RandNLA: Randomized Numerical Linear Algebra

Petros Drineas

Rensselaer Polytechnic Institute
Computer Science Department

To access my web page:

Google drineas

RandNLA: "sketch" a matrix by row/column sampling

<u>Sampling algorithm (rows)</u>

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 $i=1\ldots m$ 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 $1/(rp_i)^{1/2}$.)

$$\underset{m \times n}{\begin{pmatrix} \\ A \\ \\ \end{pmatrix}} \Longrightarrow \underset{r \times n}{\begin{pmatrix} R \end{pmatrix}}$$

RandNLA: "sketch" a matrix by row/ column sampling

<u>Sampling algorithm (rows)</u>

Input: m-by-n matrix A, sampling par

Output: r-by-n matrix R, consist

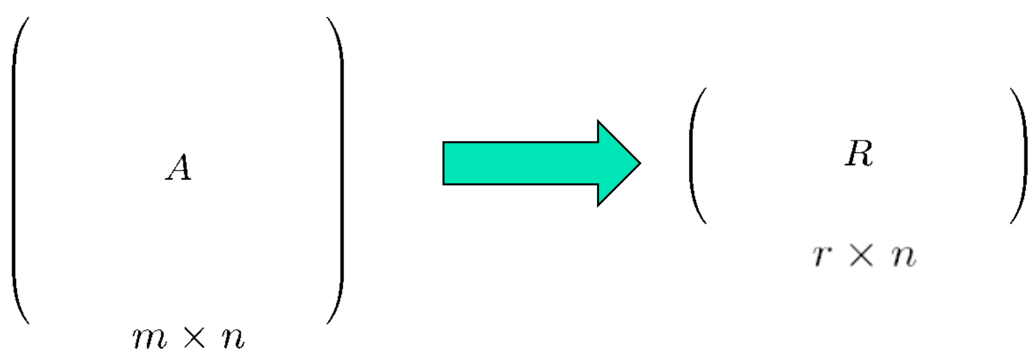
- Let p_i for $i=1\ldots m$ be sampling prob
- In r i.i.d. trials (with replacement) pick

(In each trial the i-th row of 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 $1/(rp_i)^{1/2}$.)

Column sampling is equivalent to row sampling, simply by working with A^T instead 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.,

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

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.

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

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

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*.

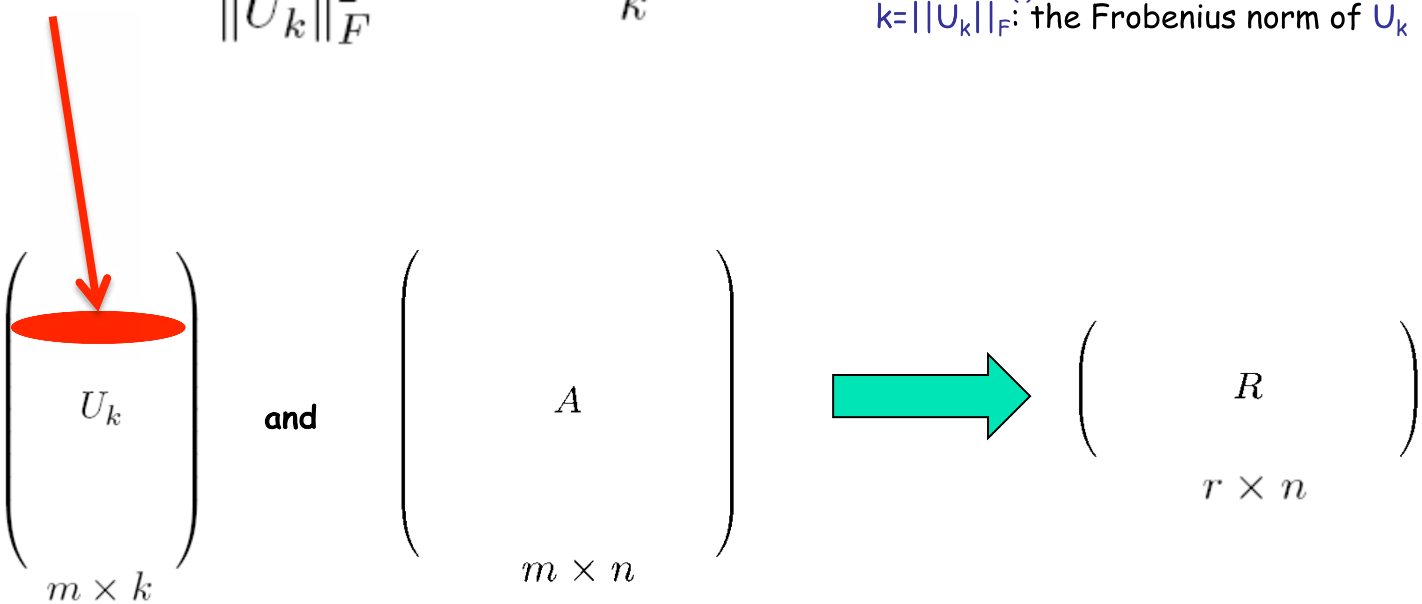
$$p_i = \frac{\left\|(U_k)_{(i)}\right\|_2^2}{\|U_k\|_F^2} = \frac{\left\|(U_k)_{(i)}\right\|_2^2}{k}$$

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



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*.

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

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:

- low-rank matrix approximations and the Singular Value Decomposition (SVD),
- the CUR and CX factorizations,
- Over- and under- constrained least-squares problems
- 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*.

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

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^T 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 $n \gg d$ whose SVD is:

$$\underbrace{\begin{pmatrix} \\ \\ A \\ \\ \\ \end{pmatrix}}_{n \times d \;\;,\;\; n \gg d} = \underbrace{\begin{pmatrix} \\ \\ U \\ \\ \\ \end{pmatrix}}_{n \times d} \underbrace{\begin{pmatrix} \Sigma \end{pmatrix}}_{d \times d} \underbrace{\begin{pmatrix} V^T \end{pmatrix}}_{d \times d}$$

Leverage scores: tall & thin matrices

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

$$\underset{n \times d \;\;,\;\; n \gg d}{\begin{pmatrix} \\ \\ A \\ \\ \\ \end{pmatrix}} = \underset{n \times d}{\begin{pmatrix} \\ \\ U \\ \\ \\ \end{pmatrix}} \underset{d \times d}{\begin{pmatrix} \Sigma \end{pmatrix}} \underset{d \times d}{\begin{pmatrix} V^T \end{pmatrix}}$$

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 $n \gg d$:

$$\underset{d \times n}{\left(\quad A \quad \right)} = \underset{d \times d}{\left(\quad U \quad \right)} \underset{d \times d}{\left(\quad \Sigma \quad \right)} \underset{d \times n}{\left(\quad V^T \quad \right)}$$

Leverage scores: short & fat matrices

Let A be a (full rank) d-by-n matrix with $n \gg d$:

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

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

(Column) Leverage scores: $p_j = \dfrac{\left\| \left(V^T\right)^{(j)} \right\|_2^2}{\|V^T\|_F^2} = \dfrac{\left\| \left(V^T\right)^{(j)} \right\|_2^2}{d}$

(set k to d)

The (column) leverage scores can now be used to sample rows 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) :

$$A \approx \underbrace{\left(\quad A_k \quad\right)}_{m \times n} = \underbrace{\left(\quad U_k \quad\right)}_{m \times k} \cdot \underbrace{\left(\quad \Sigma_k \quad\right)}_{k \times k} \cdot \underbrace{\left(\quad V_k^T \quad\right)}_{k \times n}$$

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) :

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

i-th row of U_k

j-th column of V^T

(Row) Leverage scores:

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

(Column) Leverage scores:

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

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

Other ways to create matrix sketches

- **Sampling based**
  - Volume sampling: see **Amit Deshpande's talk** tomorrow.
- **Random projections**
  - Pre or post-multiply by Gaussian random matrices, random sign matrices, etc.
  - (Faster) Pre or post-multiply by the sub-sampled Hadamard Transform.
  - (Sparsity) Pre- or post-multiply by ultra-sparse matrices (**Michael Mahoney's talk**).
- **Deterministic/streaming sketches**
  - Select columns/rows deterministically (some ideas in **Nikhil Srivastava's talk** on graph sparsification).
  - From item frequencies to matrix sketching (see **Edo Liberty's talk**).
- **Element-wise sampling**
  - Sample elements with probabilities that depend on the absolute value (squared or not) of the matrix entries.
  - Sample elements with respect to an element-wise notion of leverage scores!
- **Beyond matrices: tensors** **(Ravi Kannan's talk)**

Applications of leverage scores

- Over (or under)-constrained Least Squares problems
- Feature selection and the CX factorization
- Solving systems of linear equations with Laplacian input matrices
- Element-wise sampling

Applications of leverage scores

- Over (or under)-constrained Least Squares problems
- Feature selection and the CX factorization
- Solving systems of linear equations with Laplacian input matrices
- Element-wise sampling

<u>BUT FIRST THINGS FIRST:</u>

- Why do they work?
- How fast can we compute them?

Why do they work?

ALL proofs that use leverage score sampling use an argument of the following form:

U is an orthogonal matrix: $\begin{pmatrix} \\ \\ U \\ \\ \\ \end{pmatrix}$

$U^T U = I_d$

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

Sample/rescale r rows of U w.r.t. the leverage scores (use the sampling algorithm of slide 2):

$$p_i = \frac{\left\|U_{(i)}\right\|_2^2}{\|U\|_F^2} = \frac{\left\|U_{(i)}\right\|_2^2}{d}$$

Why do they work?

ALL proofs that use leverage score sampling use an argument of the following form:

U is an orthogonal matrix:

$U^T U = I_d$

$$\underset{n \times d \;\; , \;\; n \gg d}{\begin{pmatrix} \\ \\ U \\ \\ \\ \end{pmatrix}} \Longrightarrow \underset{r \times d}{\begin{pmatrix} \tilde{U} \end{pmatrix}}$$

Sample/rescale r rows of U w.r.t. the leverage scores (use the sampling algorithm of slide 2):

$$p_i = \frac{\|U_{(i)}\|_2^2}{\|U\|_F^2} = \frac{\|U_{(i)}\|_2^2}{d}$$

Why do they work?

ALL proofs that use leverage score sampling use an argument of the following form:

$\tilde{U}$ is a full-rank matrix!

U is an orthogonal matrix:

$U^T U = I_d$

$$\begin{pmatrix} \\ \\ U \\ \\ \\ \end{pmatrix} \Rightarrow \begin{pmatrix} \tilde{U} \end{pmatrix}$$

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

$r \times d$

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

Then, with probability at least $1-\delta$:

$$\left\| U^T U - \tilde{U}^T \tilde{U} \right\|_2 = \left\| I - \tilde{U}^T \tilde{U} \right\|_2 \leq \varepsilon$$

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

Why do they work?

Recall: with probability at least 1-δ:

$$\left\|U^T U - \tilde{U}^T \tilde{U}\right\|_2 = \left\|I - \tilde{U}^T \tilde{U}\right\|_2 \leq \varepsilon$$

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

- This implies that Ũ has full rank.
- The result follows from randomized matrix multiplication algorithms and a matrix-Bernstein bound.

  (see the tutorials by Drineas, Mahoney, and Tropp at the Simons Big Data Bootcamp, Sep 2-5, 2013)
- These bounds allow us to manipulate the pseudo-inverse of Ũ and products of Ũ with other matrices.

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)

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):

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

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

Running time:

It suffices to set $s = O(d\varepsilon^{-1}\ \mathrm{polylog}(n/\varepsilon))$.

Overall running time is $O(nd\varepsilon^{-1}\ \mathrm{polylog}(n/\varepsilon))$.

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 $(1+\varepsilon)$ approximations for all leverage scores.

m-by-n matrices:

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

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

- **<u>Trivial:</u>** 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.

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

m-by-n matrices:

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

<u>Algorithm:</u>

- Approximate the top k left (or right) singular vectors of A.
- Use the approximations to estimate the leverage scores.

Overall running time is $r = O(ndk\varepsilon^{-1} \text{ polylog}(n/\varepsilon))$.

Applications of leverage scores

- Over (or under)-constrained Least Squares problems
- Feature selection and the CX factorization
- Solving systems of linear equations with Laplacian input matrices
- Element-wise sampling

Least-squares problems

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

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

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

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

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

Pseudoinverse of A

QR Decomposition:

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

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

Singular Value Decomposition:

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

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

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

Algorithm: Sampling for L_2 regression

(Drineas, Mahoney, Muthukrishnan SODA 2006,
Drineas, Mahoney, Muthukrishnan, & Sarlos NumMath2011)

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

$$\underbrace{\begin{pmatrix} \\ \\ A \\ \\ \\ \end{pmatrix}}_{n \times d,\quad n >> d} \begin{pmatrix} x_{opt} \end{pmatrix} \approx \underbrace{\begin{pmatrix} \\ \\ b \\ \\ \\ \end{pmatrix}}_{n \times 1}$$

__Algorithm__

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

Drineas, Mahoney, Muthukrishnan SODA 2006,
Drineas, Mahoney, Muthukrishnan, & Sarlos NumMath2011

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

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

Algorithm

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

$$\|b - Ax_{opt}\|_2^2 \leq \|b - A\tilde{x}_{opt}\|_2^2 \leq (1+\epsilon) \|b - Ax_{opt}\|_2^2$$

The sampling complexity (the value of r) is

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

(Hiding a loglog factor for simplicity; see Drineas et al. (2011) NumMath for a precise statement.)

Applications of leverage scores

- Over (or under)-constrained Least Squares problems
- Feature selection and the CX factorization
- Solving systems of linear equations with Laplacian input matrices
- Element-wise sampling

SVD decomposes a matrix as…

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

Top k left singular vectors

The SVD has strong optimality properties.

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

The CX decomposition

Mahoney & Drineas (2009) PNAS

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

Carefully chosen X

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

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

Why?

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!

CX decomposition

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

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).

The 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 $j=1\ldots n$.
- 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; see next slide)

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

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.

Leverage scores: human genetics data

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

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

SNPs

individuals

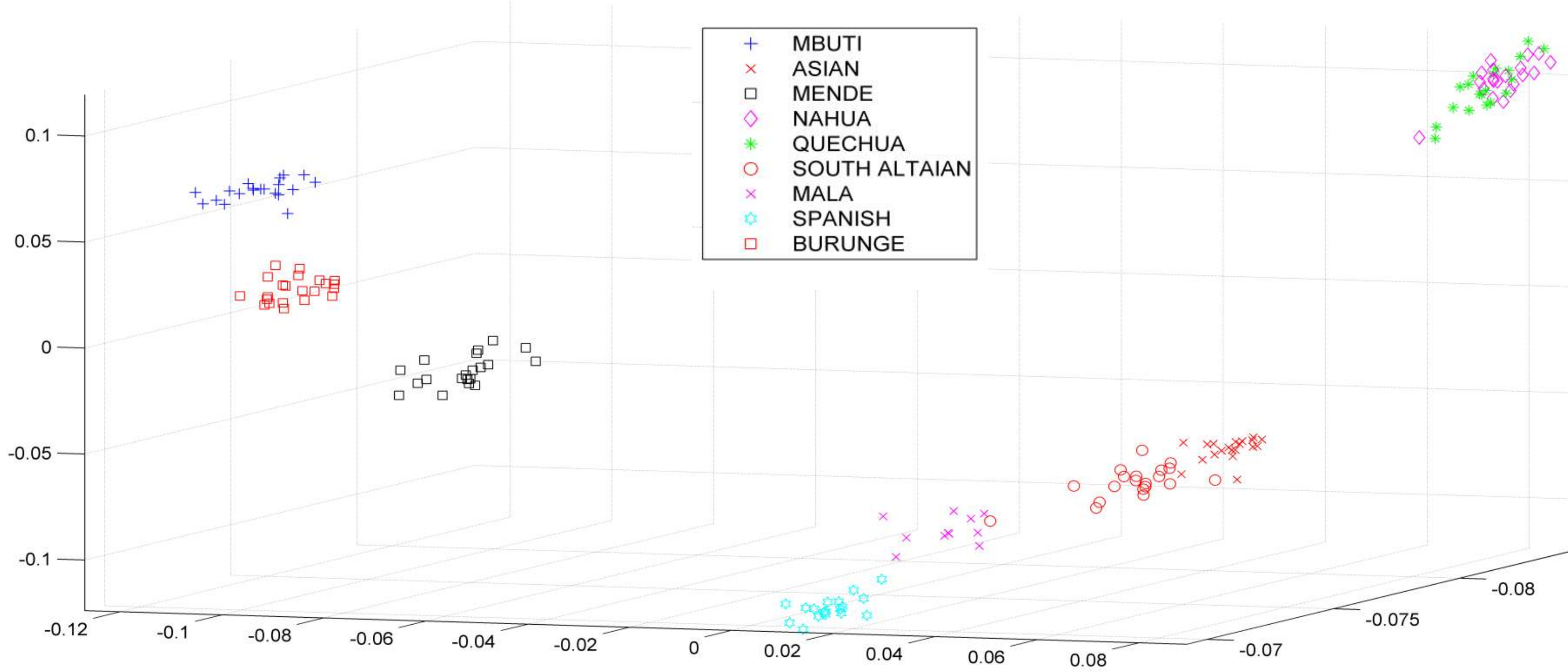
… AG CT GT GG CT CC CC CC CC AG AG AG AG AG AA CT AA GG GG CC GG AG CG AC CC AA CC AA GG TT AG CT CG CG CG AT CT CT AG CT AG GG GT GA AG …
… GG TT TT GG TT CC CC CC CC GG AA AG AG AG AA CT AA GG GG CC GG AA GG AA CC AA CC AA GG TT AA TT GG GG GG TT TT CC GG TT GG GG TT GG AA …
… GG TT TT GG TT CC CC CC CC GG AA AG AG AA AG CT AA GG GG CC AG AG CG AC CC AA CC AA GG TT AG CT CG CG CG AT CT CT AG CT AG GG GT GA AG …
… GG TT TT GG TT CC CC CC CC GG AA AG AG AG AA CC GG AA CC CC AG GG CC AC CC AA CG AA GG TT AG CT CG CG CG AT CT CT AG CT AG GT GT GA AG …
… GG TT TT GG TT CC CC CC CC GG AA GG GG GG AA CT AA GG GG CT GG AA CC AC CG AA CC AA GG TT GG CC CG CG CG AT CT CT AG CT AG GG TT GG AA …
… GG TT TT GG TT CC CC CG CC AG AG AG AG AG AA CT AA GG GG CT GG AG CC CC CG AA CC AA GT TT AG CT CG CG CG AT CT CT AG CT AG GG TT GG AA …
… GG TT TT GG TT CC CC CC CC GG AA AG AG AG AA TT AA GG GG CC AG AG CG AA CC AA CG AA GG TT AA TT GG GG GG TT TT CC GG TT GG GT TT GG AA …

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

Worldwide data

274 individuals, 9 populations, ~10,000 SNPs

Shriver et al. (2005) Hum Genom



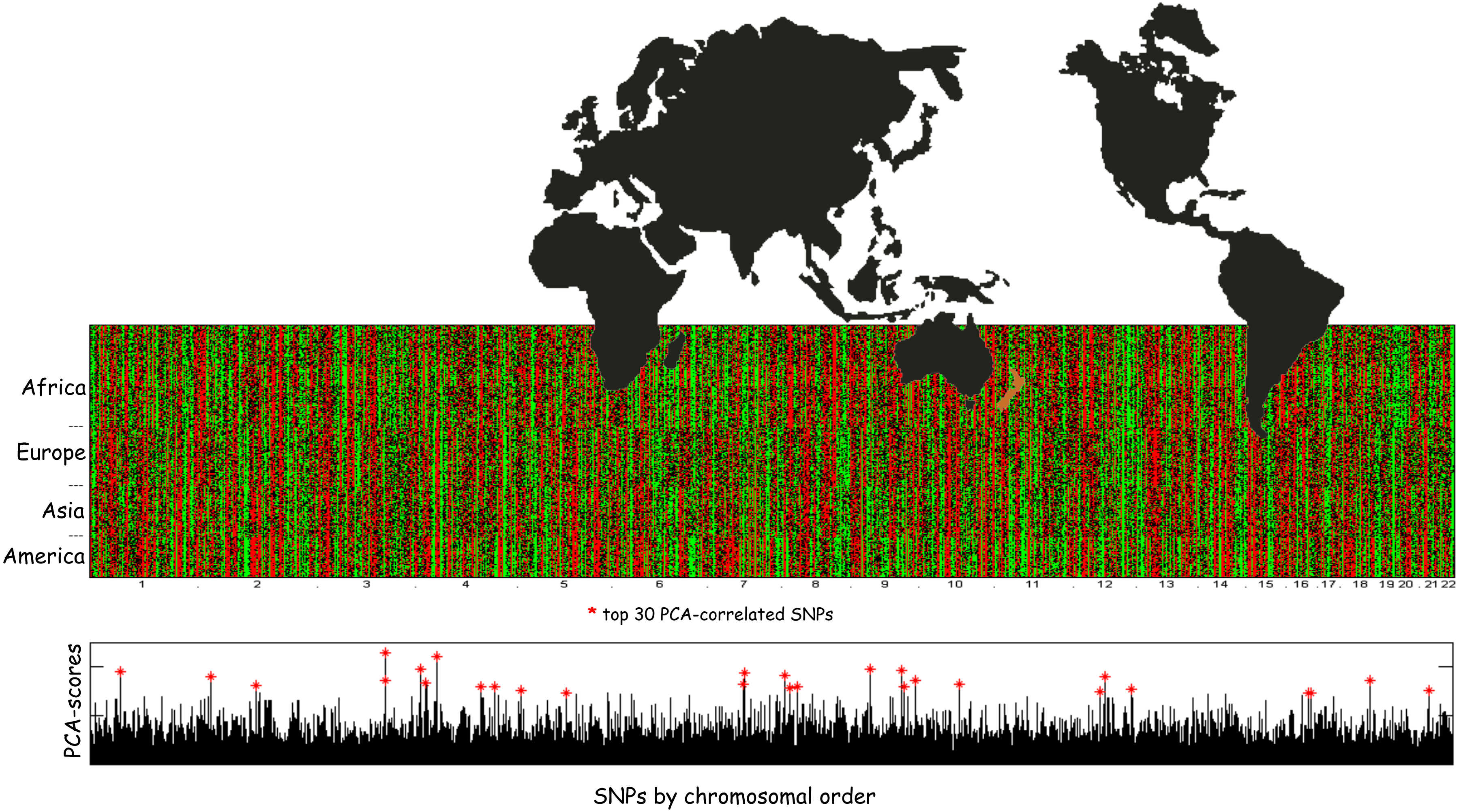
- PCA projection on the top three left singular vectors.
- Populations are clearly separated, BUT not altogether satisfactory:

    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?

Leverage scores of the columns of the 274-by-10,000 SNP matrix

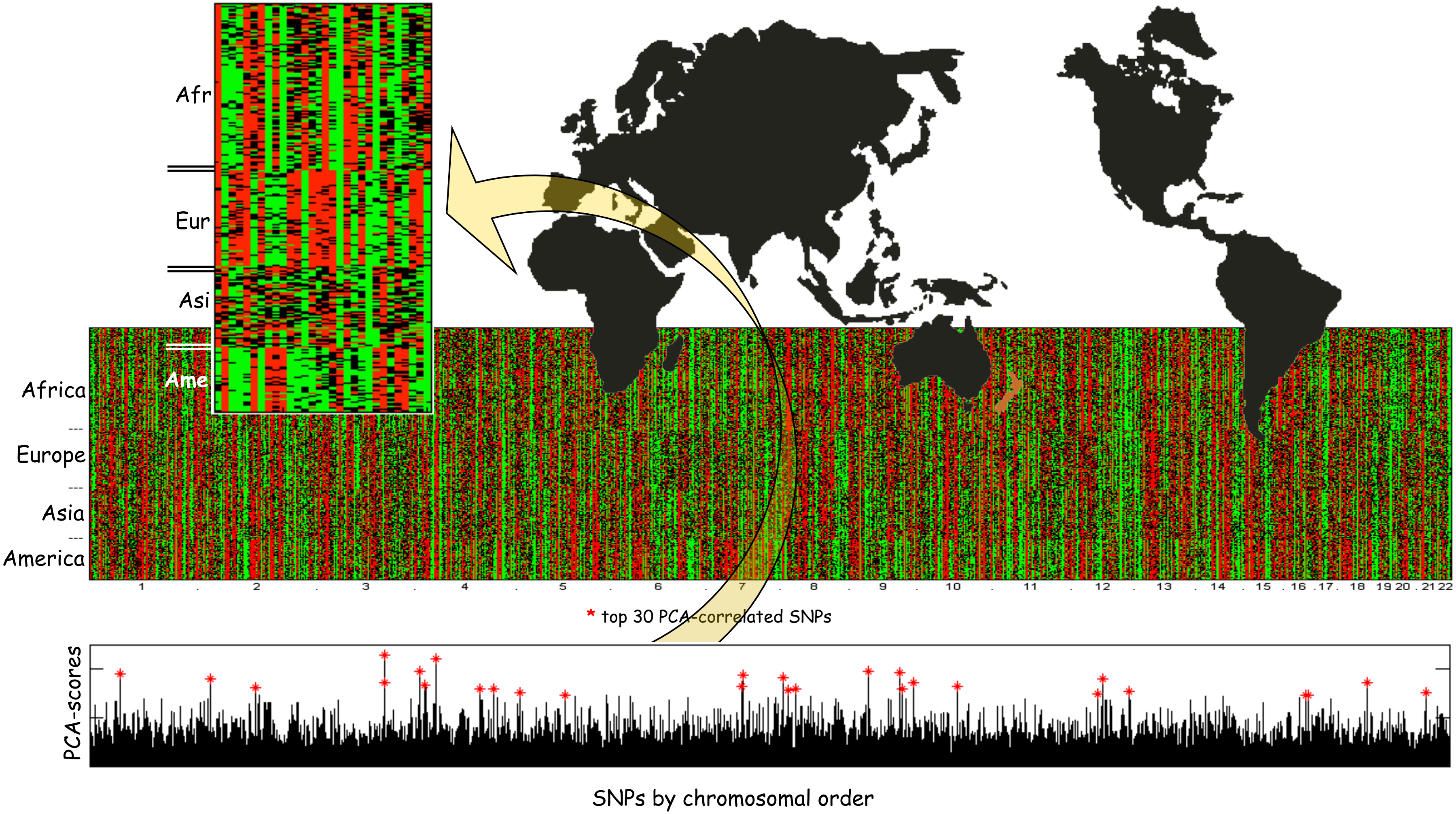


Paschou et al (2007; 2008) PLoS Genetics

Paschou et al (2010) J Med Genet

Drineas et al (2010) PLoS One

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



Paschou et al (2007; 2008) PLoS Genetics

Paschou et al (2010) J Med Genet

Drineas et al (2010) PLoS One

Applications of leverage scores

- Over (or under)-constrained Least Squares problems
- Feature selection and the CX factorization
- Solving systems of linear equations with Laplacian input matrices
- Element-wise sampling

Leverage scores & Laplacians

Consider a weighted (positive weights only!) 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:

$$L = \underbrace{\begin{pmatrix} \quad B^T \quad \end{pmatrix}}_{n \times m} \cdot \underbrace{\begin{pmatrix} \quad W \quad \end{pmatrix}}_{m \times m} \cdot \underbrace{\begin{pmatrix} \quad B \quad \end{pmatrix}}_{m \times n}$$

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:

$$L = \underbrace{\begin{pmatrix} & B^T & \end{pmatrix}}_{n \times m} \cdot \underbrace{\begin{pmatrix} \ddots & & 0 \\ & W & \\ 0 & & \ddots \end{pmatrix}}_{m \times m} \cdot \underbrace{\begin{pmatrix} \\ B \\ \\ \end{pmatrix}}_{m \times n}$$

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

(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.

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 to the effective resistances of the edges of G.

$$\underset{m \times m}{\begin{pmatrix} & & \\ & W^{1/2} & \\ & & \end{pmatrix}} \cdot \underset{m \times n}{\begin{pmatrix} & \\ B & \\ & \end{pmatrix}}$$

Edge-incidence matrix

Diagonal matrix
of edge weights

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 to the effective resistances of the edges of G.

$$\underbrace{\left(\quad W^{1/2} \quad \right)}_{m \times m} \cdot \underbrace{\left(\quad B \quad \right)}_{m \times n}$$

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.

Leverage scores & effective resistances

(Drineas & Mahoney ArXiv 2010)

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

Notation:
$x^T L x = ||x||_L$: energy norm
(as in the Spielman & Teng work)

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

Leverage scores & effective resistances

(Drineas & Mahoney ArXiv 2010)

Theorem: Let $\tilde{L}$ 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 $O(m \text{ polylog}(m))$ time.

Applications of leverage scores

- Over (or under)-constrained Least Squares problems
- Feature selection and the CX factorization
- Solving systems of linear equations with Laplacian input matrices
- Element-wise sampling

Element-wise sampling

(Drineas & Kundu 2013)

Element-wise sampling

- Introduced by Achlioptas and McSherry in STOC 2001.
- **Current state-of-the-art:** additive error bounds for arbitrary matrices and exact reconstruction under (very) restrictive assumptions using trace minimization.

(important breakthroughs by Candes, Recht, Tao, Wainright, and others)

The setup: let A be an m-by-n matrix of rank ρ, whose SVD is $A = U\Sigma V^T$.

- Sample r elements of A in r i.i.d. trials, where in each trial the (i,j)-th element of A is sampled with probability p_{ij}.
- Let Ω be the set of the r sampled indices and solve:

$$\min \|X\|_* \\ \text{s.t. } X_{ij} = A_{ij},\ \forall (i,j) \in \Omega$$

Element-wise sampling

(Drineas & Kundu 2013; similar result in Bhojanapalli et al. ArXiv 2013)

The setup: let A be an m-by-n matrix of rank ρ, whose SVD is $A = U\Sigma V^T$.

- Sample r elements of A in r i.i.d. trials, where in each trial the (i,j)-th element of A is sampled with probability p_{ij}.
- Let Ω be the set of the r sampled indices and solve: $\min \|X\|_* \quad \text{s.t. } X_{ij} = A_{ij}, \ \forall (i,j) \in \Omega$

Let $r = O\left((m+n)\,\rho^2\right)$

Let $$p_{ij} = \frac{1}{2}\frac{\|U_{(i)}\|_2^2 + \|V_{(j)}\|_2^2 - \|U_{(i)}\|_2^2\,\|V_{(j)}\|_2^2}{(m+n)\rho - \rho^2} + \frac{1}{4}\frac{(UV^T)_{ij}^2}{\rho} + \frac{1}{4}\frac{|(UV^T)_{ij}|}{\sum_{i,j=1}^{m,n}|(UV^T)_{ij}|}$$

Then, with constant probability, A can be recovered exactly.

Element-wise sampling

(Drineas & Kundu 2013; similar result in Bhojanapalli et al. ArXiv 2013)

<u>**The setup:**</u> let A be an m-by-n matrix of rank ρ, whose SVD is $A = U\Sigma V^T$.

- Sample r elements of A in r i.i.d. trials, where in each trial the (i,j)-th element of A is sampled with probability p_{ij}.
- Let Ω be the set of the r sampled indices and solve: $\min \|X\|_* \quad \text{s.t. } X_{ij} = A_{ij},\ \forall (i,j) \in \Omega$

Let $r = O\left((m+n)\,\rho^2\right)$

Let $$p_{ij} = \frac{1}{2}\frac{\|U_{(i)}\|_2^2 + \|V_{(j)}\|_2^2 - \|U_{(i)}\|_2^2\|V_{(j)}\|_2^2}{(m+n)\rho - \rho^2} + \frac{1}{4}\frac{(UV^T)_{ij}^2}{\rho} + \frac{1}{4}\frac{|(UV^T)_{ij}|}{\sum_{i,j=1}^{m,n}|(UV^T)_{ij}|}$$

Row leverage

Column leverage

Additional term

Then, with constant probability, A can be recovered exactly.

The proof uses a novel result on approximating the product of two linear operators by element-wise sampling and builds upon Recht (JMLR) 2011, Drineas & Zouzias (2011) IPL, and uses the idea of L_1 sampling from Achlioptas, Karnin, & Liberty ArXiv 2013.

Conclusions

- **Leverage scores:** a statistic on rows/columns of matrices that reveals the most influential rows/columns of a matrix.

- Can also be used for element-wise sampling!

- **Leverage scores:** equivalent to effective resistances.

- **Additional Fact:** Leverage scores can be "uniformized" by preprocessing the matrix via random projection-type matrices.
  (E.g., random sign matrices, Gaussian matrices, or Fast JL-type transforms.)