt{c} & \text{,w.p. } 1/2 \\ -1/\sqrt{c} & \text{,w.p. } 1/2 \end{cases}$$
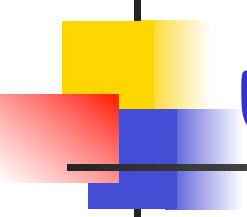
If

$$c = \Omega \left(\frac{\|A\|_F^2}{\epsilon^2} \ln m \right)$$

Similar structure with the sparse S case; some differences in the ln factor

then, with high probability:
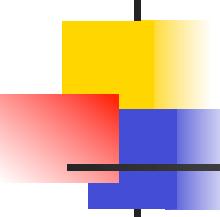
$$\left\| AA^T - CC^T \right\|_2 = \left\| AA^T - ASS^T A^T \right\|_2 \leq \varepsilon$$

Using a dense S (cont'd)

Comments:

- **This matrix multiplication approximation is oblivious to the input matrices A and B.**

- Reminiscent of random projections and the Johnson-Lindenstrauss (JL) transform.

- Bounds for the Frobenius norm are easier to prove and are very similar to the case where S is just a sampling matrix.

- It holds for arbitrary A and B (not just $B = A^T$); the sampling-based approach should also be generalizable to arbitrary A and B.

Using a dense S (cont'd)

Other choices for dense matrices S?

Why bother with a sign matrix?

(Computing the product AS and $S^T B$ is somewhat slow, taking $O(mnc)$ and $O(pnc)$ time.)

Similar bounds are known for better, i.e., computationally more efficient, choices of "random projection" matrices S, most notably:

- When S is the so-called subsampled Hadamard Transform Matrix.

(much faster; avoids full matrix-matrix multiplication; see Sarlos FOCS 2006 and Drineas et al. (2011) Num Math)

- When S is the input sparsity time projection matrix of Clarkson & Woodruff STOC 2013.
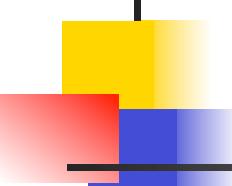
(the matrix multiplication result appears in Mahoney & Meng STOC 2013 and was improved by Nelson and Huy FOCS 2013).

TA session II

TA session II will focus on randomized matrix multiplication and is posted at
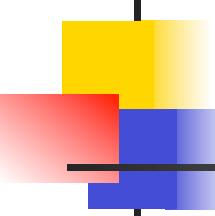
http://www.drineas.org/RandNLA-PCMI-2016/

Roadmap of my lectures

➢ Approximating matrix multiplication (first lecture)

➢ **Leverage scores and their applications** (second and third lectures)

      1. Over- (or under-) constrained least squares problems

      2. Feature selection and the CX decomposition

➢ Solving systems of linear equations with Laplacian matrices (fourth lecture)

➢ Element-wise sampling (fourth lecture)

RandNLA:
Randomized Numerical Linear Algebra

Sampling rows (or columns) from a matrix

Input: m-by-n matrix A, sampling parameter r

Output: r-by-n matrix R, consisting of r rows of A

- Let p_i for i=1...m be sampling probabilities summing up to 1;

- In r i.i.d. trials (with replacement) pick r rows of A;

   (In each trial the i-th row of A is picked with probability p_i.)

- Let R be the matrix consisting of the rows;

   (We rescale the rows of A prior to including them in R by $1/(rp_i)^{1/2}$.)

$$
\begin{pmatrix} & & \\ & A & \\ & & \end{pmatrix} \Longrightarrow \begin{pmatrix} & R & \end{pmatrix}
$$

$m \times n \qquad\qquad r \times n$

The p_i's: length-squared sampling

Length-squared sampling: sample rows with probability proportional to the square of their Euclidean norms, i.e.,

$$p_i = \frac{\|A_{i*}\|_2^2}{\|A\|_F^2}$$

Notation:
A_{i*} : the i-th row of A
$||A||_F$: the Frobenius norm of A

The p_i's: length-squared sampling

Length-squared sampling: sample rows with probability proportional to the square of their Euclidean norms, i.e.,
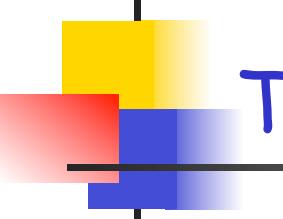
$$p_i = \frac{\|A_{i*}\|_2^2}{\|A\|_F^2}$$

Notation:
A_{i*} : the i-th row of A
$||A||_F$: the Frobenius norm of A

Leads to additive-error approximations for

■ low-rank matrix approximations and the Singular Value Decomposition (SVD),

■ the CUR and CX factorizations,

■ the Nystrom method, etc.

(Drineas, Kannan, Mahoney SICOMP 2006a, SICOMP 2006b, SICOMP 2006c, Drineas & Mahoney JMLR 2005, etc.)

The p_i's: leverage scores

Leverage score sampling: sample rows with probability proportional to the square of the Euclidean norms of the rows of the top k **left singular vectors** of A.

$$p_i = \frac{\|(U_k)_{i*}\|_2^2}{\|U_k\|_F^2} = \frac{\|(U_k)_{i*}\|_2^2}{k}$$

Notation:

U_k: the m-by-k matrix containing the top k left singular vectors of A

$(U_k)_{i*}$: the i-th row of U_k

$k=\|U_k\|_F$: the Frobenius norm of U_k

Leverage score sampling: sample rows with probability proportional to the square of the Euclidean norms of the rows of the top k **left singular vectors** of A.

$$p_i = \frac{\|(U_k)_{i*}\|_2^2}{\|U_k\|_F^2} = \frac{\|(U_k)_{i*}\|_2^2}{k}$$

Notation:
U$_k$: the m-by-k matrix containing the top k left singular vectors of A
(U$_k$)$_{i*}$: the i-th row of U$_k$
k=||U$_k$||$_F$: the Frobenius norm of U$_k$

$$\left(\begin{array}{c} U_k \\ \\ m \times k \end{array} \right) \quad \textbf{and} \quad \left(\begin{array}{c} A \\ \\ m \times n \end{array} \right) \quad \Longrightarrow \quad \left(\begin{array}{c} R \\ r \times n \end{array} \right)$$

The p_i's: leverage scores

Leverage score sampling: sample rows with probability proportional to the square of the Euclidean norms of the rows of the top k **left singular vectors** of A.

$$p_i = \frac{\|(U_k)_{i*}\|_2^2}{\|U_k\|_F^2} = \frac{\|(U_k)_{i*}\|_2^2}{k}$$

Notation:
U_k: the m-by-k matrix containing the top k left singular vectors of A
$(U_k)_{i*}$: the i-th row of U_k
$k = \|U_k\|_F$: the Frobenius norm of U_k

Leads to relative-error approximations for:

- Over- and under- constrained least-squares problems,

- low-rank matrix approximations and the Singular Value Decomposition (SVD),

- relatedly, the CUR and CX factorizations, and

- solving systems of linear equations with Laplacian input matrices

The p_i's: leverage scores

Leverage score sampling: sample rows with probability proportional to the square of the Euclidean norms of the rows of the top k **left singular vectors** of A.
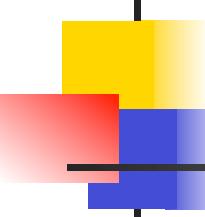
$$p_i = \frac{\|(U_k)_{i*}\|_2^2}{\|U_k\|_F^2} = \frac{\|(U_k)_{i*}\|_2^2}{k}$$

Notation:
U_k: the m-by-k matrix containing the top k left singular vectors of A
$(U_k)_{i*}$: the i-th row of U_k
$k = \|U_k\|_F$: the Frobenius norm of U_k

Column sampling is equivalent to row sampling by focusing on $A^\top$ and looking at its top k left singular vectors...

(Which, of course, are the top k right singular vectors of A, often denoted as V_k , an n-by-k matrix.)

Leverage scores: tall & thin matrices

Let A be a (full rank) n-by-d matrix with n≫d whose SVD is:

$$
\begin{pmatrix} A \end{pmatrix} = \begin{pmatrix} U \end{pmatrix} \begin{pmatrix} \Sigma \end{pmatrix}_{d \times d} \begin{pmatrix} V^T \end{pmatrix}_{d \times d}
$$

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

Leverage scores: tall & thin matrices

Let A be a (full rank) n-by-d matrix with n≫d whose SVD is:

$$\begin{pmatrix} & \\ & A \\ & \\ & \end{pmatrix} = \begin{pmatrix} & \\ & U \\ & \\ & \end{pmatrix} \begin{pmatrix} \Sigma \\ \\ \end{pmatrix} \begin{pmatrix} V^T \\ \\ \end{pmatrix}$$

$$n \times d \quad , \quad n \gg d \qquad n \times d \qquad\qquad d \times d \qquad\qquad d \times d$$
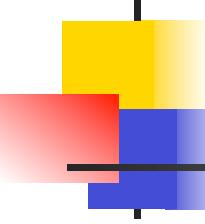
i-th row of U_k

(Row) Leverage scores: $p_i = \dfrac{\|U_{i*}\|_2^2}{\|U\|_F^2} = \dfrac{\|U_{i*}\|_2^2}{d}$
(set k to d)
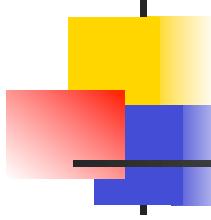
The (row) leverage scores can now be used to sample rows from A to create a sketch.

Leverage scores: short & fat matrices
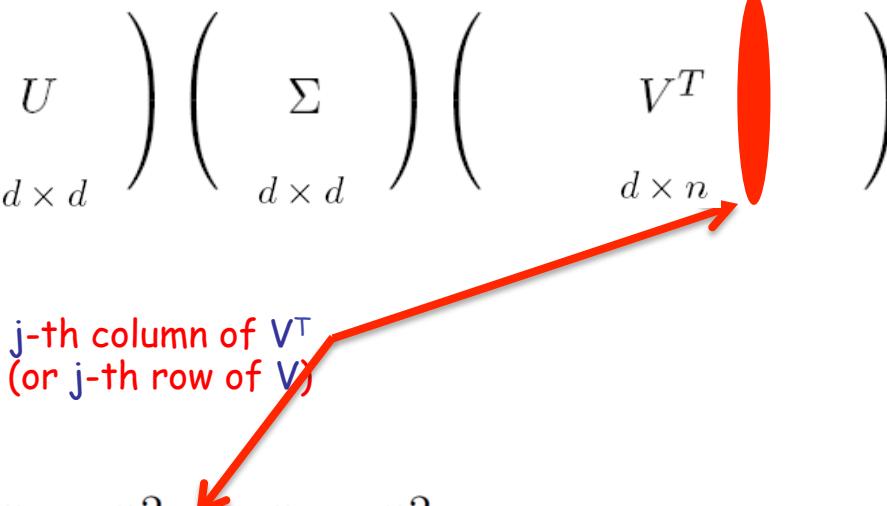
Let A be a (full rank) d-by-n matrix with n≫d:

$$
\begin{pmatrix} & A & \\ & {\scriptstyle d \times n} & \end{pmatrix} = \begin{pmatrix} & U & \\ & {\scriptstyle d \times d} & \end{pmatrix} \begin{pmatrix} & \Sigma & \\ & {\scriptstyle d \times d} & \end{pmatrix} \begin{pmatrix} & V^T & \\ & {\scriptstyle d \times n} & \end{pmatrix}
$$

Leverage scores: short & fat matrices
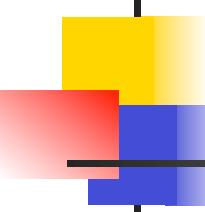
Let A be a (full rank) d-by-n matrix with $n \gg d$:

$$\begin{pmatrix} & A & \\ & d \times n & \end{pmatrix} = \begin{pmatrix} & U & \\ & d \times d & \end{pmatrix} \begin{pmatrix} & \Sigma & \\ & d \times d & \end{pmatrix} \begin{pmatrix} & V^T & \\ & d \times n & \end{pmatrix}$$

j-th column of V^T
(or j-th row of V)

(Column) Leverage scores: $\quad p_j = \dfrac{\|V_{j*}\|_2^2}{\|V\|_F^2} = \dfrac{\|V_{j*}\|_2^2}{d}$

(set k to d)

The (column) leverage scores can now be used to sample columns from A to create a sketch.

Leverage scores: general case

Let A be an m-by-n matrix A and let A_k be its best rank-k approximation (as computed by the SVD) :

$$A \approx \left(\quad A_k \quad \right) = \left(\quad U_k \quad \right) \cdot \left(\quad \Sigma_k \quad \right) \cdot \left(\quad V_k^T \quad \right)$$

$$m \times n \qquad\qquad m \times k \qquad k \times k \qquad\qquad k \times n$$

Leverage scores: general case

Let A be an m-by-n matrix A and let A_k be its best rank-k approximation (as computed by the SVD) :

$$A \approx \left(\begin{array}{c} A_k \end{array} \right) = \left(\begin{array}{c} U_k \end{array} \right) \cdot \left(\begin{array}{c} \Sigma_k \end{array} \right) \cdot \left(\begin{array}{c} V_k^T \end{array} \right)$$

$$m \times n \qquad m \times k \qquad k \times k \qquad k \times n$$

i-th row of U_k j-th column of V_k^T

(Row) Leverage scores: (Column) Leverage scores:

$$p_i = \frac{\|(U_k)_{i*}\|_2^2}{k}$$

$$p_j = \frac{\|(V_k)_{j*}\|_2^2}{k}$$

The (row/column) leverage scores can now be used to sample rows/columns from A.

54

Computing leverage scores

➤ **Trivial:** via the Singular Value Decomposition

$O(nd^2)$ time for n-by-d matrices with n>d.

$O(\min\{m^2n, mn^2\})$ time for general m-by-n matrices.

➤ **Non-trivial:** relative error $(1+\varepsilon)$ approximations for all leverage scores.

Tall & thin matrices (short & fat are similar):

$$
\begin{pmatrix} & & \\ & A & \\ & & \end{pmatrix}
$$

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

Approximating leverage scores:

1. Pre-multiply A by – say – the subsampled Randomized Hadamard Transform matrix (an s-by-n matrix P).

2. Compute the QR decomposition PA = QR.

3. Estimate the lengths of the rows of AR^{-1} (another random projection is used for speed)

55

Computing leverage scores

➢ **Trivial:** via the Singular Value Decomposition

$O(nd^2)$ time for **n**-by-**d** matrices with **n**>**d**.

$O(\min\{m^2n, mn^2\})$ time for general **m**-by-**n** matrices.

➢ **Non-trivial:** relative error **(1+ε)** approximations for all leverage scores.
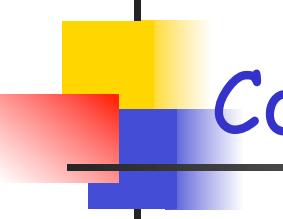
Tall & thin matrices (short & fat are similar):

$$\begin{pmatrix} \\ \\ A \\ \\ \\ \end{pmatrix}$$

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

Running time:

It suffices to set $s = O(d\varepsilon^{-2}\ \text{polylog}(n/\varepsilon))$.

Overall running time is $O(nd\varepsilon^{-2}\ \text{polylog}(n/\varepsilon))$.

Computing leverage scores

➢ **Trivial:** via the Singular Value Decomposition

$O(nd^2)$ time for **n**-by-**d** matrices with **n**>**d**.

$O(\min\{m^2n, mn^2\})$ time for general **m**-by-**n** matrices.

➢ **Non-trivial:** relative error **(1+ε)** approximations for all leverage scores.

m-by-**n** matrices:

$$\begin{pmatrix} & & \\ & A & \\ & & \end{pmatrix}$$
$$m \times n$$

Caution:
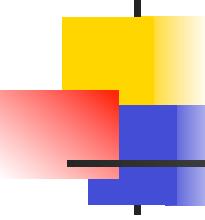
A direct formulation of the problem is ill-posed.

(The **k** and **(k+1)**-st singular values could be very close estimating the corresponding singular vectors could result in a "swap".)

A robust objective is to estimate the leverage scores of some rank **k** matrix **X** that is "close" to the best rank **k** approximation to **A**.

(see Drineas et al. (2012) ICML and JMLR for details)

Computing leverage scores

➤ **Trivial:** via the Singular Value Decomposition

$O(nd^2)$ time for n-by-d matrices with n>d.

$O(\min\{m^2n, mn^2\})$ time for general m-by-n matrices.

➤ **Non-trivial:** relative error $(1+\varepsilon)$ approximations for all leverage scores.

m-by-n matrices:

$$\begin{pmatrix} & & \\ & A & \\ & & \\ & m \times n & \end{pmatrix}$$

Algorithm:

➤ Approximate the top k left (or right) singular vectors of A.

➤ Use the approximations to estimate the leverage scores.

Overall running time is $r = O(mnk\varepsilon^{-2}\ \text{polylog}(n/\varepsilon))$.

Ridge-leverage scores
(Cohen, Musco, and Musco 2015)

A beautiful alternative: ridge-leverage scores. Recall:

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

$$m \times n \qquad m \times k \qquad k \times k \qquad k \times n$$

i-th row of U$_k$

j-th column of V$_k$^{T}

(Row) Leverage scores:

(Column) Leverage scores:

$$p_i = \frac{\|(U_k)_{i*}\|_2^2}{k}$$

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

The (row/column) leverage scores can now be used to sample rows/columns from A.

59

Ridge-leverage scores

(Cohen, Musco, and Musco 2015)

A beautiful alternative: ridge-leverage scores. Recall:

$$A \approx \left(\quad A_k \quad \right) = \left(\quad U_k \quad \right) \cdot \left(\quad \Sigma_k \quad \right) \cdot \left(\quad V_k^T \quad \right)$$

$$m \times n \qquad m \times k \qquad k \times k \qquad k \times n$$
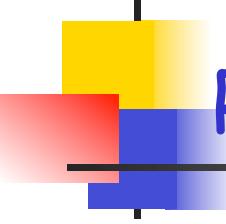
i-th row of U_k

(Row) Leverage scores:

$$p_i = \frac{\|(U_k)_{i*}\|_2^2}{k}$$

We will compare and contrast the leverage scores of **A** (with respect to the rank parameter **k**) vs. the norms of the rows of **A**.

This will help us introduce the ridge leverage scores for the rows of **A**.

60

Ridge-leverage scores
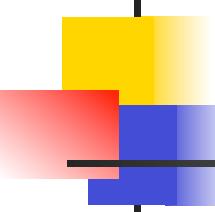
Let A have rank ρ and let U be the matrix of its left singular vectors.

Consider the matrix:

$$U \begin{pmatrix} \sigma_1^2 & & & & & & \\ & \ddots & & & & & \\ & & \sigma_k^2 & & & & \\ & & & \sigma_{k+1}^2 & & & \\ & & & & \ddots & & \\ & & & & & \sigma_\rho^2 \end{pmatrix} U^T$$

(with "top k" bracketing σ_1^2 through σ_k^2)

The above matrix is actually AA^T and its diagonal entries are equal to the (squares of the) Euclidean **norms of the rows of the matrix A.**

Ridge-leverage scores

Now consider the following matrix (k ones and ρ-k zeros in the diagonal):

$$U \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix} U^T$$

top k

We essentially **threshold** the singular values: the top **k** are mapped to ones and the remaining to zeros.

The diagonal entries of the above matrix are equal to the (squares of the) Euclidean **norms of the rows of the matrix** U_k, a.k.a. the row leverage scores of *A* with respect to the rank parameter **k**.

Ridge-leverage scores

Now consider the following "softer" thresholding scheme :

$$U \begin{pmatrix} \frac{\sigma_1^2}{\sigma_1^2+\lambda} & & & & & & \\ & \ddots & & & & & \\ & & \frac{\sigma_k^2}{\sigma_k^2+\lambda} & & & & \\ & & & \frac{\sigma_{k+1}^2}{\sigma_{k+1}^2+\lambda} & & & \\ & & & & \ddots & & \\ & & & & & \frac{\sigma_\rho^2}{\sigma_\rho^2+\lambda} \end{pmatrix} U^T$$

top k

If a singular value is much larger than λ then the respective fraction is close to one; otherwise, it is close to $\lambda\uparrow{-1}$.
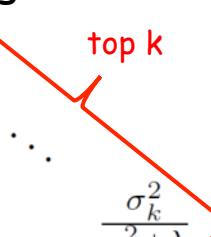
The diagonal entries of the above matrix are the **ridge leverage scores of A**.

Intuitively, it is a softer thresholding than the "standard" leverage scores.

Now consider the following "softer" thresholding scheme :
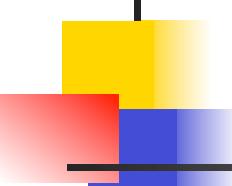
$$
U \begin{pmatrix}
\frac{\sigma_1^2}{\sigma_1^2+\lambda} & & & & & & \\
& \ddots & & & & & \\
& & \frac{\sigma_k^2}{\sigma_k^2+\lambda} & & & & \\
& & & \frac{\sigma_{k+1}^2}{\sigma_{k+1}^2+\lambda} & & & \\
& & & & \ddots & & \\
& & & & & \frac{\sigma_\rho^2}{\sigma_\rho^2+\lambda}
\end{pmatrix} U^T
$$

top k

The diagonal entries of the above matrix are the **ridge leverage scores of A**.
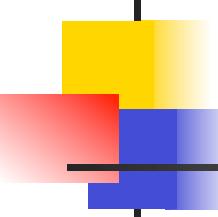
- The work by Cohen, Musco & Musco 2015 proves that the ridge leverage scores (for a specific choice of λ) have many of the good properties of leverage scores while being somewhat more intuitive and easier to compute and explain.

- Connections to ridge regression are also discussed and explored.

64

Roadmap of my lectures

➢ Approximating matrix multiplication (first lecture)

➢ **Leverage scores and their applications** (second and third lectures)

    1. Over- (or under-) constrained least squares problems

    2. Feature selection and the CX decomposition

➢ Solving systems of linear equations with Laplacian matrices (fourth lecture)

➢ Element-wise sampling (fourth lecture)

Problem definition and motivation

In many applications (e.g., statistical data analysis and scientific computation), one has <u>n observations</u> of the form:

$$y_i = y(t_i), i = 1, \ldots, n$$

Model y(t) (unknown) as a linear combination of <u>d basis functions</u>:
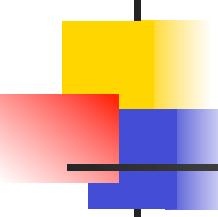
$$y(t) \approx x_1 \phi_1(t) + \cdots + x_d \phi_d(t)$$

A is an n-by-d "design matrix" (n » d):

$$A_{ij} = \phi_j(t_i)$$

In matrix-vector notation,

$$y \approx Ax$$
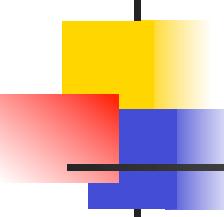
Least-norm approximation problems

Recall a linear measurement model:

$$y = Ax + \varepsilon \quad \begin{cases} y \text{ are the measurements} \\ x \text{ is the unknown} \\ \varepsilon \text{ is an error process} \end{cases}$$

In order to estimate x, solve:

$$\hat{x} = \arg\min \|y - Ax\|$$

Application: data analysis in science
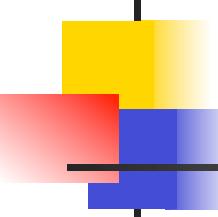
- First application: Astronomy

  Predicting the orbit of the asteroid Ceres (in 1801!).

  Gauss (1809) -- see also Legendre (1805) and Adrain (1808).

  First application of "least squares optimization" and runs in $O(nd^2)$ time!

- Data analysis: Fit parameters of a biological, chemical, economical, physical (astronomical), social, internet, etc. model to experimental data.

Norms of common interest

Let y = b and define the residual: $r = Ax - b \in R^n$

Least-squares approximation:
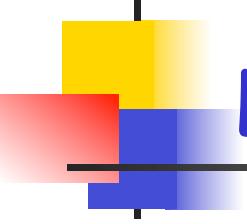
$$\text{minimize: } ||Ax - b||_2^2 = r_1^2 + r_2^2 + \cdots + r_n^2$$

Chebyshev or mini-max approximation:

$$\text{minimize: } ||Ax - b||_\infty = \max\{|r_1|, \ldots, |r_n|\}$$

Sum of absolute residuals approximation:

$$\text{minimize: } ||Ax - b||_1 = |r_1| + |r_2| + \cdots + |r_n|$$

Least-squares problems

$$\mathcal{Z}_2^2 = \min_{x \in \mathbb{R}^d} \|b - Ax\|_2^2 = \|b - Ax_{opt}\|_2^2 \quad \Longrightarrow \quad \begin{pmatrix} & A & \end{pmatrix} \begin{pmatrix} x_{opt} \end{pmatrix} \approx \begin{pmatrix} b \end{pmatrix}$$
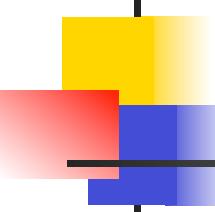
$$n \times d \ , \ n \gg d$$

We are interested in over-constrained least-squares problems, n ≫ d.

(Under-constrained problems: see Tygert 2009 and Drineas et al. (2012) JMLR)

Typically, there is no x_{opt} such that $Ax_{opt} = b$.

Want to find the "best" x_{opt} such that $Ax_{opt} \approx b$.

Exact solution to L_2 regression

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: $R^TRx = A^Tb$.

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

QR Decomposition:

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

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

$$\mathcal{Z}_2^2 = \|b\|_2^2 - \|AA^+b\|_2^2$$

$$x_{opt} = A^+b$$

Pseudoinverse of A

Singular Value Decomposition:

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

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

Complexity is $O(nd^2)$, but constant factors differ.

Algorithm: Sampling for L_2 regression

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

$$\begin{pmatrix} \\ A \\ \\ \end{pmatrix} \begin{pmatrix} x_{opt} \end{pmatrix} \approx \begin{pmatrix} \\ b \\ \\ \end{pmatrix}$$

$n \times d, \quad n >> d$

$n \times 1$

Algorithm

1. Compute the row-leverage scores of A (p_i, i=1…n)

2. In r i.i.d. trials pick r rows of A and the corresponding elements of b with respect to the p_i.

   (Rescale sampled rows of A and sampled elements of b by $(1/(rp_i)^{1/2}$.)

3. Solve the induced problem.

Algorithm: Sampling for least squares
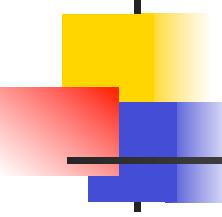
$$\tilde{\mathcal{Z}}_2^2 = \min_{x \in \mathbb{R}^d} \left\| \tilde{b} - \tilde{A}x \right\|_2^2 = \left\| \tilde{b} - \tilde{A}\tilde{x}_{opt} \right\|_2^2$$

$$\begin{pmatrix} \tilde{A} \\ r \times d \end{pmatrix} \begin{pmatrix} \tilde{x}_{opt} \end{pmatrix} \approx \begin{pmatrix} \tilde{b} \\ r \times 1 \end{pmatrix}$$

Algorithm

1. Compute the row-leverage scores of A (p_i, i=1...n)

2. In r i.i.d. trials pick r rows of A and the corresponding elements of b with respect to the p_i.

   (Rescale sampled rows of A and sampled elements of b by $(1/(rp_i)^{1/2}$.)

3. Solve the induced problem.

Theorem
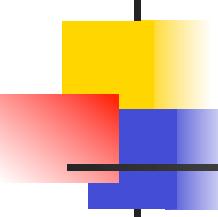
If the p_i are the row leverage scores of A, then, with probability at least 0.8,

$$\|b - Ax_{opt}\|_2 \leq \|b - A\tilde{x}_{opt}\|_2 \leq (1 + \epsilon) \|b - Ax_{opt}\|_2$$

The sampling complexity (the value of r) is

$$r = O\left(\frac{d}{\epsilon} + d \ln d\right)$$

Proof: a structural result

Consider the over-constrained least-squares problem:
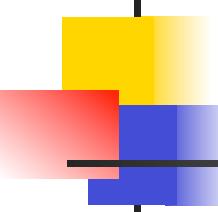
$$\mathcal{Z}_2^2 = \min_{x \in \mathbb{R}^d} \|b - Ax\|_2^2 = \|b - Ax_{opt}\|_2^2$$

and the "preconditioned" problem

$$\tilde{\mathcal{Z}}_2^2 = \min_{x \in \mathbb{R}^d} \|X(b - Ax)\|_2^2 = \|Xb - XA\tilde{x}_{opt}\|_2^2$$

Recall: A is n-by-d with n >> d; X is r-by-n with r << n.

➢ Think of XA as a "sketch" of A.

➢ Our approach (using the leverage scores) focused on sketches of A that consist of (rescaled) rows of A, thus X is a sampling matrix (matrix multiplication slides).

➢ More general matrices X will come later.

Proof: a structural result

$$\mathcal{Z}_2^2 = \min_{x \in \mathbb{R}^d} \|b - Ax\|_2^2 = \|b - Ax_{opt}\|_2^2 \qquad \tilde{\mathcal{Z}}_2^2 = \min_{x \in \mathbb{R}^d} \|X(b - Ax)\|_2^2 = \|Xb - XA\tilde{x}_{opt}\|_2^2$$
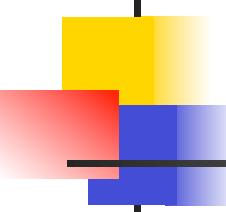
Let U_A be the n-by-d matrix of the left singular vectors of A (economy SVD).

If X satisfies (constants are somewhat arbitrary):

$$b^\perp = b - U_A U_A^T b \qquad \sigma_{min}^2(XU_A) \geq 1/\sqrt{2}$$

$$\left\| U_A^T X^T X b^\perp \right\|_2^2 \leq \epsilon \mathcal{Z}_2^2/2,$$

then,

$$\|A\tilde{x}_{opt} - b\|_2 \leq (1 + \epsilon)\mathcal{Z}_2$$

$$\|x_{opt} - \tilde{x}_{opt}\|_2 \leq \frac{1}{\sigma_{min}(A)}\sqrt{\epsilon}\mathcal{Z}_2$$

The second condition
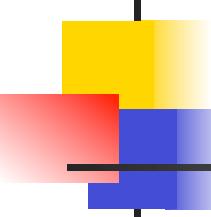
If X is a sampling and rescaling matrix and if leverage scores (row norms of U_A), were used as sampling probabilities, both bounds are satisfied.

Let's start with the second bound: simply apply the matrix multiplication result using as matrix A the matrix U_A and as matrix B the vector $b^{\perp}$:

$$\mathbf{E}\left(\left\|U_A^T X X^T b^{\perp} - U_A^T b^{\perp}\right\|_2^2\right) \leq \frac{1}{r}\|U_A\|_F^2 \left\|b^{\perp}\right\|_2^2 = \frac{d}{r}\mathcal{Z}_2^2$$

Using Markov's inequality and setting r = O(d/ε) satisfies the second condition with probability at least – say – 0.9.

Remark: one subtle point is that the sampling probabilities only depend on U_A (recall our remark here).

The first condition

The first bound can be proven as follows:

U_A is an orthogonal matrix:

$U_A^{\mathsf{T}} U_A = I_d$

$$U_A$$

$n \times d \quad , \quad n \gg d$
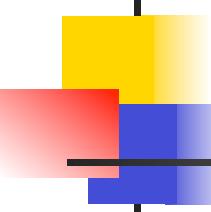
XU_A is a full-rank matrix!

$$XU_A$$

$r \times d$

$r = O\left(\dfrac{d}{\epsilon^2} \ln\left(\dfrac{d}{\epsilon^2 \sqrt{\delta}}\right)\right)$

Then, with probability at least 1-δ:

$$\left\|U_A^T U_A - U_A^T X^T X U_A\right\|_2 = \left\|I - U_A^T X^T X U_A\right\|_2 \leq \varepsilon$$

It follows that, for all i: $\sqrt{1 - \varepsilon} \leq \sigma_i\left(XU_A\right) \leq \sqrt{1 + \varepsilon}$

The first condition
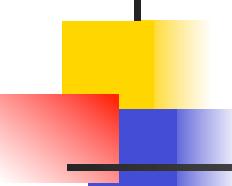
Recall: with probability at least 1-δ:

$$\left\|U_A^T U_A - U_A^T X^T X U_A\right\|_2 = \left\|I - U_A^T X^T X U_A\right\|_2 \leq \varepsilon$$

It follows that, for all i: $\sqrt{1-\varepsilon} \leq \sigma_i(XU_A) \leq \sqrt{1+\varepsilon}$
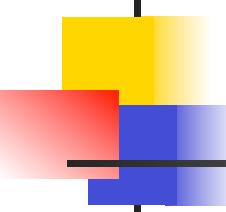
- ➢ Setting ε = 1/2 and 1 - δ = 0.9 implies the second condition for XU_A.
- ➢ Notice that the sampling complexity is O(d ln d).
- ➢ Applying the union bound to bound the failure probability for both conditions concludes the proof that both conditions hold with constant probability.

Roadmap of my lectures

➢ Approximating matrix multiplication (first lecture)

➢ **Leverage scores and their applications** (second and third lectures)

    1. Over- (or under-) constrained least squares problems

    2. Feature selection and the CX decomposition

➢ Solving systems of linear equations with Laplacian matrices (fourth lecture)

➢ Element-wise sampling (fourth lecture)

An example in 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

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

HGDP data

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

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*

82

HGDP data

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

HapMap Phase 3 data

- 1,207 samples
- 11 populations

HapMap Phase 3

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*

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

83

CEU

TSI

JPT, CHB, & CHD

MEX

GIH

ASW, MKK, LWK, & YRI

HapMap Phase 3

The Human Genome Diversity Panel (HGDP)

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.
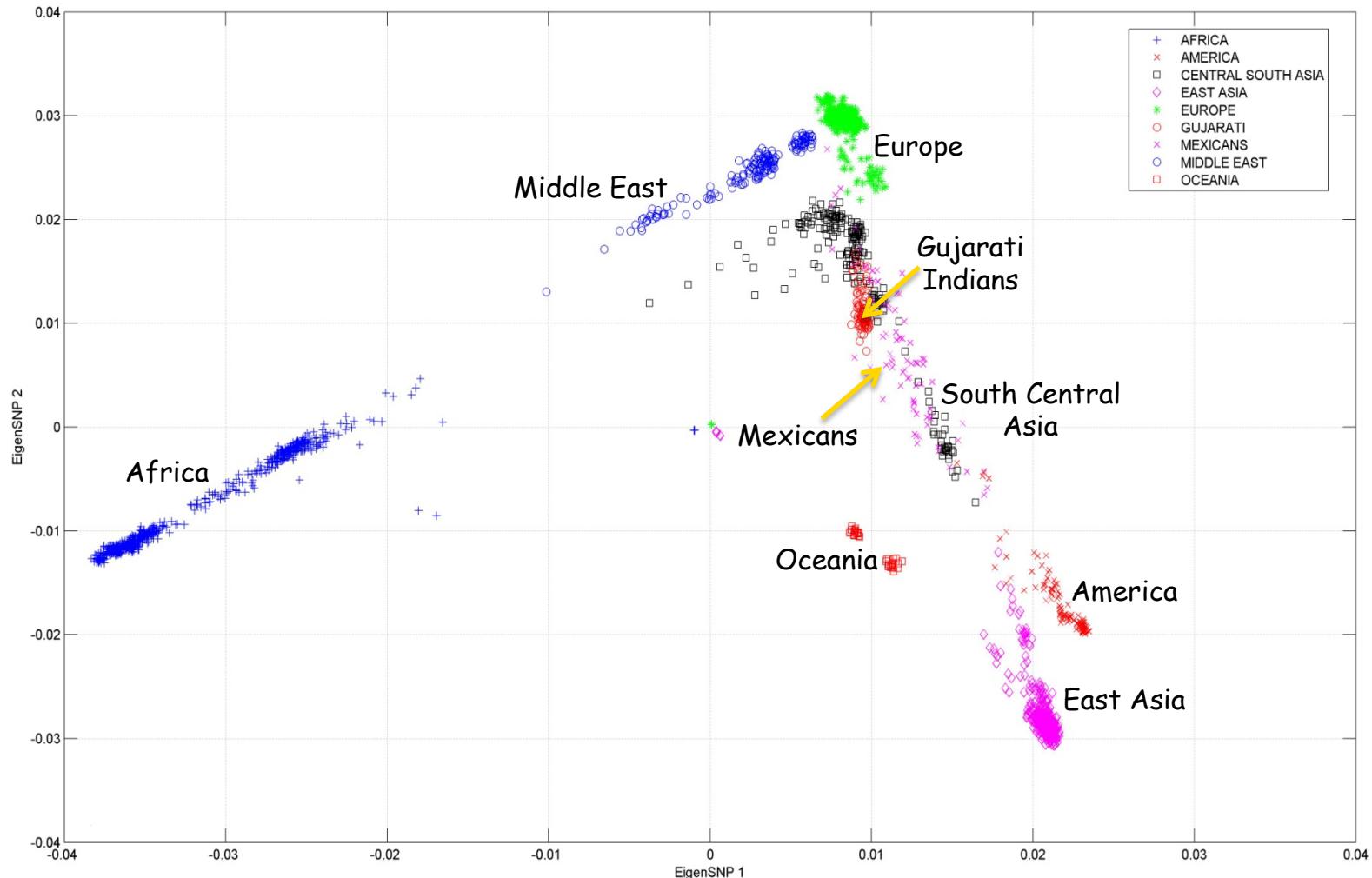
Matrix dimensions:

2,240 subjects (rows)
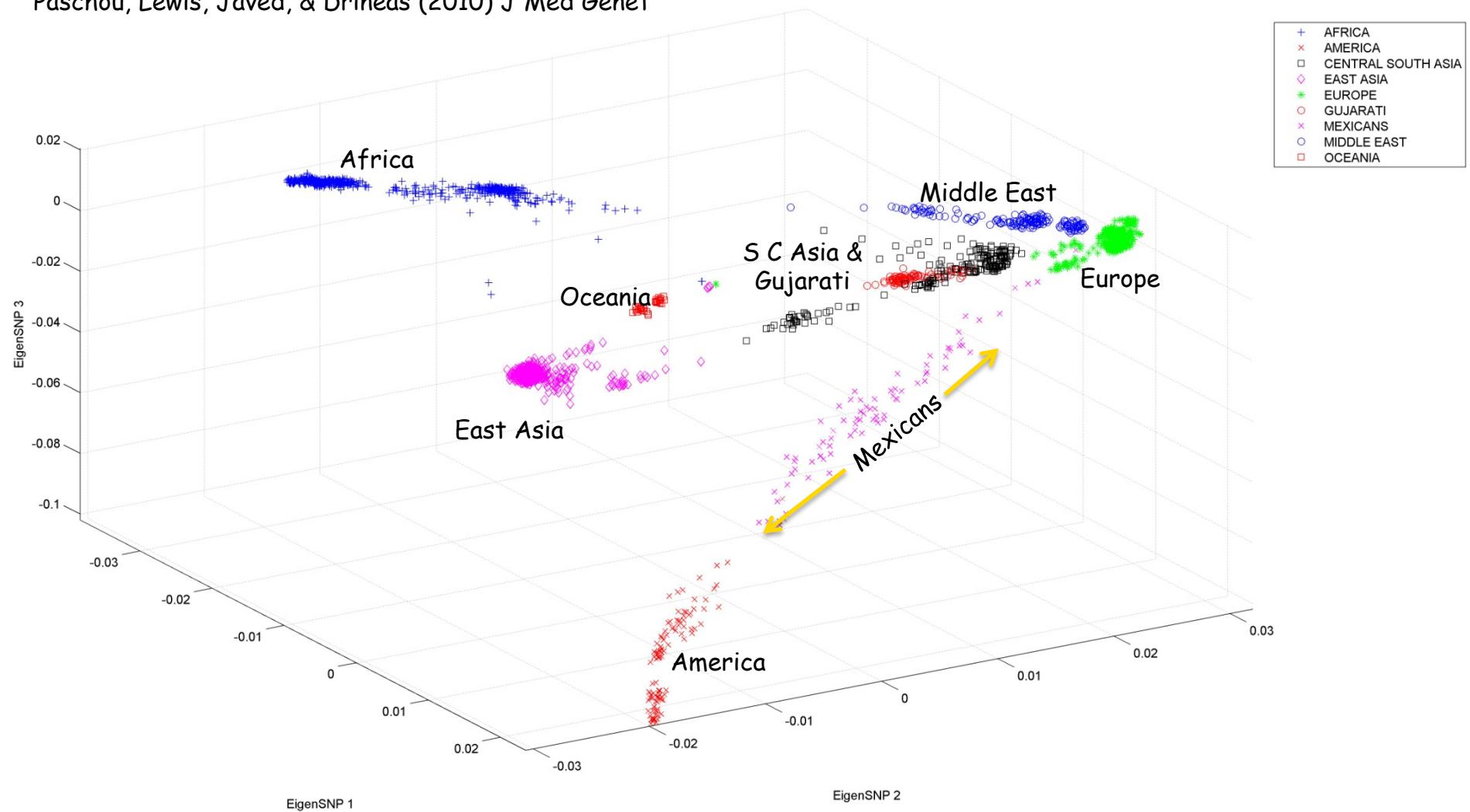
447,143 SNPs (columns)

Dense matrix:

over one billion entries

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*

84

- <u>Top two Principal Components</u> (PCs or eigenSNPs)

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

- Very good correlation between geography and the top two eigenSNPs.

- Mexican population seems out of place: we move to the top three PCs.

85
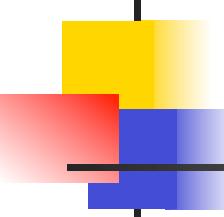
Not altogether satisfactory: the principal components are linear combinations of all SNPs, and – of course – can not be assayed!

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

Formally: spanning the same subspace.

86

- PCA plots of genetic data from multiple populations around the Mediterranean Sea indicate that the Mediterranean acted as a **"barrier"** during the colonization of Europe from our species.

- Using PCA (and many other analyses) we proposed what is a called a **maritime route** for the colonization of Europe.

- Interpreting the singular vectors is, again, tricky; we are now working on identifying actual SNPs that capture the information in the singular vectors.

SVD decomposes a matrix as...

$$\begin{pmatrix} & m \times n & \\ & A & \\ & & \end{pmatrix} \approx \begin{pmatrix} & m \times k & \\ & U_k & \\ & & \end{pmatrix} \begin{pmatrix} & k \times n & \\ & X & \\ & & \end{pmatrix}$$

The SVD has strong optimality properties.

Top **k** left singular vectors

➢ It is easy to see that $X = U_k^\top A = \Sigma_k V_k^\top$.

➢ SVD has strong optimality properties.

➢ The columns of U_k are linear combinations of up to all columns of A.

The CX decomposition

$$
\underbrace{\left(\begin{array}{c} \\ \\ A \\ \\ \\ \end{array} \right)}_{m \times n} \approx \underbrace{\left(\begin{array}{c} \\ \\ C \\ \\ \\ \end{array} \right)}_{m \times c} \underbrace{\left(\begin{array}{c} \\ X \\ \\ \end{array} \right)}_{c \times n}
$$

Carefully chosen X

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

c columns of A, with c being as close to k as possible

<u>Why?</u>

If A is a data matrix with rows corresponding to objects and columns to features, then selecting representative columns is equivalent to selecting representative features to capture the same structure as the top eigenvectors.

We want c as close to k as possible!

89

CX decomposition

$$\begin{pmatrix} & m \times n & \\ & A & \\ & & \end{pmatrix} \approx \begin{pmatrix} & m \times c & \\ & C & \\ & & \end{pmatrix} \begin{pmatrix} & c \times n & \\ & X & \\ & & \end{pmatrix}$$
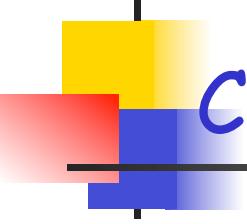
c columns of A, with c being
as close to k as possible

Easy to prove that optimal X = C⁺A.

(with respect to unitarily invariant norms; C⁺ is the Moore-Penrose pseudoinverse of C)

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

Also known as: the Column Subset Selection Problem (CSSP).

Column Subset Selection Problem (CSSP)
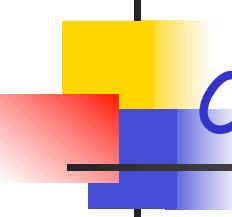
$$\min \left\| A - CC^+A \right\|_F = \min \left\| A - P_C A \right\|_F$$

Given an m-by-n matrix A, find k columns of A forming an m-by-k matrix C that minimizes the above error over all $O(n^k)$ choices for C.

C^+: pseudoinverse of C.

(just in case: if $C = U\Sigma V^\mathsf{T}$, then $C^+ = V\Sigma^{-1}U^\mathsf{T}$)

$P_C = CC^+$ is the projector matrix on the subspace spanned by the columns of C.
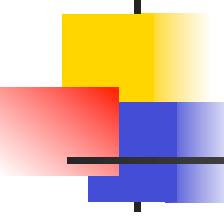
Column Subset Selection Problem (CSSP)

$$\min \left\| A - CC^{+}A \right\|_{F} = \min \left\| A - P_{C}A \right\|_{F}$$

Given an m-by-n matrix A, find k columns of A forming an m-by-k matrix C that minimizes the above error over all $O(n^k)$ choices for C.

Complexity of the problem? $O(n^k mn)$ trivially works; at least one variant is NP-hard if k grows as a function of n.

(NP-hardness in Civril & Magdon-Ismail '07)

Spectral norm
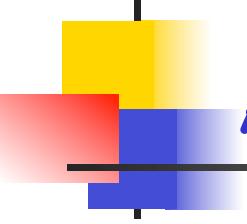
Given an m-by-n matrix A, find k columns of A forming an m-by-k matrix C such that

$$\|A - P_C A\|_2$$

is minimized over all $O(n^k)$ possible choices for C.

<u>Remarks:</u>

1. $P_C A$ is the projection of A on the subspace spanned by the columns of C.
2. More generally, any Schatten p-norm could be used.

A lower bound for the CSSP problem
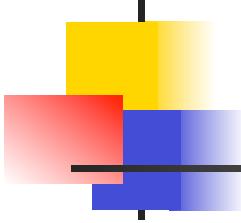
For any **m-by-k** matrix **C** consisting of at most **k** columns of **A**

$$\|A - P_C A\|_2 \quad \geq \quad \|A - P_{U_k} A\|_2$$

A_k

Remarks:
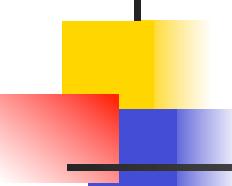
1. This is also true if we replace the spectral norm by the Frobenius norm or any unitarily invariant norm.

2. This is a – potentially – weak lower bound.

TA session III

TA session III will focus on leverage scores and randomized solvers for least-squares problem and is posted at
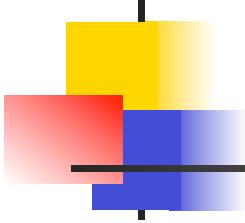
http://www.drineas.org/RandNLA-PCMI-2016/

Roadmap of my lectures

➢ Approximating matrix multiplication (first lecture)

➢ **Leverage scores and their applications** (second and third lectures)

　　1. Over- (or under-) constrained least squares problems

　　2. Feature selection and the CX decomposition (cont'd)

➢ Solving systems of linear equations with Laplacian matrices (fourth lecture)

➢ Element-wise sampling (fourth lecture)

Back to the CX decomposition

We would like to get theorems of the following form:
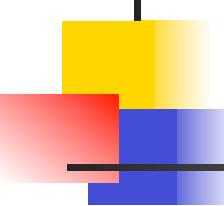
low-degree polynomial
in m, n, and k

Given an *m-by-n* matrix A, there exists an ***efficient*** algorithm that picks a ***small*** number of columns of A such that with ***reasonable*** probability:

Close to k/ε

constant, high, almost
surely, etc.
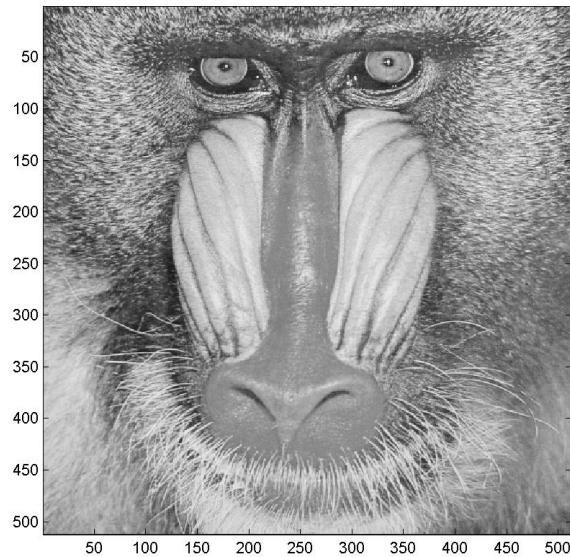
$$\|A - CX\|_F = \left\|A - CC^\dagger A\right\|_F \le (1 + \varepsilon)\, \|A - A_k\|_F$$

Let's start with a simpler, weaker result, connecting the *spectral* norm of *A-CX* to matrix multiplication.
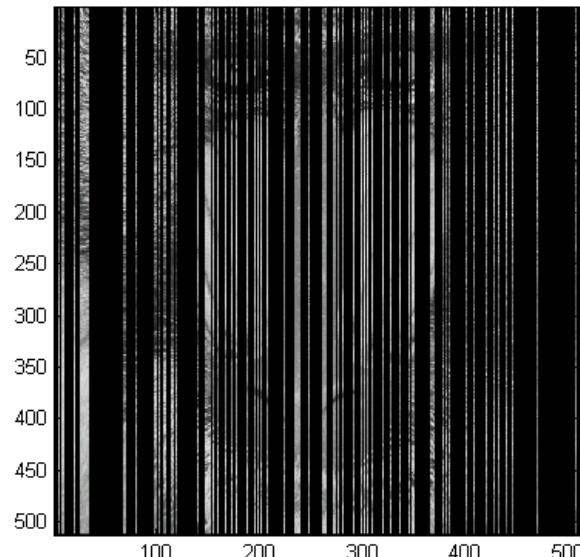
(A similar result can be derived for the Frobenius norm, but takes more effort to prove; see Drineas, Kannan, & Mahoney (2006) SICOMP)

Approximating singular vectors
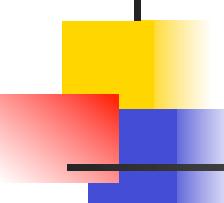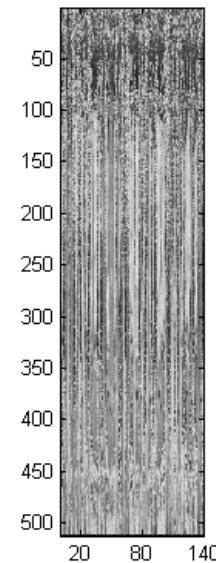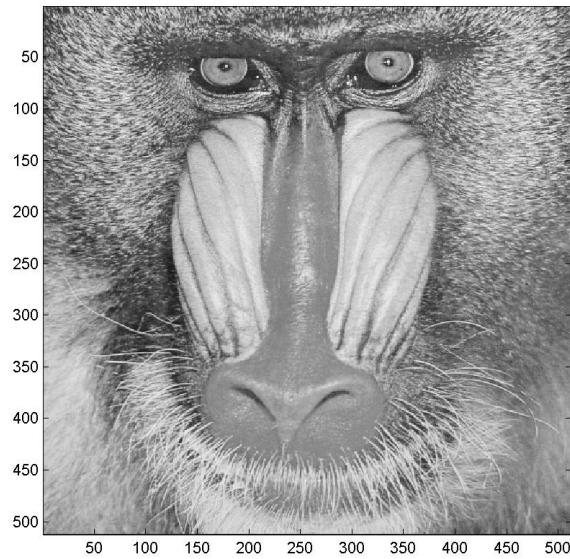
Original matrix → Sampling (c = 140 columns)

1. Sample c (=140) columns of the original matrix A and rescale them appropriately to form a 512-by-c matrix C.

2. Show that $A-CX$ is "small".

(C^+ is the pseudoinverse of C and $X = C^+A$)

98

Approximating singular vectors

Original matrix

Sampling (c = 140 columns)

1. Sample c (=140) columns of the original matrix A and rescale them appropriately to form a 512-by-c matrix C.

2. Show that A-CX is "small".

(C^+ is the pseudoinverse of C and X= C^+A)

Approximating singular vectors (cont'd)

A

CX

The fact that $AA^{\mathsf{T}} - CC^{\mathsf{T}}$ is small will imply that A-CX is small as well.
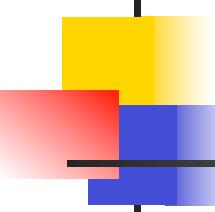
Proof (spectral norm)

Using the triangle inequality and properties of norms,

$$
\begin{aligned}
\left\| A - CC^{\dagger}A \right\|_2^2 &= \left\| \left(I - CC^{\dagger} \right) A \right\|_2^2 \\
&= \left\| \left(I - CC^{\dagger} \right) AA^T \left(I - CC^{\dagger} \right)^T \right\|_2 \\
&= \left\| \left(I - CC^{\dagger} \right) \left(AA^T - CC^T \right) \left(I - CC^{\dagger} \right)^T \right\|_2 \\
&\leq \left\| AA^T - CC^T \right\|_2
\end{aligned}
$$

projector matrices

We used the fact that $(I-CC^+)CC^T$ is equal to zero.

Proof (spectral norm), cont'd
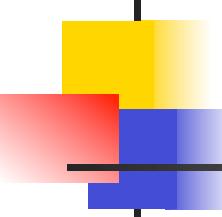
Assume that our sampling is done in c i.i.d. trials and the sampling probabilities are:

$$\Pr\left(j_t = i\right) = \frac{\|A_{*i}\|_2^2}{\|A\|_F^2}$$

We can use our matrix multiplication result:

(For simplicity, we will just upper bound the spectral norm by the Frobenius norm.)

$$\mathbb{E}\left[\left\|A - CC^\dagger A\right\|_2^2\right] \leq \mathbb{E}\left[\left\|AA^T - CC^T\right\|_2\right]$$

$$\leq \frac{1}{\sqrt{c}}\|A\|_F^2$$

Is this a good bound?

$$\mathbb{E}\left[\left\|A - CC^{\dagger}A\right\|_2^2\right] \leq \mathbb{E}\left[\left\|AA^T - CC^T\right\|_2\right]$$

$$\leq \frac{1}{\sqrt{c}}\|A\|_F^2$$

Problem 1: If $c = n$ we do not get zero error.

That's because of sampling with replacement.
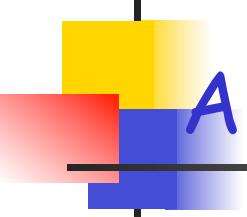
(We know how to analyze uniform sampling without replacement, but we have no bounds on non-uniform sampling without replacement.)

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

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.

Such deterministic bounds exist when $c = k$ and depend on $(k\,(n - k\,))^{1/2}\,||A\text{-}A_k||_2$

A relative error algorithm
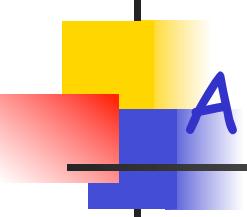
Input: m-by-n matrix A, target rank k

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

Output: C, the matrix consisting of the selected columns

Sampling algorithm

• Let p_j be the column leverage scores of A, for j=1…n.

• 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.

                                                  (c is a function of ε and k)

• Let C be the matrix consisting of the chosen columns.

A relative error algorithm

Input: m-by-n matrix A, target rank k

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

Output: C, the matrix consisting of the selected columns

Sampling algorithm

• Let p_j be the column leverage scores of A, for j=1…n.

• 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.

<div align="right">(c is a function of ε and k)</div>

• Let C be the matrix consisting of the chosen columns.

Note: there is no rescaling of the columns of C in this algorithm; however, since our error matrix is $A-CX = A-CC^+A$, rescaling the columns of C (as we did in our matrix multiplication algorithms), does not change $A-CX = A-CC^+A$.

Remember the leverage scores?

Let **A** be an **m**-by-**n** matrix **A** and let A_k be its best rank-**k** approximation (as computed by the SVD) :

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

$$m \times n \qquad m \times k \qquad k \times k \qquad k \times n$$

i-th row of U_k j-th column of $V_k{}^T$
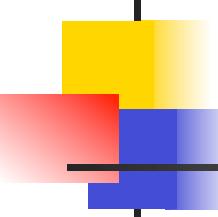
(Row) Leverage scores: (Column) Leverage scores:

The (row/column) leverage scores can now be used to sample rows/columns from **A**.

$$p_i = \frac{\|(U_k)_{i*}\|_2^2}{k}$$

$$p_j = \frac{\|(V_k)_{j*}\|_2^2}{k}$$

Relative-error Frobenius norm bounds

Given an m-by-n matrix A, let C be formed as described in the previous algorithm. Then, with probability at least 0.9,

$$\|A - CX\|_F = \left\|A - CC^\dagger A\right\|_F \leq (1 + \varepsilon) \|A - A_k\|_F$$

The sampling complexity (the value of c) is

$$c = O\left(\frac{k}{\epsilon^2} \ln\left(\frac{k}{\epsilon^2}\right)\right)$$
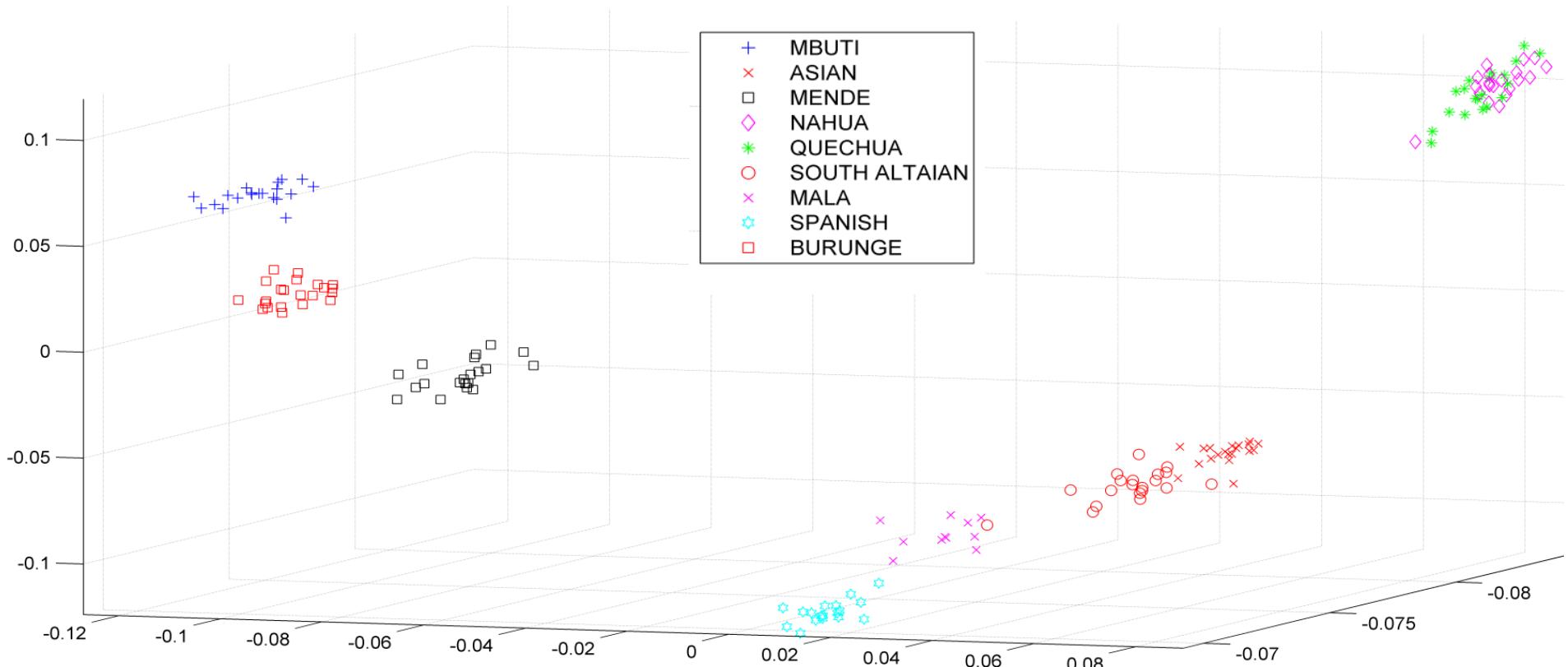
The running time of the algorithm is dominated by the computation of the (column) leverage scores.

Worldwide data

South Altaians

Spanish

Japanese

Mende   **Mbuti**

Nahua

Mala

Burunge

Quechua

● Africa   ● Europe   ● E Asia   ● America

274 individuals, 9 populations, ~10,000 SNPs
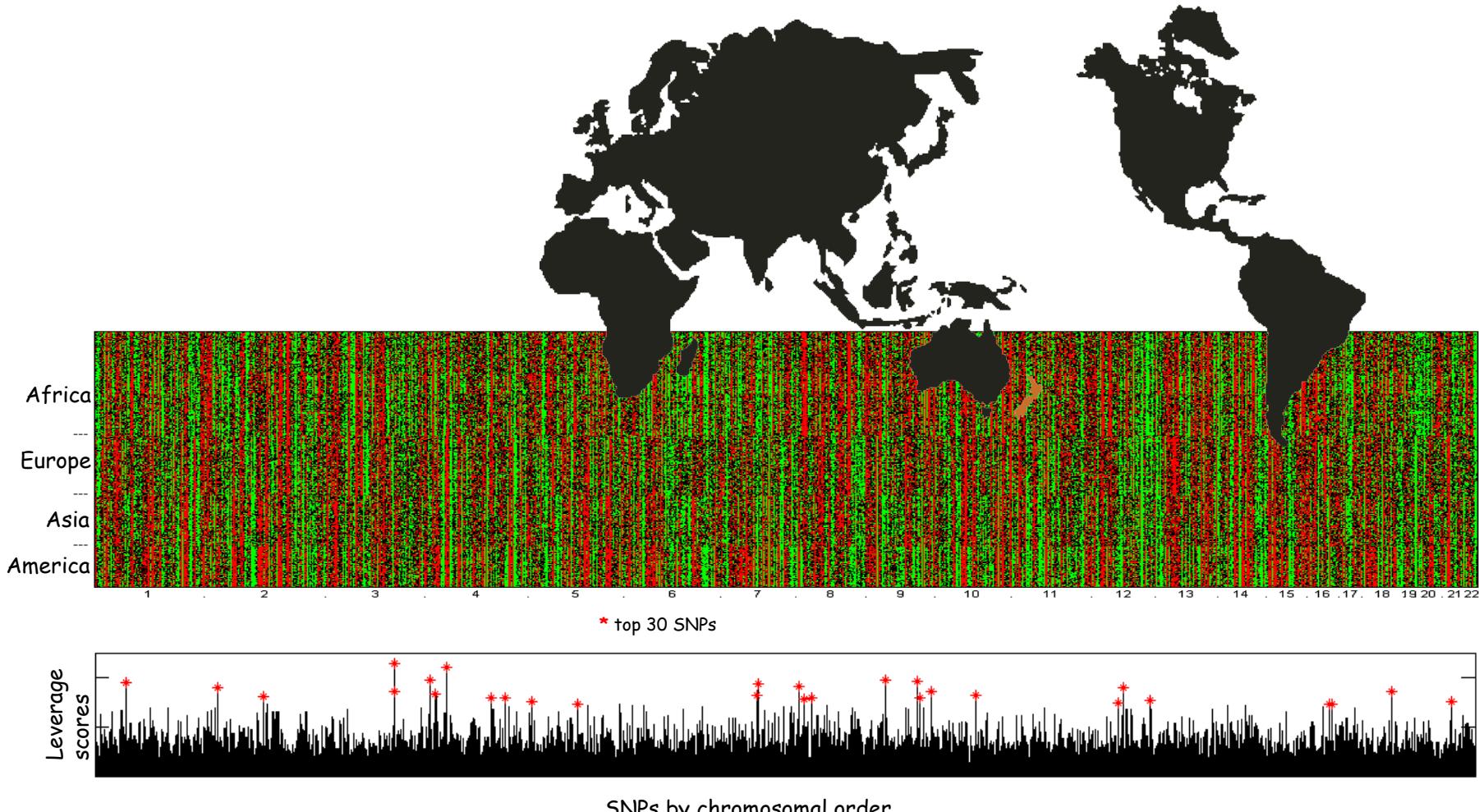
Shriver et al. (2005) Hum Genom

> PCA projection on the top three left singular vectors.

> Populations are clearly separated, but **recall that:**

>> The principal components are **linear combinations of all SNPs.**

>> **Hard to interpret or genotype.**

> Can we find **<u>actual SNPs</u>** that capture the information in the left singular vectors?

BACK TO POPULATION GENETICS DATA
Selecting PCA SNPs for individual assignment to four continents
(Africa, Europe, Asia, America)

* top 30 SNPs

SNPs by chromosomal order
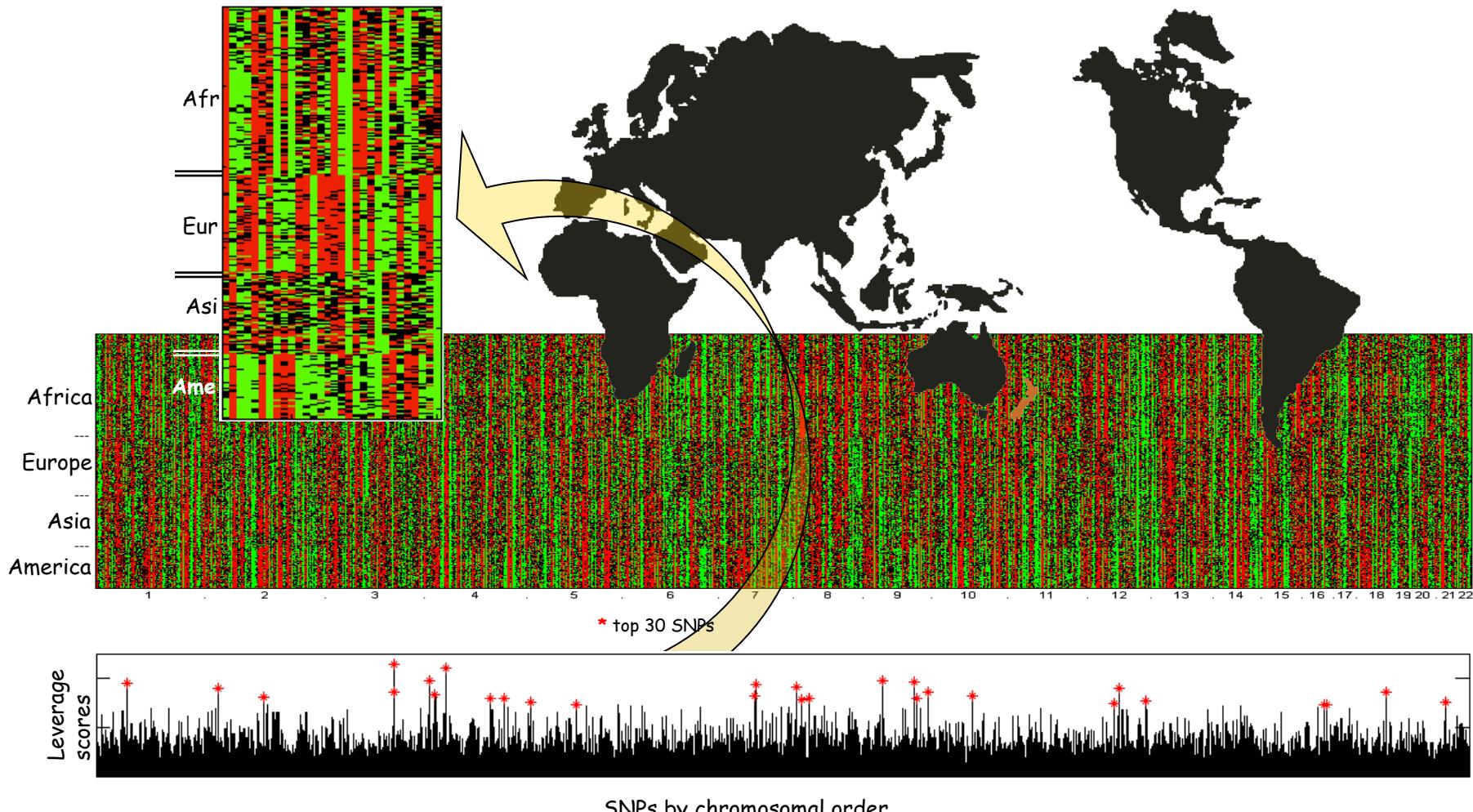
Paschou et al (2007; 2008) PLoS Genetics

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)

* top 30 SNPs

Leverage scores

SNPs by chromosomal order
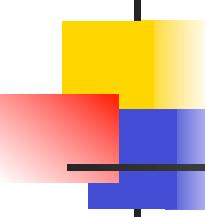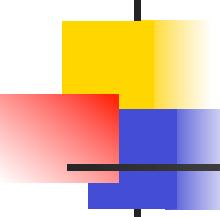
Paschou et al (2007; 2008) PLoS Genetics
Paschou et al (2010) J Med Genet
Drineas et al (2010) PLoS One

Leverage Scores: concerns
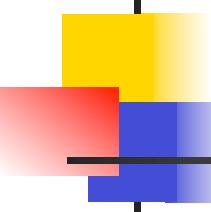
➢ Highly correlated SNPs (features) get similar scores.

  ➢ As a result, correlated features could be selected multiple times, thus introducing redundancy in the selected features.

  ➢ How do we remove this redundancy?

➢ Stability of the selected features

  ➢ How sensitive are the leverage scores on perturbations of the input data?

  ➢ What happens if we randomly remove a few samples (objects)?

  ➢ Some empirical evidence.

  ➢ Some recent theory by Holodnak, Wentworth, & Ipsen (2015) SIMAX.

➢ Some SNPs (features) are only relevant for a subset of the samples (objects).

  ➢ It would be nice to have leverage scores at a finer resolution than columns/rows.

  ➢ A notion of element-wise leverage scores?

Is running time an issue?

- **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)

    - To compare mtDNA derived from 37 ancient Minoan bones to 120 extant and ancient populations we ran (multiple) SVDs on (approx.) 14,000-by-14,000 matrices.

    (Hughey, Paschou, Drineas, et al. (2013) Nat Comm; used the random projection ideas that will come later)

113

Is running time an issue?

- **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)

  - To compare mtDNA derived from 37 ancient Minoan bones to 120 extant and ancient populations we ran (multiple) SVDs on (approx.) 14,000-by-14,000 matrices.

  (Hughey, Paschou, Drineas, et al. (2013) Nat Comm; used the random projection ideas that will come later)

- **Running time is <u>always</u> a concern, <u>but</u>**

- **machine-precision accuracy is <u>not</u> necessary!**

  - Data are noisy.

  - Approximate singular vectors work well in our setting.
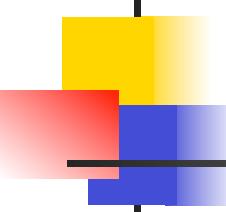
Towards a relative error bound...

Structural result (deterministic):

$$\|A - CX\|_F = \left\|A - CC^\dagger A\right\|_F \leq \|A - A_k\|_F + \left\|(A - A_k) S \left(V_k^T S\right)^\dagger\right\|_F$$

This holds for any n-by-c matrix S such that C = AS **as long as the k-by-c matrix V_k^T S has full rank (equal to k).**
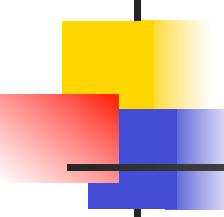
Towards a relative error bound...

Structural result (deterministic):

$$\|A - CX\|_F = \left\|A - CC^{\dagger}A\right\|_F \leq \|A - A_k\|_F + \left\|(A - A_k)\, S\, (V_k^T S)^{\dagger}\right\|_F$$

This holds for any n-by-c matrix S such that C = AS **as long as the k-by-c matrix V_k^TS has full rank (equal to k).**

- The proof of the structural result critically uses the fact that with X = C$^+$A is the *argmin* for any unitarily invariant norm of the error A-CX.

- Variants of this structural result have appeared in various papers.

(e.g., (i) Drineas, Mahoney, Muthukrishnan (2008) SIMAX, (ii) Boutsidis, Drineas, Mahoney SODA 2011, (iii) Halko, Martinsson, Tropp (2011) SIREV, (iv) Boutsidis, Drineas, Magdon-Ismail FOCS 2011, etc.)

The rank of $V_k^\top S$
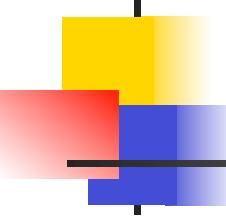
Structural result (deterministic):

$$\|A - CX\|_F = \left\|A - CC^\dagger A\right\|_F \leq \|A - A_k\|_F + \left\|(A - A_k)\, S\, (V_k^T S)^\dagger\right\|_F$$

This holds for any n-by-c matrix S such that C = AS **as long as the k-by-c matrix $V_k^\top S$ has full rank (equal to k).**

Let S be a sampling and rescaling matrix, where the sampling probabilities are the leverage scores: our matrix multiplication results (and the fact that the square of the Frobenius norm of V_k if equal to k) guarantee that, for our choice of c (with constant probability):

$$\left\|V_k^T V_k - V_k^T S S^T V_k\right\|_2 = \left\|I_k - V_k^T S S^T V_k\right\|_2 \leq \varepsilon$$

117

The rank of $V_k{}^\mathsf{T}S$ (cont'd)
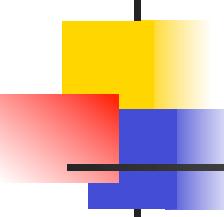
From matrix perturbation theory, if

$$\left\|V_k^T V_k - V_k^T S S^T V_k\right\|_2 = \left\|I_k - V_k^T S S^T V_k\right\|_2 \leq \varepsilon$$

it follows that all singular values (σ_i) of $V_k{}^\mathsf{T}S$ satisfy:

$$\sqrt{1-\varepsilon} \leq \sigma_i\left(V_k^T S\right) \leq \sqrt{1+\varepsilon}$$

By choosing ε small enough, we can guarantee that $V_k{}^\mathsf{T}S$ has full rank (with constant probability).

Bounding the second term
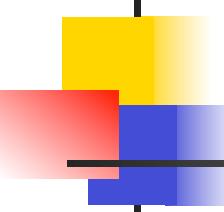
Structural result (deterministic):

$$\|A - CX\|_F = \left\|A - CC^\dagger A\right\|_F \leq \|A - A_k\|_F + \left\|(A - A_k)\, S\, (V_k^T S)^\dagger\right\|_F$$

This holds for any n-by-c matrix S such that C = AS **as long as the k-by-c matrix V_k^TS has full rank (equal to k).**

Using strong submultiplicativity for the second term:

$$\left\|(A - A_k)\, S\, (V_k^T S)^\dagger\right\|_F \leq \|(A - A_k)\, S\|_F \left\|(V_k^T S)^\dagger\right\|_2$$

$$= \sigma_{\min}^{-1}(V_k^T S)\, \|(A - A_k)\, S\|_F$$

Bounding the second term (cont'd)
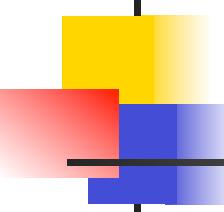
To conclude:

$$\left\| (A - A_k) \, S \, (V_k^T S)^\dagger \right\|_F \;\leq\; \|(A - A_k) S\|_F \left\| (V_k^T S)^\dagger \right\|_2$$

$$= \; \sigma_{\min}^{-1} (V_k^T S) \, \|(A - A_k) S\|_F$$

(i) We already have a bound for all singular values of $V_k^T S$ (go back two slides).

(ii) It is easy to prove that, using our sampling and rescaling,

$$\mathbb{E}\left[\|(A - A_k) S\|_F^2 \right] = \|A - A_k\|_F^2$$

Collecting, we get a (2+ε) constant-factor approximation.

Bounding the second term (cont'd)
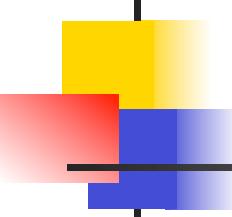
To conclude:

$$\left\|(A - A_k)\, S\, (V_k^T S)^\dagger \right\|_F \leq \|(A - A_k)\, S\|_F \left\|(V_k^T S)^\dagger\right\|_2$$

$$= \sigma_{\min}^{-1}(V_k^T S)\, \|(A - A_k)\, S\|_F$$

(i) We already have a bound for all singular values of $V_k^T S$ (go back two slides).

(ii) It is easy to prove that, using our sampling and rescaling,

$$\mathbb{E}\left[\|(A - A_k)\, S\|_F^2\right] = \|A - A_k\|_F^2$$

Collecting, we get a (2+ε) constant-factor approximation.

A more careful (albeit, longer) analysis can improve the result to a (1+ε) relative-error approximation.

Using a dense matrix S
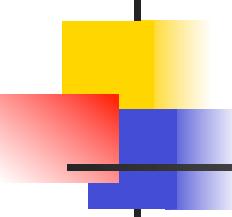
Our proof would also work if instead of the sampling matrix S, we used, for example, the dense random sign matrix S:

$$S_{ij} = \begin{cases} +1/\sqrt{c} & \text{,w.p. } 1/2 \\ -1/\sqrt{c} & \text{,w.p. } 1/2 \end{cases}$$

The intuition is clear: the most critical part of the proof is based on approximate matrix multiplication to bound the singular values of $V_k^\mathsf{T}S$.

This also works when S is a dense random projection matrix.

Using a dense matrix S
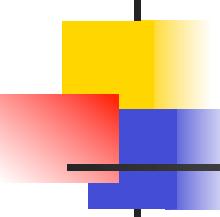
Notes:

Negative: C=AS does not consist of columns of A (interpretability is lost).

Positive: It can be shown that the span of C=AS contains "relative-error" approximations to the top k left singular vectors of A, which can be computed in $O(nc^2)$ time.

Thus, we can compute approximations to the top k left singular vectors of A in $O(mnc + nc^2)$ time, **already faster than** the naïve $O(\min\{mn^2, m^2n\})$ time of the **full SVD**.

Using a dense matrix S
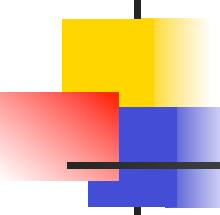
Notes:

Negative: $C=AS$ does not consist of columns of A (interpretability is lost).

Positive: It can be shown that the span of $C=AS$ contains "relative-error" approximations to the top k left singular vectors of A, which can be computed in $O(nc^2)$ time.

Thus, we can compute approximations to the top k left singular vectors of A in $O(mnc +nc^2)$ time, **already faster than** the naïve $O(\min\{mn^2, m^2n\})$ time of the **full SVD**.

Even better: Using very fast random projections (the Fast Hadamard Transform, or the Clarkson-Woodruff input sparsity time random projection), we can reduce the (first term of the) running time further.

Implementations are simple and work very well in practice!

Better approaches for PCA

To get highly accurate approximations for singular vectors, use iterative methods.
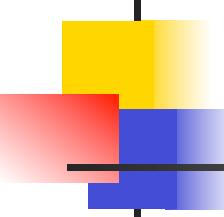
1. Block subspace iteration

Given an m-by-n matrix A and a positive integer q, compute

$$K = \left(AA^T\right)^q AX$$

where X is an n-by-p (with p ≈ k) random matrix, e.g., a random Gaussian matrix.

Compute the best rank-k approximation to A within the subspace spanned by the columns of K (much easier to do than it sounds…): denote it by $A \downarrow k$.

Better approaches for PCA

To get highly accurate approximations for singular vectors, use iterative methods.

1. Block subspace iteration
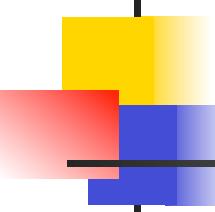
Given an m-by-n matrix A and a positive integer q, compute

$$K = \left(AA^T\right)^q AX$$

where X is an n-by-p (with p ≈ k) random matrix, e.g., a random Gaussian matrix.

Compute the best rank-k approximation to A within the subspace spanned by the columns of K (much easier to do than it sounds…): denote it by .

- Strong bounds can be proven for the Frobenius and spectral norms of the matrix .
- We will skip details for block subspace iteration and move on to Block Lanczos methods.

Better approaches for PCA

2. Block Lanczos methods

Given an m-by-n matrix A (of rank ρ) and a positive integer q, compute
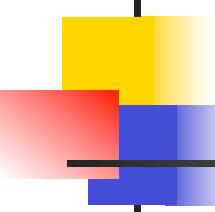
$$K = \left[AX, \ \left(AA^T\right) AX, \ \left(AA^T\right)^2 AX, \ \dots , \left(AA^T\right)^q AX \right]$$

where X is an n-by-p (with p ≈ k) random matrix, e.g., a random Gaussian matrix.

Compute the best rank-k approximation to A within the subspace spanned by the columns of K (much easier to do than it sounds…): denote it by $A \downarrow k$.

- Assume a gap g(>0) between the k and (k+1)-st singular values (can be relaxed):

$$\sigma_k \geq (1 + g)\, \sigma_{k+1} > 0$$

Better approaches for PCA

2. Block Lanczos methods

Given an m-by-n matrix A (of rank ρ) and a positive integer q, compute

$$K = \left[AX, \ (AA^T) \, AX, \ (AA^T)^2 \, AX, \ \dots \ , (AA^T)^q \, AX \right]$$

where X is an n-by-p (with p ≈ k) random matrix, e.g., a random Gaussian matrix.

Compute the best rank-k approximation to A within the subspace spanned by the columns of K (much easier to do than it sounds…): denote it by $A \downarrow k$.

■ Assume a gap g(>0) between the k and (k+1)-st singular values (can be relaxed):

$$\sigma_k \geq (1 + g) \, \sigma_{k+1} > 0$$

Bottom ρ-k singular vectors of A

■ Also assume (γ₁ and γ₂ are constants):

$$\sigma_{\min}^2 \left(V_k^T X \right) \geq \gamma_1^2 \qquad \text{and} \qquad \left\| V_{k,\perp}^T X \right\|_F^2 \leq \gamma_2^2 (\rho - k)$$

128

Better approaches for PCA
(Musco & Musco NIPS 2015, Drineas, Ipsen, Iyer, and Magdon-Ismail 2016)

2. Block Lanczos methods
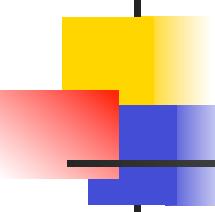
Given an m-by-n matrix A (of rank ρ) and a positive integer q, compute

$$q = O\left(\frac{\log(\rho/\epsilon)}{\sqrt{g}}\right)$$

$$K = \left[AX, \ (AA^T)AX, \ (AA^T)^2 AX, \ \dots \ , (AA^T)^q AX\right]$$

where X is an n-by-p (with p ≈ k) random matrix, e.g., a random Gaussian matrix.

Compute the best rank-k approximation to A within the subspace spanned by the columns of K (much easier to do than it sounds…): denote it by $A \downarrow k$. Then,

$$\left\|A - \tilde{A}_k\right\|_F \leq \|A - A_k\|_F + \epsilon\sigma_{k+1}$$

$$\left\|A - \tilde{A}_k\right\|_2 \leq \|A - A_k\|_2 + \epsilon\sigma_{k+1}$$

Selecting fewer columns

Problem

How many columns do we need to include in the matrix C in order to get relative-error approximations ?
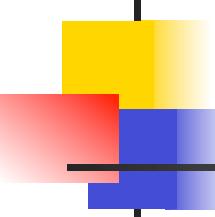
Recall: with $O((k/\varepsilon^2)\log(k/\varepsilon^2))$ columns, we get (subject to a failure probability)

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

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

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

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 $O(k \log k)$?
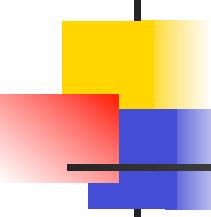
Answer:

A relative-error bound is possible by selecting **$c=2k/\varepsilon+o(1)$** 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…**

Towards such a result
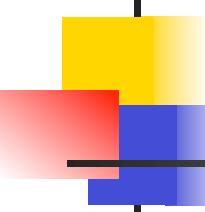
First, let the top-k right singular vectors of A be V_k.

A structural result (deterministic):

$$\|A - CX\|_F = \left\|A - CC^{\dagger}A\right\|_F \leq \|A - A_k\|_F + \left\|(A - A_k)S\left(V_k^T S\right)^{\dagger}\right\|_F$$

Again, this holds for any n-by-c matrix S assuming that **the matrix $V_k{}^T S$ has full rank** (equal to k).

Towards such a result (cont'd)

First, let the top-k right singular vectors of A be V_k.

A structural result (deterministic):

$$\|A - CX\|_F = \left\|A - CC^\dagger A\right\|_F \leq \|A - A_k\|_F + \left\|(A - A_k)\, S\, (V_k^T S)^\dagger\right\|_F$$
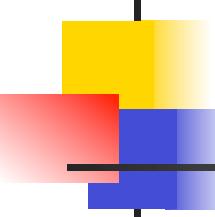
Again, this holds for any n-by-c matrix S assuming that **the matrix V_k^TS has full rank** (equal to k)

We would like to get a sampling and rescaling matrix S such that, simultaneously,

$$\sigma_{\min}(V_k^T S) \geq 1 - \sqrt{\frac{k}{c}} \qquad \text{and} \qquad \|(A - A_k)\, S\|_F \leq c_0 \|A - A_k\|_F$$

(for some small, fixed constant c_0; actually c_0 = 1 in our final result).

Setting c = O(k/ε), we get a (2+ε) constant factor approximation (for c_0 = 1).

133

Towards such a result (cont'd)

We would like to get a sampling and rescaling matrix S such that, simultaneously,
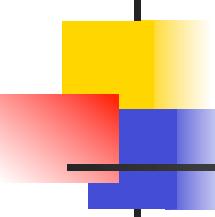
$$\sigma_{\min}\left(V_k^T S\right) \geq 1 - \sqrt{\frac{k}{c}} \qquad \text{and} \qquad \left\|(A - A_k) S\right\|_F \leq c_0 \left\|A - A_k\right\|_F$$

(for some small, fixed constant c_0).

Lamppost: the work of Batson, Spielman, & Srivastava STOC 2009 (graph sparsification)

[We had to generalize their work to use a new barrier function which controls the Frobenius and spectral norm of two matrices simultaneously. We then used a second phase of adaptive sampling to reduce the (2+ε) approximation to (1+ε).]

We will omit these details, and instead state the Batson, Spielman, & Srivastava STOC 2009 result as approximate matrix multiplication.
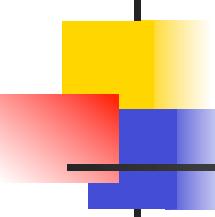
The Batson-Spielman-Srivastava result

Let V_k be an n-by-k matrix such that $V_k^TV_k = I_k$, with k < n, and let c be a sampling parameter.

There exists a deterministic algorithm which runs in $O(cnk^2)$ time and constructs an n-by-c sampling and rescaling matrix S such that for $c = k/\varepsilon^2$.

$$\left\|V_k^T V_k - V_k^T S S^T V_k\right\|_2 = \left\|I_k - V_k^T S S^T V_k\right\|_2 \leq 3\epsilon$$
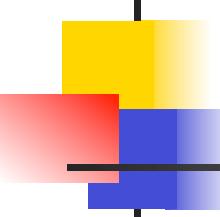
The Batson-Spielman-Srivastava result

Let V_k be an n-by-k matrix such that $V_k^TV_k = I_k$, with k < n, and let c be a sampling parameter.

There exists a deterministic algorithm which runs in $O(cnk^2)$ time and constructs an n-by-c sampling and rescaling matrix S such that for $c = k/\varepsilon^2$.

$$\left\| V_k^T V_k - V_k^T S S^T V_k \right\|_2 = \left\| I_k - V_k^T S S^T V_k \right\|_2 \le 3\epsilon$$

- It is essentially a matrix multiplication result!
- Expensive to compute, but very accurate and deterministic.
- Works for small values of the sampling parameter c.
- The rescaling in S is critical and non-trivial.
- The algorithm is basically an iterative, greedy approach that uses two potential functions to guarantee that the singular values of V_k^TS stay "away" from an upper and lower barrier.
- The algorithm selects *any* column that "respects" the two potential functions: at any iteration it does a linear search to find such columns.

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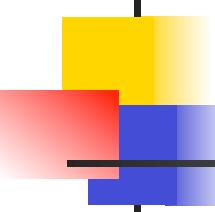
$(c+1)/(c+1-k)$ relative error bounds.

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

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

Guruswami & Sinop, FOCS 2011

Applications of column-based reconstruction in Quadratic Integer Programming.

Adaptive sampling
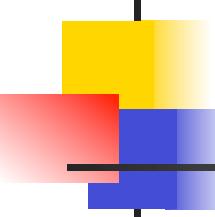
Adaptive sampling: pick columns in rounds.

➢ In the first round, pick c columns of A using our prototypical column sampling algorithm with the simple (Euclidean-norm based) probabilities:

$$p_i = \frac{\|A_{*i}\|_2^2}{\|A\|_F^2}$$

➢ At the t-th round, compute the residual matrix E = A – CC⁺A, where C is now the matrix containing **all the columns** that have been selected in the previous t-1 rounds.

➢ Compute simple (Euclidean-norm based) column sampling probabilities on E and sample columns of A according to these probabilities:

$$p_i = \frac{\|E_{*i}\|_2^2}{\|E\|_F^2}$$

Adaptive sampling

Nice, simple, intuitive idea: first appeared in Deshpande *et al.* (2006) ToC.

One can prove the following error bound: after t rounds, with probability at least 1-tδ, the resulting error is:
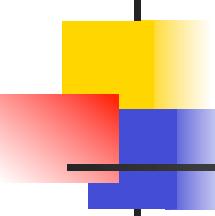
$$\|A - CC^+A\|_F^2 \leqslant \frac{1}{1-\epsilon}\|A - A_k\|_F^2 + \epsilon^t\|A\|_F^2$$

where, in each round,

$$c = O\left(\frac{k\log(1/\delta)}{\epsilon^2}\right)$$

columns of A are sampled.

Simple, inductive proof from Mahoney and Drineas (2007) LAA, using the randomized matrix multiplication bounds.

Adaptive sampling

Nice, simple, intuitive idea: first appeared in Deshpande *et al.* (2006) ToC.

One can prove the following error bound: after t rounds, with probability at least 1-tδ, the resulting error is:
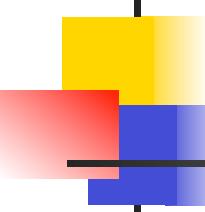
$$\|A - CC^+A\|_F^2 \leqslant \frac{1}{1-\epsilon}\|A - A_k\|_F^2 + \epsilon^t\|A\|_F^2$$

where, in each round,

$$c = O\left(\frac{k\log(1/\delta)}{\epsilon^2}\right)$$

columns of A are sampled.

In Paul, Magdon-Ismail, and Drineas NIPS 2015 we extended the above result to leverage-score sampling (a new idea was necessary there): the error after t rounds depends on $A-A_{tk}$ instead of $A-A_k$!

Volume sampling

Fundamental idea: Sample a set of columns with probability proportional to the volume that they span!

Let S be a subset of k columns of A and let Δ(S) be the volume of the simplex formed by these columns and the origin. Then, volume sampling picks a set S of columns with probability proportional to:

$$P_S = \frac{\text{vol}(\Delta(S))^2}{\sum_{T:|T|=k} \text{vol}(\Delta(T))^2}.$$

If one could sample a set of k columns of A to form an m-by-k matrix C with respect to the above probabilities, then, in expectation,

$$\mathbf{E}\left(\|A - CC^+ A\|_F^2\right) \leq (k+1)\|A - A_k\|_F^2$$

Volume sampling

Fundamental idea: Sample a set of columns with probability proportional to the volume that they span!

Let S be a subset of k columns of A and let Δ(S) be the volume of the simplex formed by these columns and the origin. Then, volume sampling picks a set S of columns with probability proportional to:

$$P_S = \frac{\mathrm{vol}(\Delta(S))^2}{\sum_{T:|T|=k} \mathrm{vol}(\Delta(T))^2}.$$

If one could sample a set of k columns of A to form an m-by-k matrix C with respect to the above probabilities, then, in expectation,

$$\mathbf{E}\left(\|A - CC^+A\|_F^2\right) \leq (k+1)\|A - A_k\|_F^2$$

The above bound could be combined with O(log k) rounds of adaptive sampling (with slightly different number of columns selected in each round to minimize the overall number of columns) to achieve a relative-error guarantee.

Volume sampling

Fundamental idea: Sample a set of columns with probability proportional to the volume that they span!
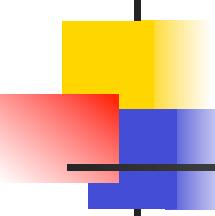
Let S be a subset of k columns of A and let Δ(S) be the volume of the simplex formed by these columns and the origin. Then, volume sampling picks a set S of columns with probability proportional to:

$$P_S = \frac{\text{vol}(\Delta(S))^2}{\sum_{T:|T|=k} \text{vol}(\Delta(T))^2}.$$

If one could sample a set of k columns of A to form an m-by-k matrix C with respect to the above probabilities, then, in expectation,

$$\mathbf{E}\left(\|A - CC^+A\|_F^2\right) \le (k+1)\|A - A_k\|_F^2$$

BUT: computing the sampling probabilities P_S is not an easy task...they must be approximated!

Volume sampling
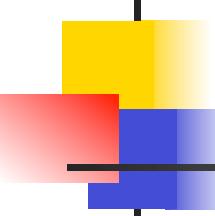
Let S be a subset of k columns of A and let Δ(S) be the volume of the simplex formed by these columns and the origin. Then, volume sampling picks a set S of columns with probability proportional to:

$$P_S = \frac{\text{vol}(\Delta(S))^2}{\sum_{T:|T|=k} \text{vol}(\Delta(T))^2}.$$

Interestingly, adaptive sampling can be used to approximate the volume sampling probabilities!

Roughly speaking, a k-round adaptive sampling algorithm where in each round one column of A is picked will pick a subset of columns S with probability $\tilde{P}_S$ that satisfies:

$$\tilde{P}_S \leq k! \, P_S$$

Volume sampling

Let S be a subset of k columns of A and let Δ(S) be the volume of the simplex formed by these columns and the origin. Then, volume sampling picks a set S of columns with probability proportional to:

$$P_S = \frac{\text{vol}(\Delta(S))^2}{\sum_{T:|T|=k} \text{vol}(\Delta(T))^2}.$$
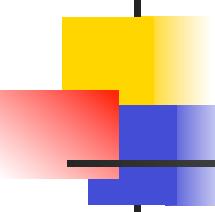
Interestingly, adaptive sampling can be used to approximate the volume sampling probabilities!

Roughly speaking, a k-round adaptive sampling algorithm where in each round one column of A is picked will pick a subset of columns S with probability $\tilde{P}_S$ that satisfies:
$$\tilde{P}_S \leq k! \, P_S$$

Using adaptive sampling to simulate volume sampling returns a set S of k columns of A to form an m-by-k matrix C such that, in expectation,

$$\mathbf{E}\left(\|A - CC^+A\|_F^2 \right) \leq (k+1)! \, \|A - A_k\|_F^2$$

Volume sampling

Using adaptive sampling to simulate volume sampling returns a set S of k columns of A to form an m-by-k matrix C such that, in expectation,
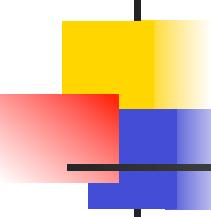
$$\mathbf{E}\left(\|A - CC^+A\|_F^2\right) \le (k+1)!\,\|A - A_k\|_F^2$$

Combining with O(k log k) rounds of adaptive sampling reduces the above error to relative error by sampling

$$O\left(\frac{k}{\epsilon} + k^2 \log k\right)$$
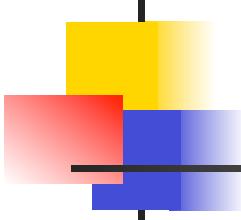
columns.

For more developments on volume sampling (including faster algorithms and relative error accuracy guarantees by selecting fewer columns) see Deshpande & Rademacher FOCS 2010 and Guruswami & Sinop SODA 2012.
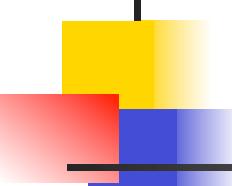
Creating matrix sketches

➢ **Sampling based**

   ➢ Adaptive sampling

   ➢ Volume sampling (work by Deshpande, Guruswami, Rademacher, Sinop, Vempala, etc.)

➢ **Element-wise sampling**

  ➢ Sample elements with probabilities that depend on the absolute value (squared or not) of the matrix entries (work by Achlioptas, Drineas, McSherry, Zouzias, etc.)

  ➢ Sample elements with respect to an element-wise notion of leverage scores

➢ **Deterministic/streaming sketches**

   ➢ Select columns/rows deterministically (work by Spielman, Srivastava, etc.)

   ➢ From item frequencies to matrix sketching (work by Liberty, Woodruff, etc.)

➢ **Random projections**

   ➢ (Slow) Pre or post-multiply by Gaussian random matrices, random sign matrices, etc.

  (work by Drineas, Magen, Zouzias, etc.)

   ➢ (Faster) Pre or post-multiply by the sub-sampled Hadamard Transform

  (work by Drineas, Sarlos, Mahoney, Muthukrishnan, etc.)

   ➢ (Sparsity) Pre- or post-multiply by ultra-sparse matrices

  (work by Clarkson, Woodruff, Mahoney, Meng, Nelson, etc.)

TA session IV
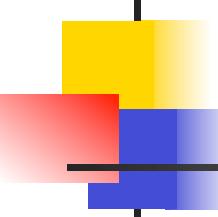
TA session IV will focus on structural inequalities and the CX decomposition and is posted at http://www.drineas.org/RandNLA-PCMI-2016/

Roadmap of my lectures

➢ **Approximating matrix multiplication** (first lecture)

➢ **Leverage scores and their applications** (second and third lectures)

      1. Over- (or under-) constrained least squares problems

      2. Feature selection and the CX decomposition

➢ **Solving systems of linear equations with Laplacian matrices** (fourth lecture)

➢ **Element-wise sampling** (fourth lecture)

Leverage scores & Laplacians

Consider a weighted (<u>positive weights only!</u>) undirected graph G and let L be the Laplacian matrix of G.

Assuming n vertices and $m > n$ edges, L is an n-by-n matrix, defined as follows:

$$L = \underbrace{\left(\quad B^T \quad \right)}_{n \times m} \cdot \underbrace{\left(\quad W \quad \right)}_{m \times m} \cdot \underbrace{\left(\quad B \quad \right)}_{m \times n}$$

Leverage scores & Laplacians

Consider a weighted (<u>positive weights only!</u>) undirected graph G and let L be the Laplacian matrix of G.

Assuming n vertices and $m > n$ edges, L is an n-by-n matrix, defined as follows:

$$L = \begin{pmatrix} & B^T & \end{pmatrix} \cdot \begin{pmatrix} & W & 0 \\ 0 & & \end{pmatrix} \cdot \begin{pmatrix} & B & \end{pmatrix}$$

$n \times m \qquad m \times m \qquad m \times n$

Diagonal matrix of edge weights

<u>Edge-incidence matrix</u>

(each row has **two non-zero entries** and corresponds to an edge; pick arbitrary orientation and use +1 and -1 to denote the "head" and "tail" node of the edge).

Clearly, $L = (B^T W^{1/2})(W^{1/2}B) = (B^T W^{1/2})(B^T W^{1/2})^T$.

Leverage scores & effective resistances

Effective resistances:

Let G denote an electrical network, in which each edge e corresponds to a resistor of resistance $1/w_e$ (the edge weight).

The effective resistance R_e between two vertices is equal to the potential difference induced between the two vertices when a unit of current is injected at one vertex and extracted at the other vertex.
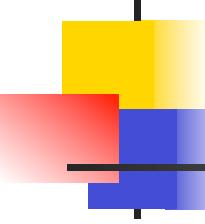
Leverage scores & effective resistances
(Spielman & Srivastava STOC 2008)

Effective resistances:

Let G denote an electrical network, in which each edge e corresponds to a resistor of resistance $1/w_e$ (the edge weight).

The effective resistance R_e between two vertices is equal to the potential difference induced between the two vertices when a unit of current is injected at one vertex and extracted at the other vertex.

Formally, the effective resistances are the diagonal entries of the m-by-m matrix:

$$R = BL^{+}B^{T} = B(B^{T}WB)^{+}B^{T}$$

Leverage scores & effective resistances
(Drineas & Mahoney ArXiv 2010)
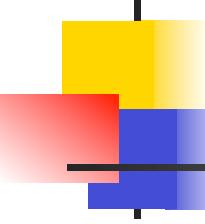
Lemma: The (row) leverage scores of the m-by-n matrix $W^{1/2}B$ are equal (up to rescaling) to the effective resistances of the edges of G.

$$\begin{pmatrix} & & \\ & W^{1/2} & \\ & & \end{pmatrix} \cdot \begin{pmatrix} & & \\ & B & \\ & & \end{pmatrix}$$

$$m \times m \qquad\qquad m \times n$$

Diagonal matrix
of edge weights

Edge-incidence matrix

Leverage scores & effective resistances

(Drineas & Mahoney ArXiv 2010)

Lemma: The (row) leverage scores of the m-by-n matrix $W^{1/2}B$ are equal (up to rescaling) to the effective resistances of the edges of G.

$$\left(\quad W^{1/2} \quad \right) \cdot \left(\quad B \quad \right)$$

$$m \times m \qquad\qquad m \times n$$
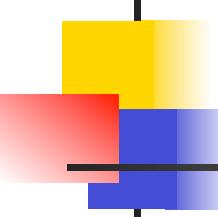
Edge-incidence matrix

Diagonal matrix
of edge weights

GRAPH SPARSIFICATION

➢ Sample r edges to sparsify our graph G with respect to the row leverage scores of $W^{1/2}B$ (equivalently, the effective resistances of the edges of G).

➢ This process sparsifies the Laplacian L to construct a sparser Laplacian.

155

Leverage scores & effective resistances

Theorem: Let L be the sparsified Laplacian that emerges by sampling r edges of G with respect to the row leverage scores of the m-by-n matrix $W^{1/2}B$.

Consider the following two least-squares problems (for any vector b):
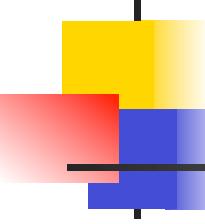
$$x_{opt} = \arg\min_{x \in \mathbb{R}^n} \|Lx - b\|_2 = L^+ b$$

$$\tilde{x}_{opt} = \arg\min_{x \in \mathbb{R}^n} \left\|\tilde{L}x - b\right\|_2 = \tilde{L}^+ b$$

Notation:
$x^{\mathsf{T}}Lx = \|x\|_L$: energy norm
(as in the Spielman & Teng work)

Let $r = O\left(\frac{n}{\epsilon} \ln \frac{n}{\epsilon}\right)$

Then, with probability at least 2/3: $\left\|x_{opt} - \tilde{x}_{opt}\right\|_L \leq \epsilon \left\|x_{opt}\right\|_L$

Leverage scores & effective resistances

Theorem: Let L be the sparsified Laplacian that emerges by sampling r edges of G with respect to the row leverage scores of the m-by-n matrix $W^{1/2}B$.

Consider the following two least-squares problems (for any vector b):
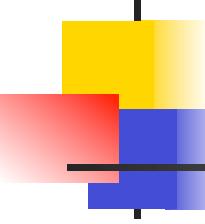
$$x_{opt} = \arg\min_{x \in \mathbb{R}^n} \|Lx - b\|_2 = L^+ b$$

$$\tilde{x}_{opt} = \arg\min_{x \in \mathbb{R}^n} \left\|\tilde{L}x - b\right\|_2 = \tilde{L}^+ b$$

Let $r = O\left(\frac{n}{\epsilon} \ln \frac{n}{\epsilon}\right)$

Then, with probability at least 2/3: $\|x_{opt} - \tilde{x}_{opt}\|_L \leq \epsilon \|x_{opt}\|_L$

The proof is relatively simple: we write down the SVD-based closed-form formulas for the x_{opt} and $\mathcal{X}_{opt}$ and bound their energy norm difference directly; we need to use the fact that sampling edges amounts to sampling rows of B.

157

Leverage scores & effective resistances

Theorem: Let L be the sparsified Laplacian that emerges by sampling r edges of G with respect to the row leverage scores of the m-by-n matrix $W^{1/2}B$.

Consider the following two least-squares problems (for any vector b):
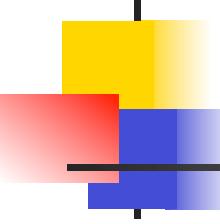
$$x_{opt} = \arg \min_{x \in \mathbb{R}^n} \|Lx - b\|_2 = L^+ b$$

$$\tilde{x}_{opt} = \arg \min_{x \in \mathbb{R}^n} \left\|\tilde{L}x - b\right\|_2 = \tilde{L}^+ b$$

Let $r = O\left(\frac{n}{\epsilon} \ln \frac{n}{\epsilon}\right)$

Then, with probability at least 2/3: $\|x_{opt} - \tilde{x}_{opt}\|_L \leq \epsilon \|x_{opt}\|_L$

Computational savings depend on **(i)** efficiently computing leverage scores/effective resistances, and **(ii)** efficiently solving the "sparse" problem.

Running time issues

Approximating effective resistances (Spielman & Srivastava STOC 2008)

They can be approximated using the Laplacian solver of Spielman and Teng.

Breakthrough by Koutis, Miller, & Peng (FOCS 2010, FOCS 2011):

Low-stretch spanning trees provide a means to approximate effective resistances!

This observation (and a new, improved algorithm to approximate low-stretch spanning trees) led to almost optimal algorithms for solving Laplacian systems of linear equations.

Are leverage scores a viable alternative to approximate effective resistances?

Not yet! Our approximation algorithms are not good enough for $W^{1/2}B$, which is very sparse.

($2m$ non-zero entries).

We must take advantage of the sparsity and approximate the leverage scores/effective resistances in $O(m \, polylog(m))$ time.

Running time issues

Approximating effective resistances (Spielman & Srivastava STOC 2008)

They can be approximated using the Laplacian solver of Spielman and Teng.

Breakthrough by Koutis, Miller, & Peng (FOCS 2010, FOCS 2011):

Low-stretch spanning trees provide a means to approximate effective resistances!

This observation (and a new, improved algorithm to approximate low-stretch spanning trees) led to almost optimal algorithms for solving Laplacian systems of linear equations.

Are leverage scores a viable alternative to approximate effective resistances?

Not yet! Our approximation algorithms are not good enough for $W^{1/2}B$, which is very sparse.

($2m$ non-zero entries).

We must take advantage of the sparsity and approximate the leverage scores/effective resistances in $O(m \, \text{polylog}(m))$ time.

Roadmap of my lectures

➢ **Approximating matrix multiplication** (first lecture)

➢ **Leverage scores and their applications** (second and third lectures)

      1. Over- (or under-) constrained least squares problems

      2. Feature selection and the CX decomposition

➢ **Solving systems of linear equations with Laplacian matrices** (fourth lecture)

➢ **Element-wise sampling** (fourth lecture)

RandNLA: from row/column sampling

Sampling rows (or columns) from a matrix

Input: m-by-n matrix A, sampling parameter r

Output: r-by-n matrix R, consisting of r rows of A

- Let p_i for $i=1...m$ be sampling probabilities summing up to 1;
- In r i.i.d. trials (with replacement) pick r rows of A;

  (In each trial the i-th row of A is picked with probability p_i.)

- Let R be the matrix consisting of the rows;

  (We rescale the rows of A prior to including them in R by $1/(rp_i)^{1/2}$.)

$$\begin{pmatrix} & & \\ & A & \\ & & \end{pmatrix} \implies \begin{pmatrix} & R & \\ & r \times n & \end{pmatrix}$$

$$m \times n$$

… to element-wise sampling

Sampling algorithm

Input: m-by-n matrix A, sampling parameter s

Output: sparse m-by-n matrix $S_\Omega(A)$ consisting of s elements of A

• Let p_{ij} (for all (i,j) in $\{1…m\}\times\{1…n\}$) be sampling probabilities summing up to 1;

• In s i.i.d. trials (with replacement) pick s elements of A;

(In each trial the (i,j)-th element of A is picked with probability p_{ij})

• Let $S_\Omega(A)$ be the matrix consisting of the selected elements, rescaled;

($\Omega = \{(i_t, j_t), t=1…s\}$ is the set of sampled pairs of indices)

Element-wise sampling

Sampling algorithm

Input: m-by-n matrix A, sampling parameter s

Output: sparse m-by-n matrix $S_\Omega(A)$ consisting of s elements of A

- Let p_{ij} (for all (i,j) in $\{1...m\}\times\{1...n\}$) be sampling probabilities summing up to 1;
- In s i.i.d. trials (with replacement) pick s elements of A;

    (In each trial the (i,j)-th element of A is picked with probability p_{ij})

- Let $S_\Omega(A)$ be the matrix consisting of the selected elements, rescaled;

    ($\Omega = \{(i_t, j_t),\ t=1...s\}$ is the set of sampled pairs of indices)

$$\begin{pmatrix} & & \\ & A & \\ & m \times n & \end{pmatrix} \text{is} \qquad A = \sum_{i,j=1}^{m,n} A_{ij} e_i e_j^T$$

Element-wise sampling

Input: m-by-n matrix A, sampling parameter s

Output: sparse m-by-n matrix $S_\Omega(A)$ consisting of s elements of A

- Let p_{ij} (for all (i,j) in $\{1...m\}\times\{1...n\}$) be sampling probabilities summing up to 1;
- In s i.i.d. trials (with replacement) pick s elements of A;

   (In each trial the (i,j)-th element of A is picked with probability p_{ij})

- Let $S_\Omega(A)$ be the matrix consisting of the selected elements, rescaled;

   ($\Omega = \{(i_t, j_t),\ t=1...s\}$ is the set of sampled pairs of indices)

$$
\begin{pmatrix} & & \\ & S_\Omega(A) & \\ & & \\ & m \times n & \end{pmatrix} \quad \text{is} \quad \mathcal{S}_\Omega(A) = \sum_{t=1}^{s} \frac{1}{s p_{i_t j_t}} A_{i_t j_t} e_{i_t} e_{j_t}^T
$$

165

Motivation

A is an m-by-n data matrix, representing m objects with respect to n features.

Succinct Representation of Data Matrices
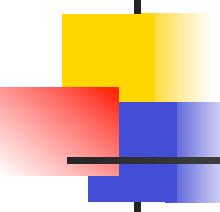
Sparsify the input matrix and work with the sparse "sketch" in downstream data analysis applications (e.g., clustering and/or classification tasks).

Potential advantages of a sparse sketch: (i) faster computation, (ii) less communication across different processors, and (iii) regularization of the input problem.

Exploratory Data Analysis
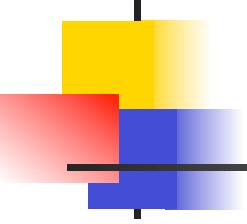
Goal: identify object-feature combinations that exert disproportionate influence on the matrix based on a notion of *element-wise leverage scores*.
Such entries of A could be outliers or really important object-feature combinations.

Element-wise sampling: overview

- Achlioptas and McSherry STOC 2001 (JACM 2007) introduced element-wise sampling and presented additive-error bounds, similar to the work on row/column sampling with respect to row/column Euclidean norms.

- **Matrix completion:** introduced by Candes and Recht in 2008. Massive amount of follow-up work, guarantees exact reconstruction of a matrix from a uniform (!) sample of entries, under (very) strong assumptions on incoherence (in our parlance, the leverage scores).

- **Current state-of-the-art in matrix completion (Chen et al. ICML 2014):** exact reconstruction (zero relative error) for arbitrary *low-rank matrices* using element-wise leverage scores.

- Even more recent results exist; will be discussed later…

- **Open (?) problem: relative error guarantees for arbitrary matrices.**

Element-wise sampling: additive error

Let the sampling probabilities p_{ij} be **(roughly)**:

$$p_{ij} = \frac{A_{ij}^2}{\|A\|_F^2}$$
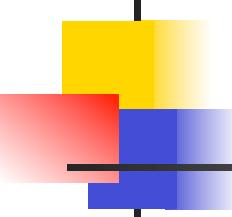
Let the number of sampled entries s be at least:

$$s = \Omega\left((m+n)\,\mathrm{polylog}\,(m+n)\,/\epsilon^2\right)$$

Then, with high probability (typically at least $1-(m+n)^{-1}$):

$$\|A - \mathcal{S}_\Omega(A)\|_2 \leq \epsilon \|A\|_F$$

Recall that $S_\Omega(A)$ is a matrix consisting of the (rescaled) sampled entries of A.

Element-wise sampling: additive error

Let the sampling probabilities p_{ij} be **(roughly):**

$$p_{ij} = \frac{A_{ij}^2}{\|A\|_F^2}$$

Let the number of sampled entries s be at least:
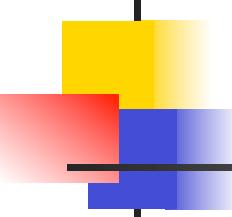
$$s = \Omega\left((m+n)\operatorname{polylog}(m+n)/\epsilon^2\right)$$

Then, with high probability (typically at least $1-(m+n)^{-1}$):

$$\|A - \mathcal{S}_\Omega(A)\|_2 \leq \epsilon \|A\|_F$$

Let $(S_\Omega(A))_k$ be the best rank-k approximation to $S_\Omega(A)$; then, we can prove that:

$$\|A - (\mathcal{S}_\Omega(A))_k\|_2 \leq \|A - A_k\|_2 + \epsilon \|A\|_F$$

Element-wise sampling: additive error

Let the sampling probabilities p_{ij} be **(roughly):**

$$p_{ij} = \frac{A_{ij}^2}{\|A\|_F^2}$$

We need to worry about small entries; rescaling them causes trouble.

Let the number of sampled entries s be at least:
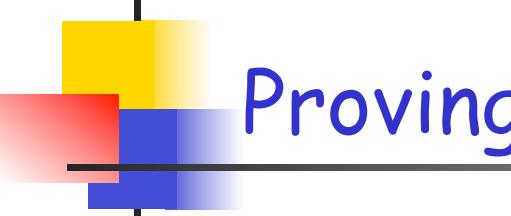
$$s = \Omega\left((m+n)\operatorname{polylog}(m+n)/\epsilon^2\right)$$

Then, with high probability (typically at least $1-(m+n)^{-1}$):

$$\|A - \mathcal{S}_\Omega(A)\|_2 \le \epsilon\|A\|_F$$

Let $(S_\Omega(A))_k$ be the best rank-k approximation to $S_\Omega(A)$; then, we can prove that:
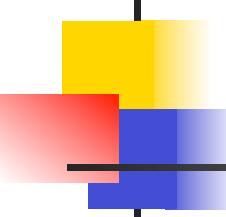
$$\|A - (\mathcal{S}_\Omega(A))_k\|_2 \le \|A - A_k\|_2 + \epsilon\|A\|_F$$

Proving the last inequality

Let $S_\Omega(A)$ be denoted by A for notational simplicity.

$$\left\| A - (\tilde{A})_k \right\|_2 \leq \|A - A_k\|_2 + 2 \left\| A - \tilde{A} \right\|_2$$

Ideal error

Sparsification error

This result actually holds for any two matrices A and A and allows us to approximate the best rank-k approximation to A by the best rank-k approximation to A.

Very simple proof.

Element-wise sampling: additive error
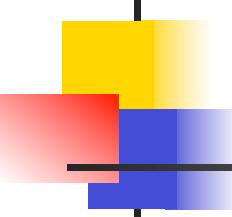
Achlioptas & McSherry STOC 2001, JACM 2007

Dealing with small entries: Sample them with higher probability, proportional to the absolute value of A_{ij} (times polylog factors); relatively large exponent in the polylog factor.

The proof is straight-forward using a wonderful "blackbox": a result of Füredi & Komlós (Combinatorica 1981) on the largest eigenvalue of a random symmetric matrix.

The Füredi & Komlós proof is based on the "moments method" (dating back to Wigner 1955) :

$$\sum_{i=1}^{n} \lambda_i^k = \text{trace } A^k$$

Element-wise sampling: additive error
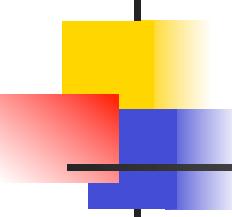
Achlioptas & McSherry STOC 2001, JACM 2007

<u>Dealing with small entries:</u> Sample them with higher probability, proportional to the absolute value of A_{ij} (times polylog factors); relatively large exponent in the polylog factor.

The proof is straight-forward using a wonderful "blackbox": a result of Füredi & Komlós (Combinatorica 1981) on the largest eigenvalue of a random symmetric matrix.

The Füredi & Komlós proof is based on the "moments method" (dating back to Wigner 1955) :

$$\sum_{i=1}^{n} \lambda_i^k = \text{trace } A^k$$

• The above equality holds for the *eigenvalues* of a symmetric A and all k.

• It can be applied for a large *even* k to upper bound the largest eigenvalue of A.

• Nice and elegant as an idea, but bounding the trace of A^k boils down to a tough and tricky combinatorial exercice.

• Also used by Nelson & Huy FOCS 2013 to improve the analysis of the sparse random projection presented by Clarkson & Woodruff STOC 2013 (also analyzed by Mahoney & Meng STOC 2013).
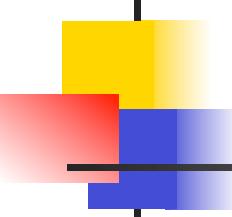
Element-wise sampling: additive error

<u>**Drineas & Zouzias IPL 2011**</u>

• Zero out "sufficiently small" entries of A and sample from the remaining entries with probabilities that depend on A_{ij}^2.

• But, one needs to know the threshold *a priori*…

• **A major plus:** we used a matrix-Bernstein inequality and the proof is very simple.

<u>**Tensor extension:**</u> Nguyen, Drineas & Tran (Information and Inference, IMA 2014):

> ➤ Since no tensor-Bernstein inequality exists, we had to come up with an appropriate inequality using the entropy-concentration method of Rudelson & Vershynin.

> ➤ Extremely complicated (and under-appreciated …)

Element-wise sampling: additive error

Achlioptas, Karnin & Liberty NIPS 2013
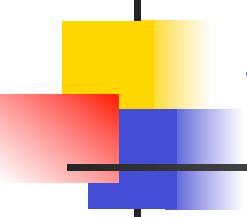
• Sample entries with respect to the absolute value of A_{ij}.

• No need to zero out small entries, so no *a priori* thresholds.

• The bounds are comparable to Drineas & Zouzias IPL 2011.

• Again, a simple proof via matrix-Bernstein.

Kundu & Drineas ArXiv 2014

Sample entries with respect to both their absolute value and the square of the absolute value, without zeroing out any small entries; more precisely, for all (i,j) in {1…m}x{1…n}:

$$p_{ij} = \frac{A_{ij}^2}{2\,\|A\|_F^2} + \frac{|A_{ij}|}{2\sum_{i,j=1}^{m,n} |A_{ij}|}$$

<u>Obvious variant:</u> use different weights for the two sampling probabilities.

The proof

Recall:

$$A = \sum_{i,j=1}^{m,n} A_{ij} e_i e_j^T \quad \in \quad \mathbb{R}^{m \times n}.$$

$$\tilde{A} = \mathcal{S}_\Omega(A) = \frac{1}{s} \sum_{t=1}^{s} \frac{A_{i_t j_t}}{p_{i_t j_t}} e_{i_t} e_{j_t}^T \quad \in \quad \mathbb{R}^{m \times n}.$$
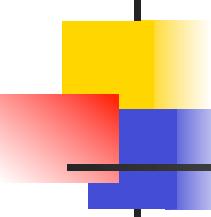
Easy to prove: $\quad \mathbb{E}\left[\tilde{A} - A\right] = 0_{m \times n}$

Sampling probabilities:

$$(L_1 \text{ probabilities}) \quad p_{ij} = \frac{|A_{ij}|}{\|A\|_1} \qquad \|A\|_1 = \sum_{i,j=1}^{m,n} |A_{ij}|$$

$$(L_2 \text{ probabilities}) \quad p_{ij} = \frac{A_{ij}^2}{\|A\|_F^2}$$

A matrix-Bernstein inequality

(Recht 2009)

Let $\mathbf{M}_1, ..., \mathbf{M}_s$ be independent, zero-mean random matrices in $\mathbb{R}^{m \times n}$. Suppose, for all $t \in [s]$,

$$\|\mathbf{M}_t\|_2 \leq \gamma$$

and
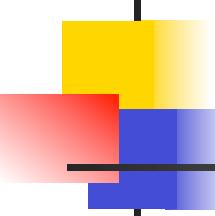
$$\max_{t \in [s]} \left\{ \left\|\mathbb{E}(\mathbf{M}_t \mathbf{M}_t^T)\right\|_2, \left\|\mathbb{E}(\mathbf{M}_t^T \mathbf{M}_t)\right\|_2 \right\} \leq \rho^2.$$

Then, for any $\epsilon > 0$,

$$\left\| \frac{1}{s} \sum_{t=1}^{s} \mathbf{M}_t \right\|_2 \leq \epsilon$$

holds, subject to a failure probability at most

$$(m+n) \exp\left(\frac{-s\epsilon^2/2}{\rho^2 + \gamma\epsilon/3} \right).$$

177

A matrix-Bernstein inequality

(Recht 2009)

Let $\mathbf{M}_1, ..., \mathbf{M}_s$ be independent, zero-mean random matrices in $\mathbb{R}^{m \times n}$. Suppose, for all $t \in [s]$,

$$\|\mathbf{M}_t\|_2 \leq \gamma$$

and

$$\max_{t \in [s]} \left\{ \left\|\mathbb{E}(\mathbf{M}_t \mathbf{M}_t^T)\right\|_2, \left\|\mathbb{E}(\mathbf{M}_t^T \mathbf{M}_t)\right\|_2 \right\} \leq \rho^2.$$
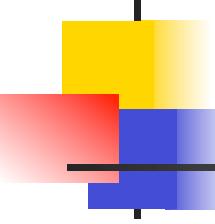
Then, for any $\epsilon > 0$,

$$\left\| \frac{1}{s} \sum_{t=1}^{s} \mathbf{M}_t \right\|_2 \leq \epsilon$$

We will complete the proof using this matrix-Bernstein inequality and the sampling probabilities of slide 175 in the TA session.

holds, subject to a failure probability at most

$$(m+n) \exp\left(\frac{-s\epsilon^2/2}{\rho^2 + \gamma\epsilon/3} \right).$$

Matrix completion

Given an m-by-n matrix A and a set $\Omega = \{(i_t, j_t), t=1...s\}$ consisting of pairs of indices, consider the following optimization problem:

$$\min_{\Phi \in \mathbb{R}^{m \times n}} \|\Phi\|_* = \sum_i \sigma_i(\Phi)$$

$$\text{subject to: } \Phi_{i_t j_t} = A_{i_t j_t}, \text{for all } (i_t, j_t) \in \Omega$$

Matrix completion

Given an m-by-n matrix A and a set $\Omega = \{(i_t, j_t), t=1...s\}$ consisting of pairs of indices, consider the following optimization problem:

$$\min_{\Phi \in \mathbb{R}^{m \times n}} \|\Phi\|_* = \sum_i \sigma_i(\Phi)$$

$$\text{subject to: } \Phi_{i_t j_t} = A_{i_t j_t}, \text{for all } (i_t, j_t) \in \Omega$$

In words, given a set of entries of A, we seek a matrix Φ that agrees with the matrix A on the given entries and has minimal nuclear norm.

Matrix completion
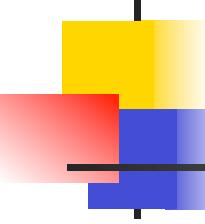
(Landmark papers: Candes and Recht 2009, Candes and Tao 2010, Recht 2011, Gross 2011)

Given an m-by-n matrix A and a set $\Omega = \{(i_t, j_t), t=1...s\}$ consisting of pairs of indices, consider the following optimization problem:

$$\min_{\Phi \in \mathbb{R}^{m \times n}} \|\Phi\|_* = \sum_i \sigma_i (\Phi)$$

$$\text{subject to: } \Phi_{i_t j_t} = A_{i_t j_t}, \text{ for all } (i_t, j_t) \in \Omega$$

In words, given a set of entries of A, we seek a matrix Φ that agrees with the matrix A on the given entries and has minimal nuclear norm.

- This problem is convex and can be solved in polynomial time.
- Assume that A is **low-rank and incoherent**: e.g., A has rank ρ << min{m,n} and the leverage scores of the left and right singular vectors of A are uniform.
- Then, with high probability, A can be reconstructed **exactly** from a uniform sample of s = O ((m+n) ρ polylog(m+n)) entries.
- In other words, solving the above optimization problem, returns A as the unique minimizer.

The input matrix A

Our input is an m-by-n matrix A of rank ρ << min{m,n}:

$$A = \begin{pmatrix} & U & \\ & & \end{pmatrix} \begin{pmatrix} & \Sigma & \\ & & \end{pmatrix} \begin{pmatrix} & V^T & \\ & & \end{pmatrix}$$

$m \times \rho$ $\rho \times \rho$ $\rho \times n$

U_{i*}

V_{j*}

The input matrix A

Our input is an m-by-n matrix A of rank ρ << min{m,n}:

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

U_{i*}

V_{j*}

Incoherence assumptions

Appear in almost all prior work on matrix completion – recently removed

$$\|U_{i*}\|_2^2 \approx O\left(\frac{\rho}{m} \ln m\right)$$

$$\|V_{j*}\|_2^2 \approx O\left(\frac{\rho}{n} \ln n\right)$$

Element-wise leverage scores
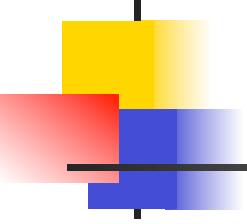
➢ For any m-by-n matrix A of rank ρ, $A = U\Sigma V^\top$, keep entry (i,j) with probability p_{ij} such that

$$p_{ij}^{[1]} \geq \min\left\{c_0 \frac{(\mu_i + \nu_j)\rho \log^2(m+n)}{\min\{m,n\}}, \quad 1\right\}$$

$$p_{ij}^{[1]} \geq \frac{1}{\min\{m,n\}^{10}}$$

Here, μ_i and ν_j are a rescaled version of the leverage scores for row i and column j.

Then, solving the nuclear norm minimization problem returns A as the unique solution (with probability at least $1 - (m+n)^{-1}$) by keeping (in expectation) $O((m+n)\rho \log^2(m+n))$ entries of A.

➢ Improves Version 1 of their Aug 2013 ArXiv paper which necessitated $O((m+n)^{1.5}\rho\log^2(m+n))$ samples.

➢ A very involved proof.
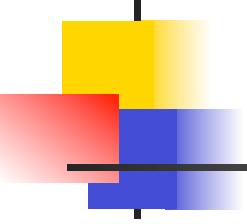
184

More progress

➢ We adapt the notation of the ArXiv Oct 16, 2014 version of their paper.

➢ **Main result:** for any m-by-n matrix A with m > n of arbitrary rank, we can get an approximation $\tilde{A}_k$ to A that is almost as good as A_k (with probability at least 1-δ):

$$\left\| A - \tilde{A}_k \right\|_2 \leq \left\| A - A_k \right\|_2 + \epsilon \left\| A - A_k \right\|_F + \zeta$$

More progress
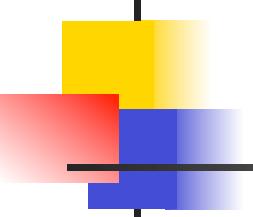
➤ We adapt the notation of the ArXiv Oct 16, 2014 version of their paper.

➤ **Main result:** for any m-by-n matrix A with m > n of arbitrary rank, we can get an approximation $\tilde{A}_k$ to A that is almost as good as A_k (with probability at least 1-δ):

$$\left\| A - \tilde{A}_k \right\|_2 \leq \| A - A_k \|_2 + \epsilon \| A - A_k \|_F + \zeta$$

➤ **What kind of info do we need from the matrix?** A sample of s entries, where

$$s = O \left(\frac{m \log m}{\delta \epsilon^2} k^3 \kappa^2 \log \left(\frac{\|A\|_2}{\zeta} \right) \right)$$

More progress

Bhojanapalli, Jain & Sanghavi SODA 2015
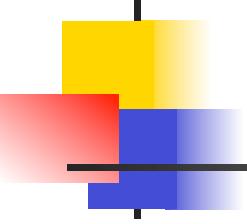
➤ We adapt the notation of the ArXiv Oct 16, 2014 version of their paper.

➤ **Main result:** for any m-by-n matrix A with m > n of arbitrary rank, we can get an approximation $\tilde{A}_k$ to A that is almost as good as A_k (with probability at least 1-δ):

$$\left\| A - \tilde{A}_k \right\|_2 \le \|A - A_k\|_2 + \epsilon \|A - A_k\|_F + \zeta$$

➤ **What kind of info do we need from the matrix?** A sample of s entries, where

$$s = O\left(\frac{m \log m}{\delta \epsilon^2} k^3 \kappa^2 \log\left(\frac{\|A\|_2}{\zeta} \right) \right) \qquad \kappa = \frac{\sigma_1(A)}{\sigma_k(A)}$$

More progress

Bhojanapalli, Jain & Sanghavi SODA 2015

➢ We adapt the notation of the ArXiv Oct 16, 2014 version of their paper.

➢ **Main result:** for any m-by-n matrix A with m > n of arbitrary rank, we can get an approximation $\tilde{A}_k$ to A that is almost as good as A_k (with probability at least 1-δ):
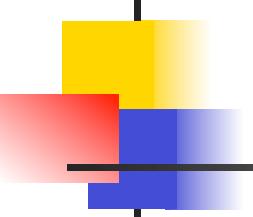
$$\left\| A - \tilde{A}_k \right\|_2 \leq \left\| A - A_k \right\|_2 + \epsilon \left\| A - A_k \right\|_F + \zeta$$

➢ **What kind of info do we need from the matrix?** A sample of s entries, where

$$s = O\left(\frac{m \log m}{\delta \epsilon^2} k^3 \kappa^2 \log\left(\frac{\|A\|_2}{\zeta} \right) \right)$$

➢ **How do we sample?** We sample each entry, independently, with probability

$$p_{ij} = \min\left\{ s\left(\frac{\|A_{i*}\|_2^2 + \|A_{*j}\|_2^2}{2(m+n)\|A\|_F^2} + \frac{|A_{ij}|}{2\sum_{i,j}|A_{ij}|} \right), 1 \right\}$$
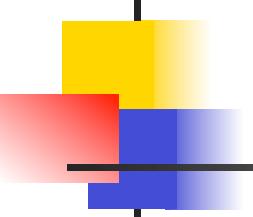
188

More progress
Bhojanapalli, Jain & Sanghavi SODA 2015

➤ We adapt the notation of the ArXiv Oct 16, 2014 version of their paper.

➤ **Main result:** for any m-by-n matrix A with m > n of arbitrary rank, we can get an approximation $\tilde{A}_k$ to A that is almost as good as A_k (with probability at least 1-δ):

$$\left\| A - \tilde{A}_k \right\|_2 \leq \left\| A - A_k \right\|_2 + \epsilon \left\| A - A_k \right\|_F + \zeta$$

➤ **The algorithm:**

- An alternating minimization approach attempting to find the best fit to the observed entries.

- Runs in rounds, but only uses the samples that have been collected a priori.

➤ **The running time is**

$$O\left(\mathrm{nnz}\left(A\right) + sk^2\right)$$

$$s = O\left(\frac{m \log m}{\delta \epsilon^2} k^3 \kappa^2 \log\left(\frac{\|A\|_2}{\zeta}\right)\right)$$

Observations…

Bhojanapalli, Jain & Sanghavi SODA 2015

Table 1 in the paper claims a relative error bound with respect to the Frobenius norm, e.g.,

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

A great result!

- The proof does not seem to be included in the paper…

- The above result is achieved via element-wise sampling and without any use of leverage scores or some adaptive sampling procedure!

- Not sure if we even have an analog of this for row/column sampling!

- The number of samples depends on κ, but (imho) this is quite mild and the authors even claim that they might be able to reduce this to $\log(\kappa)$!

<u>**Comparison to matrix completion?**</u>

If $A = A_k$, the resulting error does not reduce to zero.
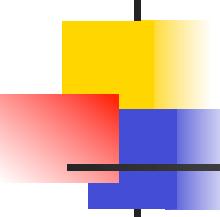
Is the dependency on this (weak) condition number κ so powerful?

TA session V

TA session V will focus on effective resistances vs. leverage scores and element-wise sampling and is posted at

http://www.drineas.org/RandNLA-PCMI-2016/

Conclusions

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

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

- **Row/column/element-wise leverage scores** are fundamental in sampling-based approaches.