Fast Monte Carlo Algorithms for Matrices

Michael W. Mahoney

Department of Mathematics
Yale University
michael.mahoney@yale.edu
http://www.cs.yale.edu/homes/mmahoney

Joint work with:
Petros Drineas and Ravi Kannan

For more information, see the papers at:
http://www.cs.yale.edu/homes/mmahoney/matrix

Goal: *To develop and analyze fast Monte Carlo algorithms for performing useful computations on large matrices.*

- Matrix Multiplication
- Computation of the Singular Value Decomposition
- Computation of the CUR Decomposition
- Testing Feasibility of Linear Programs

Such matrix computations generally require time which is *superlinear* in the number of nonzero elements of the matrix, e.g., n^3 in practice.

These and related algorithms useful in applications where data sets are modeled by matrices and are extremely large.

Applications of these Algorithms

Matrices arise, e.g., since n *objects* (documents, genomes, images, web pages), each with m *features*, may be represented by a matrix $A \in \mathbb{R}^{m \times n}$.

- Covariance Matrices
- Latent Semantic Indexing
- DNA Microarray Data
- Eigenfaces and Image Recognition
- Similarity Query
- Matrix Reconstruction
- Numerous Linear Programming Applications
- Design of Approximation Algorithms for Classical CS NP-hard Optimization Problems

Linear Algebra Review

For $A \in \mathbb{R}^{m \times n}$ let $A^{(j)}$, $j = 1, \ldots, n$, denote the j-th column of A and $A_{(i)}$, $i = 1, \ldots, m$, denote the i-th row of A.

$\|A\|_2 = \sup_{x \in \mathbb{R}^n,\ x \neq 0} \frac{|Ax|}{|x|}$
$\|A\|_F = \left(\sum_{i=1}^m \sum_{j=1}^n A_{ij}^2 \right)^{1/2} = \left(\mathbf{Tr}\left(A^T A \right) \right)^{1/2}$
$\|A\|_2 \leq \|A\|_F \leq \sqrt{n} \, \|A\|_2$

Theorem. [SVD] *If $A \in \mathbb{R}^{m \times n}$, then there exist orthogonal matrices U and V and a matrix $\Sigma =$ **diag**$(\sigma_1, \ldots, \sigma_\rho)$, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_\rho \geq 0$, such that*

$$A = U \Sigma V^T = U_r \Sigma_r V_r^T = \sum_{t=1}^{r} \sigma_t u^t v^{t^T}.$$

$U = [u^1 u^2 \ldots u^m]$, $V = [v^1 v^2 \ldots v^n]$, and Σ constitute the Singular Value Decomposition (SVD) of A.

- σ_i are the singular values of A
- u^i, v^i are the i-th left and the i-th right singular vectors

Linear Algebra Review, Cont.

Recall that:

- $\begin{cases} Av^i = \sigma_i u^i \\ A^T u^i = \sigma_i v^i \end{cases}$

- $\begin{cases} \|A\|_2 = \sigma_1 \\ \|A\|_F^2 = \sum_{i=1}^r \sigma_i^2 \end{cases}$

Theorem. *Let* $A_k = U_k \Sigma_k V_k^T = \sum_{t=1}^k \sigma_t u^t v^{tT}$*:*

- $A_k = U_k U_k^T A = \left(\sum_{t=1}^k u^t u^{tT} \right) A$

- $A_k = A V_k V_k^T = A \left(\sum_{t=1}^k v^t v^{tT} \right)$

- $\|A - A_k\|_2 = \min_{D \in \mathbb{R}^{m \times n} : \textit{rank}(D) \le k} \|A - D\|_2$

- $\|A - A_k\|_F^2 = \min_{D \in \mathbb{R}^{m \times n} : \textit{rank}(D) \le k} \|A - D\|_F^2$

- $\max_{t : 1 \le t \le n} |\sigma_t(A + E) - \sigma_t(A)| \le \|E\|_2$

- $\sum_{k=1}^n (\sigma_k(A + E) - \sigma_k(A))^2 \le \|E\|_F^2$

Overview and Summary

- Pass-Efficient Model and Random Sampling
- Matrix Multiplication
- Singular Value Decomposition
- CUR Decomposition
- Lower Bounds
- Testing Feasibility of Linear Programs
- Approximating Max-Cut and Max-2-CSP Problems

Computation on Massive Data Sets

Data are *too large* to fit into main memory; they are either *not stored* or are *stored in external memory*.

Algorithms that compute on data streams examine the stream, keep a small "sketch" of the data, and perform computations on the sketch.

Algorithms are *randomized* and *approximate*.

Performance is evaluated by measures such as:

- the time to process an item
- the number of passes over the data
- the additional workspace and time
- the quality of the approximation returned

MP78: studied "the relation between the amount of internal storage available and the number of passes required to select the K-th highest of N inputs."

The Pass-Efficient Model

Amount of *disk space* has increased enormously; *RAM* and *computing speeds* have increased less rapidly.

We can *store* large amounts of data but we **cannot** *process* these data with traditional algorithms.

In the *Pass-Efficient Model*:

- Data are assumed to be *stored on disk*.
- The only access the algorithm has to the data is with a pass, where a *pass* is a sequential read of the entire input from disk where only a constant amount of processing time is permitted per bit read.
- An algorithm is allowed *additional RAM space* and *additional computation time*.

An algorithm is *pass-efficient* if it requires a small constant number of passes and sublinear additional time and space to compute a *description* of the solution.

If data are $A \in \mathbb{R}^{m \times n}$, then algorithms which require additional time and space that is $O(m+n)$ or $O(1)$ are pass-efficient.

Random Sampling

Typically, random sampling is used to estimate some parameter defined over a large set by looking at only a very small subset.

Uniform Sampling: every piece of data is equally likely to be picked.

- Advantages
  - "Coins" can be tossed "blindly."
  - Even if the number of data elements is not known in advance, can select one element u.a.r. in one pass over the data.
  - Much recent work on quantities that may be approximated with a small uniformly drawn sample.
- Disadvantages
  - Many quantities cannot be approximated well with a small random sample that is uniformly drawn.
  - E.g., compute the average of n numbers, where there is a lot of cancellation.

Random Sampling, Cont.

Nonuniform Sampling:

- Advantages
  - Can obtain much more generality and big gains, e.g., can approximately solve problems in sparse as well as dense matrices.
  - Smaller sampling complexity for similar bounds.
- Disadvantages
  - Must determine the nonuniform probabilities; multiple passes over the data usually needed.

Main conclusion of the work: A "*sketch*" consisting of a small number of *judiciously* chosen and randomly sampled **rows and columns** is sufficient for *provably* rapid and efficient *approximation* of a variety of common matrix operations.

Sampling Lemmas

SELECT Algorithm

Input: $\{a_1, \ldots, a_n\}$, $a_i \geq 0$, read in one pass, i.e., one sequential read, over the data.

Output: i^*, a_{i^*}.

- $D = 0$.
- For $i = 1$ to n,
  - $D = D + a_i$.
  - With probability a_i/D, let $i^* = i$ and $a_{i^*} = a_i$.
- Return i^*, a_{i^*}.

Lemma. [DKM] *Suppose that $\{a_1, \ldots, a_n\}$, $a_i \geq 0$, are read in one pass. Then the* SELECT *algorithm requires $O(1)$ additional storage space and returns i^* such that $\mathbf{Pr}\left[i^* = i\right] = a_i / \sum_{i'=1}^{n} a_{i'}$.*

The *sparse-unordered representation* of data is a form of data representation in which each element of the data stream consists of a pair $((i,j), A_{ij})$, where the elements in the data stream may be unordered with respect to the indices (i,j) and only the nonzero elements of the matrix A need to be presented.

Approximating Matrix Multiplication

For $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$, AB may be written as the sum of n rank-one matrices:

$$AB = \sum_{t=1}^{n} A^{(t)} B_{(t)}.$$

$$\left(\quad A \quad \right) \left(\quad B \quad \right) = \sum_{t=1}^{n} \left(A^{(t)} \right) \left(\quad B_{(t)} \quad \right).$$

BasicMatrixMultiplication (BMMA) Algorithm Summary.

- Given $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, $c \in \mathbb{Z}^+$, and $\{p_i\}_{i=1}^n$.
- Randomly sample c columns of A according to $\{p_i\}_{i=1}^n$ and rescale each column by $1/\sqrt{cp_{i_t}}$ to form $C \in \mathbb{R}^{m \times c}$.
- Sample the corresponding c rows of B and rescale each row by $1/\sqrt{cp_{i_t}}$ to form $R \in \mathbb{R}^{c \times p}$.
- Return $P = CR$.

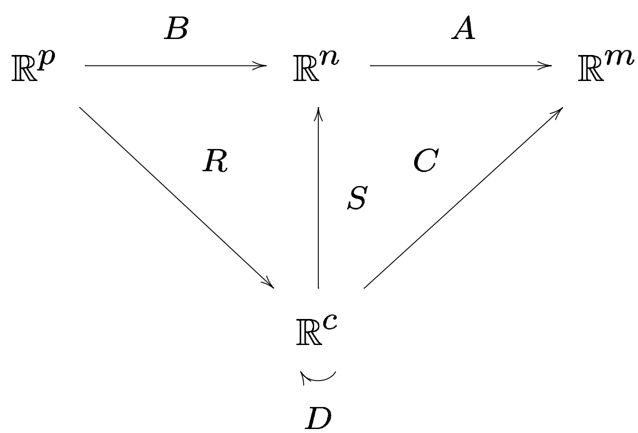
$$P = CR = \sum_{t=1}^{c} C^{(t)} R_{(t)} = \sum_{t=1}^{c} \frac{1}{cp_{i_t}} A^{(i_t)} B_{(i_t)}$$

BasicMatrixMultiplication Algorithm

Input: $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, $c \in \mathbb{Z}^+$ s.t. $1 \leq c \leq n$, and $\{p_i\}_{i=1}^n$ s.t. $p_i \geq 0$ and $\sum_{i=1}^n p_i = 1$.

Output: $P \in \mathbb{R}^{m \times p}$.

- For $t = 1$ to c,
  - Pick $i_t \in \{1, \ldots, n\}$ with $\mathbf{Pr}\left[i_t = k\right] = p_k$, $k = 1, \ldots, n$, independently and with replacement.
  - Set $C^{(t)} = A^{(i_t)}/\sqrt{cp_{i_t}}$ and $R_{(t)} = B_{(i_t)}/\sqrt{cp_{i_t}}$.
- Return $P = CR$.



$B : \mathbb{R}^p \to \mathbb{R}^n$
$A : \mathbb{R}^n \to \mathbb{R}^m$

Define the sampling matrix $S \in \mathbb{R}^{n \times c}$ as:

$$S_{ij} = \begin{cases} 1 & \text{if the } i\text{-th column of } A \text{ (and } i\text{-th row of } B\text{)} \\ & \text{ is chosen in the } j\text{-th trial} \\ 0 & \text{otherwise} \end{cases}$$

Define the rescaling matrix $D \in \mathbb{R}^{c \times c}$ as:

$$D_{tt'} = \begin{cases} 1/\sqrt{cp_{i_t}} & \text{if } t = t' \\ 0 & \text{otherwise} \end{cases}$$

Then $C = ASD$ and $R = (SD)^T B$ so that

$$P = CR = ASD(SD)^T B = \tilde{A}\tilde{B} \approx AB.$$

Advantages of the BMMA

- Conceptually simple and easily implementable.
- Algorithm and preprocessing do not need to be modified in the presence of negative entries.
- Nice interpretation if if A or B are low rank (or well approximated by low-rank matrices).
- Randomization is used only in the preprocessing step.
- Memory requirements are relatively small.
- Can use any algorithm for the smaller matrix multiplication.
- Does not tamper with the sparsity structure of the matrices.

Implementation of the BMMA

- Recall, $A : \mathbb{R}^n \to \mathbb{R}^m$ and $B : \mathbb{R}^p \to \mathbb{R}^n$.

- Uniform sampling: $O(1)$ space and time to sample and $O(m+p)$ space and time to construct C and R

- Nonuniform sampling: for nice probabilities one pass and $O(n)$ (or $O(1)$ if $B = A^T$) space and time to construct probabilities and a second pass and $O(m+p)$ space and time to construct C and R.

Def: A set of sampling probabilities $\{p_i\}_{i=1}^n$ are *nearly optimal probabilities* if $\exists$ a positive constant $\beta \leq 1$:

$$p_k \geq \frac{\beta \left|A^{(k)}\right| \left|B_{(k)}\right|}{\sum_{k'=1}^n \left|A^{(k')}\right| \left|B_{(k')}\right|}$$

Note: If $\beta = 1$ then $\mathbf{E}\left[\|AB - CR\|_F^2\right]$ is minimized.

Lemma. [DKM]

$$\begin{aligned}\mathbf{E}\left[(CR)_{ij}\right] &= (AB)_{ij} \\ \mathbf{Var}\left[(CR)_{ij}\right] &= \frac{1}{c}\sum_{k=1}^{n}\frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{c}(AB)_{ij}^2 \\ \mathbf{E}\left[\,\|AB-CR\|_F^2\right] &= \sum_{i=1}^{m}\sum_{j=1}^{p}\mathbf{Var}\left[(CR)_{ij}\right].\end{aligned}$$

Theorem. [DKM] *If* $\{p_i\}_{i=1}^n$ *are* nearly optimal probabilities *then*

$$\mathbf{E}\left[\,\|AB-CR\|_F\right] \le \frac{1}{\sqrt{\beta c}}\|A\|_F\,\|B\|_F\,.$$

Let $\delta \in (0,1)$ *and* $\eta = 1+\sqrt{(8/\beta)\log(1/\delta)}$*; then with probability at least* $1-\delta$*:*

$$\|AB-CR\|_F \le \frac{\eta}{\sqrt{\beta c}}\|A\|_F\,\|B\|_F\,.$$

Proof. Expectation straightforward; whp uses Doob martingales and Hoeffding-Azuma inequality. □

Corollary. [DKM] *If* $B = A^T$ *and* $\{p_i\}_{i=1}^n$ *are* nearly optimal probabilities, *i.e.*, $p_k \geq \frac{\beta \left| A^{(k)} \right|^2}{\|A\|_F^2}$, *then*

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

and with probability at least $1 - \delta$*:*

$$\left\| AA^T - CC^T \right\|_F \leq \frac{\eta}{\sqrt{\beta c}} \|A\|_F^2 .$$

$$\begin{pmatrix} & & \\ & A & \\ & & \end{pmatrix} \begin{pmatrix} & & \\ & A^T & \\ & & \end{pmatrix} = \begin{pmatrix} & & \\ & C & \\ & & \end{pmatrix} \begin{pmatrix} & & \\ & C^T & \\ & & \end{pmatrix}$$

Sampling with non-nearly optimal probabilities.

[t]

	$\mathbf{E}\left[\,\|AB-CR\|_F\right] \leq$	w.h.p. $\|AB-CR\|_F \leq$	comments and restrictions
$p_k \geq \frac{\beta\left\|A^{(k)}\right\|\left\|B_{(k)}\right\|}{\sum_{k'}\left\|A^{(k')}\right\|\left\|B_{(k')}\right\|}$	$\frac{1}{\sqrt{\beta c}}\|A\|_F\ \|B\|_F$	$\frac{\eta}{\sqrt{\beta c}}\|A\|_F\ \|B\|_F$	$\eta = 1+\sqrt{\frac{8}{\beta}\log\left(\frac{1}{\delta}\right)}$
$p_k \geq \frac{\beta\left\|A^{(k)}\right\|^2}{\|A\|_F^2}$	$\frac{1}{\sqrt{\beta c}}\|A\|_F\ \|B\|_F$	$\frac{\eta}{\sqrt{\beta c}}\|A\|_F\ \|B\|_F$	$\eta = 1+\frac{\|A\|_F}{\|B\|_F}\mathcal{M}\sqrt{\frac{8}{\beta}\log (}$ $\mathcal{M} = \max_\alpha \frac{\left\|B_{(\alpha)}\right\|}{\left\|A^{(\alpha)}\right\|}$
$p_k \geq \frac{\beta\left\|B_{(k)}\right\|^2}{\|B\|_F^2}$	$\frac{1}{\sqrt{\beta c}}\|A\|_F\ \|B\|_F$	$\frac{\eta}{\sqrt{\beta c}}\|A\|_F\ \|B\|_F$	$\eta = 1+\frac{\|B\|_F}{\|A\|_F}\mathcal{M}\sqrt{\frac{8}{\beta}\log (}$ $\mathcal{M} = \max_\alpha \frac{\left\|A^{(\alpha)}\right\|}{\left\|B_{(\alpha)}\right\|}$
$p_k \geq \frac{\beta\left\|A^{(k)}\right\|}{\sum_{k'=1}^{n}\left\|A^{(k')}\right\|}$	$\frac{1}{\sqrt{\beta c}}\|A\|_F\ \sqrt{n}\mathcal{M}$	$\frac{\eta}{\sqrt{\beta c}}\|A\|_F\ \sqrt{n}\mathcal{M}$	$\eta = 1+\sqrt{\frac{8}{\beta}\log\left(\frac{1}{\delta}\right)}$ $\mathcal{M} = \max_\alpha \left\|B_{(\alpha)}\right\|$
$p_k \geq \frac{\beta\left\|B_{(k)}\right\|}{\sum_{k'=1}^{n}\left\|B_{(k')}\right\|}$	$\frac{1}{\sqrt{\beta c}}\sqrt{n}\mathcal{M}\ \|B\|_F$	$\frac{\eta}{\sqrt{\beta c}}\sqrt{n}\mathcal{M}\ \|B\|_F$	$\eta = 1+\sqrt{\frac{8}{\beta}\log\left(\frac{1}{\delta}\right)}$ $\mathcal{M} = \max_\alpha \left\|A^{(\alpha)}\right\|$
$p_k \geq \frac{\beta\left\|A^{(k)}\right\|\left\|B_{(k)}\right\|}{\|A\|_F\|B\|_F}$	$\frac{1}{\sqrt{\beta c}}\|A\|_F\ \|B\|_F$	$\frac{\eta}{\sqrt{\beta c}}\|A\|_F\ \|B\|_F$	$\eta = 1+\sqrt{\frac{8}{\beta}\log\left(\frac{1}{\delta}\right)}$
$p_k = \frac{1}{n}$	See Lemma XX.	See Lemma XX.	See Lemma XX.

Element-wise Error Bounds for BMMA

Lemma. [DKM] *Let M be such that $|A_{ij}| \leq M$ and $|B_{ij}| \leq M$. Construct an approximation $P = CR$ to AB with the* $\textsc{BasicMatrixMultiplication}$ *algorithm. If $p_k = 1/n$ then with probability greater than $1 - \delta$ $\forall i, j$*

$$|(AB)_{ij} - (CR)_{ij}| < \frac{nM^2}{\sqrt{c}}\sqrt{8\ln(2mp/\delta)}$$

If $\{p_k\}_{k=1}^n$ are nearly optimal probabilities *then with probability greater than $1 - \delta$ $\forall i, j$*

$$|(AB)_{ij} - (CR)_{ij}| < \frac{n\sqrt{mp}M^2}{\sqrt{\beta c}}\sqrt{(8/\beta)\ln(2mp/\delta)}$$

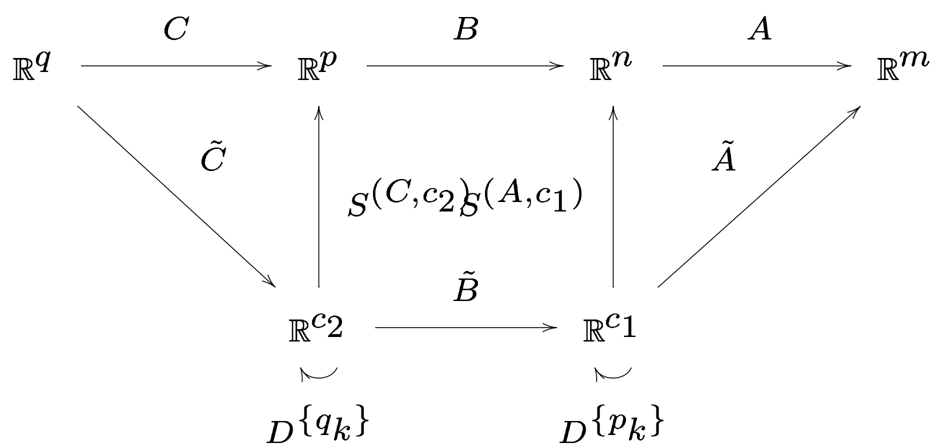
Note: Nearly optimal probabilities are worse by a factor of $\sqrt{mp/\beta}$ since they are nearly optimal with respect to minimizing $\mathbf{E}\left[\,\|AB - CR\|_F^2\right]$.

Multiplying more than two matrices

Given matrices $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{p\times q}$,

$$ABC = \sum_{s=1}^{n}\sum_{t=1}^{p} A^{(s)} B_{st} C_{(t)}.$$

First algorithm: Randomly choose c_1 times $i_s \in \{1,\ldots,n\}$ according to $\{p_i\}_{i=1}^{n}$ and and choose c_2 times $j_t \in \{1,\ldots,p\}$ according to $\{q_j\}_{j=1}^{p}$. Then form the matrices $\tilde{A} \in \mathbb{R}^{m\times c_1}$, $\tilde{B} \in \mathbb{R}^{c_1\times c_2}$, and $\tilde{C} \in \mathbb{R}^{c_2\times q}$ so that $\tilde{A}\tilde{B}\tilde{C} = \sum_{s=1}^{c_1}\sum_{t=1}^{c_2} \frac{A^{(i_s)} B_{i_s j_t} C_{(j_t)}}{c_1 c_2 p_{i_s} q_{j_t}}$.



Note: Analysis is difficult due to non-independence in sampling.

Second algorithm: Randomly choose c times $(i_s, j_t) \in \{1, \ldots, n\} \times \{1, \ldots, p\}$ according to $\{p_{kl}\}_{kl=1}^{np}$ and define

$$P = \sum_{(s,t)=1}^{c} \frac{1}{c p_{k_s l_t}} A^{(k_s)} B_{k_s l_t} C_{(l_t)}.$$

Lemma. [DKM]

$$\begin{aligned} \mathbf{E}\left[(P)_{ij}\right] &= (ABC)_{ij} \\ \mathbf{Var}\left[(P)_{ij}\right] &= \frac{1}{c}\sum_{k=1}^{n}\sum_{l=1}^{p}\frac{1}{p_{kl}}A_{ik}^2 B_{kl}^2 C_{lj}^2 - \frac{1}{c}\left(ABC\right)_{ij}^2. \end{aligned}$$

If

$$p_{kl} \geq \beta \frac{\left|A^{(k)}\right| \left|B_{kl}\right| \left|C_{(l)}\right|}{\sum_{k'} \sum_{l'} \left|A^{(k')}\right| \left|B_{k'l'}\right| \left|C_{(l')}\right|}$$

for some $\beta \leq 1$*, then*

$$\mathbf{E}\left[\|ABC - P\|_F^2\right] \leq \frac{1}{c\beta}\sum_{k}\sum_{l}\left|A^{(k)}\right| \left|B_{kl}\right| \left|C_{(l)}\right|.$$

and a similar result can be shown to hold with high probability.

Note: Difficult in general to compute optimal probabilities.

Second Matrix Multipliation Algorithm

AlternateMatrixMultiplication Algorithm

Input: $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, $c \in \mathbb{Z}^{+}$ such that $1 \leq c \leq n$, and $\left\{p_k^{(ij)}\right\}_{i,j,k=1}^{m,p,n}$ such that $p_k^{(ij)} \geq 0$ and $\sum_{k=1}^{n} p_k^{(ij)} = 1$, for all i, j.

Output: $P \in \mathbb{R}^{m \times p}$.

Algorithm:

- For $i = 1$ to m and $j = 1$ to p,
  - For $t = 1$ to c,
    - Pick $i_t \in \{1, \ldots, n\}$ with $\mathbf{Pr}\left[i_t = k\right] = p_k^{(ij)}$, $k = 1, \ldots, n$, independently and with replacement.
    - Set $P_t^{ij} = \frac{A_{i i_t} B_{i_t j}}{c p_{i_t}^{(ij)}}$.
  - Set $P_{ij} = \sum_{t=1}^{c} P_t^{ij}$.
- Return $P = \left(P_{ij}\right)$.

Notes:

- Recall: $(AB)_{ij} = \sum_{t=1}^{n} A_{it} B_{tj}$.
- Approximate the product AB by estimating each of its elements independently by randomly sampling from terms in this sum.
- Improved bound with respect to the spectral norm.
- Cannot be implemented unless either a large number of passes are performed or both matrices A and B are stored in RAM.

Lemma. [DKM] *Construct an approximation P to AB with the* AlternateMatrixMultiplication *algorithm. If $p_k^{(ij)} = 1/n$ then*

$$\mathbf{E}\left[\,\|AB - P\|_F^2\right] \leq \frac{n}{c}\sum_{k=1}^{n} \left|A^{(k)}\right|^2 \left|B_{(k)}\right|^2.$$

If $p_k^{(ij)} = A_{ik}^2 / \left|A_{(i)}\right|^2$, then

$$\mathbf{E}\left[\,\|AB - P\|_F^2\right] \leq \frac{1}{c}\,\|A\|_F^2\ \|B\|_F^2.$$

Theorem. [DKM] *Let $m+p \geq 24$, $M = \max_{ij}\{A_{ij}, B_{ij}\}$, and $c \leq \frac{m+p}{4\ln^6(m+p)}$. Construct an approximation P to AB with the* AlternateMatrixMultiplication *algorithm. Then, with probability at least $1 - 1/(m+p)$*

$$\|AB - P\|_2 \leq \frac{10}{\sqrt{c}} n M^2 \sqrt{m+p}.$$

Proof. Makes use of concentration results of the eigenvalues of random matrices from AM03, FK81, KV00. □

Third Matrix Multipliation Algorithm

ElementwiseMatrixMultiplication Algorithm

Input: $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, $\{p_{ij}\}_{i,j=1}^{m,n}$ such that $0 \leq p_{ij} \leq 1$, and $\{q_{ij}\}_{i,j=1}^{n,p}$ such that $0 \leq q_{ij} \leq 1$.

Output: $P \in \mathbb{R}^{m \times p}$.

Algorithm:

- For $i = 1$ to m and $j = 1$ to n,
  - Set
$$S_{ij} = \begin{cases} A_{ij}/p_{ij} & \text{with probability } p_{ij} \\ 0 & \text{otherwise.} \end{cases}$$
- For $i = 1$ to n and $j = 1$ to p,
  - Set
$$R_{ij} = \begin{cases} B_{ij}/q_{ij} & \text{with probability } q_{ij} \\ 0 & \text{otherwise.} \end{cases}$$
- Return $P = SR$.

Notes:

- Nonuniformly sample elements rather than rows and columns.
- Based on idead from AM01 and AM03.
- The algorithm does not keep “corresponding” elements.
- Implementable in two passes.
- We get an expected number of elements and so an expected running time.

Lemma. [DKM] *Let* $p_{ij} = \min\{1, \ell A_{ij}^2 / \|A\|_F^2\}$ *and* $q_{ij} = \min\{1, \ell' B_{ij}^2 / \|B\|_F^2\}$. *Construct an approximation* $P = SR$ *to* AB *with the* ElementwiseMatrixMultiplication *algorithm.*

$$\mathbf{E}\left[\|AB - P\|_F^2 \right] \geq \frac{mpn}{\ell\ell'} \|A\|_F^2 \, \|B\|_F^2 - \sum_{k=1}^{n} \left|A^{(k)}\right|^2 \left|B_{(k)}\right|^2 .$$

Theorem. [DKM] *Let*

$$p_{ij} = \begin{cases} \min\{1, \ell A_{ij}^2 / \|A\|_F^2\} & \text{if } |A_{ij}| > \frac{O(1)\|A\|_F \log^3 n}{\sqrt{n\ell}} \\ \min\{1, \frac{\sqrt{\ell}|A_{ij}| \log^3 n}{\sqrt{n}\|A\|_F}\} & \text{otherwise} \end{cases}$$

Let $\ell = \ell' \leq \|X\|_F^2 / \max_{i,j} X_{ij}^2$ *for* $X = A, B$ *and let* $m = n = p$ *and* $n \geq \log^6 n$. *Construct an approximation* $P = SR$ *to* AB *with the* ElementwiseMatrixMultiplication *algorithm. Then, with probability at least* $1 - 1/n$,

$$\|AB - P\|_2 \leq \left(\sqrt{\frac{15^2 n}{\ell}} + \frac{50n}{\ell} \right) \|A\|_F \, \|B\|_F$$

Proof. Makes use of concentration results of the eigenvalues of random matrices from AM03, which in turn is based on FK81, KV00. □

Summary of Matrix Multiplication

- Three algorithms to compute an approximation P to the product AB.
- Provable bounds on the error matrix $P - AB$ and run in $O(mp + mn + np)$ time.
- For the $\textsc{BasicMatrixMultiplication}$ algorithm, $c = O(1)$ columns of A are randomly chosen and rescaled to form a matrix C, the corresponding c rows of B are used to form R, and $P = CR$.
- The probability distribution $\{p_i\}_{i=1}^{n}$ over column (of A) and row (of B) pairs and the rescaling are both crucial features; if chosen judiciously:

$$\|AB - P\|_F \leq O(1/\sqrt{c})\,\|A\|_F\;\|B\|_F$$

- Implementable without storing A and B in RAM, provided two passes over the matrices $O(m + p)$ additional RAM memory.

Approximating the SVD of a Matrix

Goal: Given a matrix $A \in \mathbb{R}^{m \times n}$ we wish to approximate its top k singular values and the corresponding singular vectors in a constant number of passes through the data and additional space and time that is either $O(m+n)$ or $O(1)$, independent of m and n.

$\textsc{LinearTimeSVD}$ Algorithm Summary. (DFKVV99)

- Given $A \in \mathbb{R}^{m \times n}$, $c, k \in \mathbb{Z}^+$, and $\{p_i\}_{i=1}^n$.
- Randomly sample c columns of A according to $\{p_i\}_{i=1}^n$ and rescale each column by $1/\sqrt{cp_{i_t}}$ to form $C \in \mathbb{R}^{m \times c}$.
- Compute $C^T C \in \mathbb{R}^{c \times c}$ (recall $CC^T \approx AA^T$) and its SVD; the singular vectors of $C^T C$ are right singular vectors of C.
- Compute $H_k (= U_C)$, the top k left singular vectors of C and approximations to the left singular vectors of A.

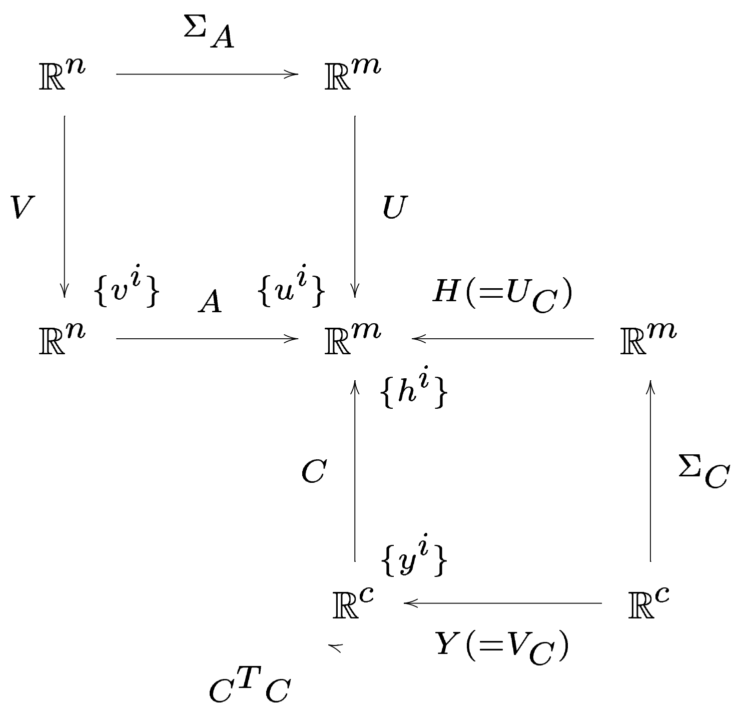
Note: Sampling probabilities p_k must be chosen carefully; assume they are nearly optimal.

LinearTimeSVD Algorithm

Input: $A \in \mathbb{R}^{m \times n}$, $c, k \in \mathbb{Z}^+$ s.t. $1 \leq k \leq c \leq n$, $\{p_i\}_{i=1}^n$ s.t. $p_i \geq 0$ and $\sum_{i=1}^n p_i = 1$.

Output: $H_k \in \mathbb{R}^{m \times k}$, $\{\lambda_i\}_{i=1}^k$ s.t. $\lambda_i \in \mathbb{R}^+$.

- For $t = 1$ to c,
  - Pick $i_t \in 1, \ldots, n$ with $\mathbf{Pr}[i_t = k] = p_k$, $k = 1, \ldots, n$.
  - Set $C^{(t)} = A^{(i_t)}/\sqrt{cp_{i_t}}$.
- Compute $C^T C$ and its singular value decomposition; say $C^T C = \sum_{t=1}^c \sigma_t^2(C) y^t {y^t}^T$.
- Compute $h^t = Cy^t/\sigma_t(C)$ for $t = 1, \ldots, k$.
- Return H_k, where $H_k^{(t)} = h^t$, and $\{\lambda_t\}_{t=1}^k$, where $\lambda_t = \sigma_t(C)$.



Implementation of Linear (and Constant) Time Approximate SVD Algorithms

- Can calculate nearly optimal p_k in one pass and $O(c)$ additional space and time

- C can then be constructed in one more pass and $O(mc)$ additional space and time
  - C not constructed; in second pass compute nearly optimal q_i in $O(w)$ space and time; construct W in third pass with $O(cw)$ space and time

- Computing C^TC requires $O(mc^2)$ additional space and time.
  - Computing W^TW requires $O(cw^2)$ additional space and time.

- Computing the SVD of C^TC requires $O(c^3)$ additional space and time.

- Computing H_k requires $O(mck)$ additional space and time for k matrix-vector multiplications.
  - $\tilde{H}_l$ not explicitly constructed.

- Since c, k are constant, overall $O(m+n)$.
  - Overall, $O(1)$.

The LinearTimeSVD Algorithm, Cont.

Theorem. [DKM] *Construct* H_k *with the* LinearTimeSVD *algorithm by sampling* c *columns of* A *with probabilities* $\{p_i\}_{i=1}^n$. *Then:*

$$\left\|A - H_k H_k^T A\right\|_F^2 \leq \|A - A_k\|_F^2 + 2\sqrt{k}\left\|AA^T - CC^T\right\|_F .$$

Proof. • $\left\|A - H_k H_k^T A\right\|_F^2 = \|A\|_F^2 - \sum_{t=1}^k \left|A^T h^t\right|^2$

• $\left|\sum_{t=1}^k \left|A^T h^t\right|^2 - \sigma_t^2(C)\right| \leq \sqrt{k}\left\|AA^T - CC^T\right\|_F$

• $\left|\sum_{t=1}^k \sigma_t^2(C) - \sigma_t^2(A)\right| \leq \sqrt{k}\left\|CC^T - AA^T\right\|_F$ □

Theorem. [DKM] *Construct* H_k *with the* LinearTimeSVD *algorithm by sampling* c *columns of* A *with probabilities* $\{p_i\}_{i=1}^n$. *Then:*

$$\left\|A - H_k H_k^T A\right\|_2^2 \leq \|A - A_k\|_2^2 + 2\left\|AA^T - CC^T\right\|_2 .$$

Proof. • $\left\|A - H_k H_k^T A\right\|_2 \leq \max_{z \in \mathcal{H}_{m-k}, |z|=1} \left|z^T A\right|$

• $\left|z^T A\right|^2 \leq 2\left\|AA^T - CC^T\right\|_2 + \sigma_{k+1}^2(A)$ for $z \in \mathcal{H}_{m-k}$, $|z| = 1$ □

The LinearTimeSVD Algorithm, Cont.

Theorem. [DFKVV99,DKM] *Construct H_k with the* LinearTimeSVD *algorithm by sampling c columns of A with nearly optimal probabilities and let $\eta = 1+\sqrt{(8/\beta)\log(1/\delta)}$. Let $\epsilon > 0$. If $c = \Omega(k\eta^2/\epsilon^4)$, then*

$$\left\|A - H_k H_k^T A\right\|_F \leq \|A - A_k\|_F + \epsilon \|A\|_F$$

in expectation and with high probability. In addition, if $c = \Omega(\eta^2/\epsilon^4)$, then

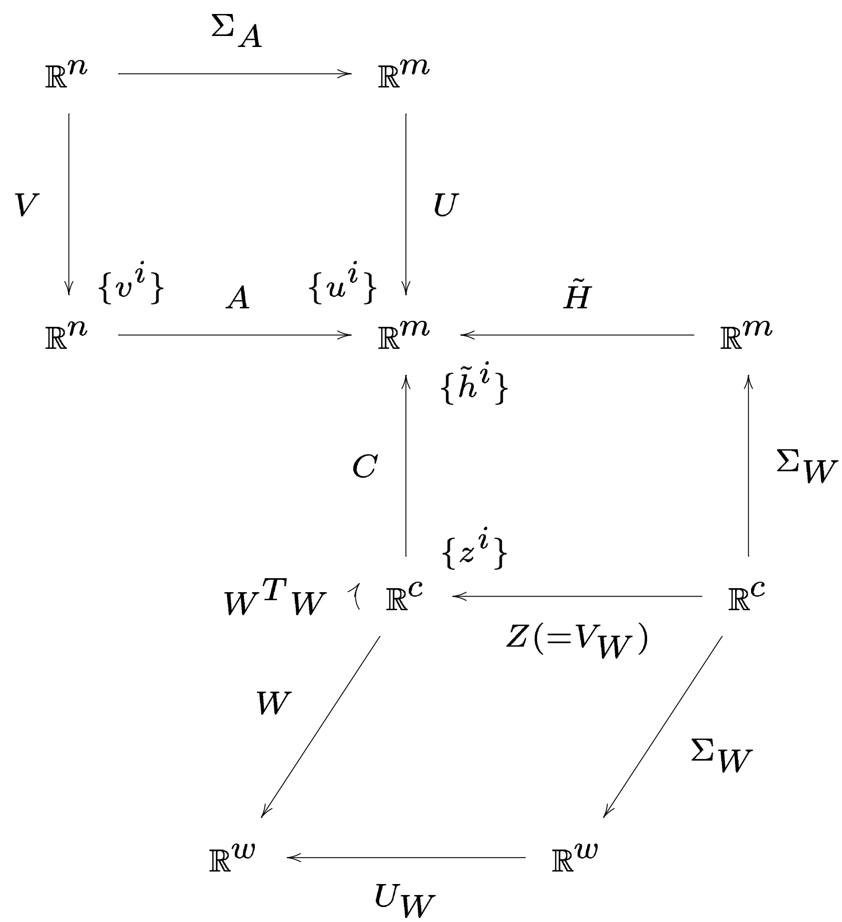
$$\left\|A - H_k H_k^T A\right\|_2 \leq \|A - A_k\|_2 + \epsilon \|A\|_F$$

in expectation and with high probability.

Proof. Combine $\|\cdot\|_F^2$ and $\|\cdot\|_2^2$ results with bound on $\left\|AA^T - CC^T\right\|_F$ from approximate matrix multiplication algorithm. $\square$

ConstantTimeSVD Algorithm Summary.

- Randomly sample c columns of A according to $\{p_i\}_{i=1}^n$ and sample w rows of C with nearly optimal probabilities and rescale to form $W \in \mathbb{R}^{w \times c}$.
- Compute $W^T W \in \mathbb{R}^{c \times c}$ and its SVD; the singular vectors of $W^T W$ are approximations to the singular vectors of $C^T C$ and thus to the right singular vectors of C.



The ConstantTimeSVD Algorithm, Cont.

ConstantTimeSVD Algorithm

Input: $A \in \mathbb{R}^{m \times n}$, $c, w, k \in \mathbb{Z}^{+}$ s.t. $1 \leq w \leq m$, $1 \leq c \leq n$, and $1 \leq k \leq min(w, c)$, and $\{p_i\}_{i=1}^{n}$ s.t. $p_i \geq 0$ and $\sum_{i=1}^{n} p_i = 1$.

Output: $\{\lambda_i\}_{i=1}^{l}$ s.t. $\lambda_i \in \mathbb{R}^{+}$ and a "description" of $\tilde{H}_l \in \mathbb{R}^{m \times l}$.

- For $t = 1$ to c,
  - Pick $i_t \in 1, \ldots, n$ with $\mathbf{Pr}\left[i_t = k\right] = p_k$, $k = 1, \ldots, n$.
  - Set $C^{(t)} = A^{(i_t)} / \sqrt{c p_{i_t}}$.
- Choose $\left\{q_j\right\}_{j=1}^{m}$ s.t. $q_j \geq 0$, $\sum_{j=1}^{m} q_j = 1$, and $q_j \geq \beta' \left|C_{(j)}\right|^2 / \|C\|_F^2$ with $\beta' < 1$.
- For $t = 1$ to w,
  - Pick $j_t \in 1, \ldots, m$ with $\mathbf{Pr}\left[j_t = j\right] = q_j$, $j = 1, \ldots, m$.
  - Set $W_{(t)} = C_{(j_t)} / \sqrt{w q_{j_t}}$.
- Compute $W^T W$ and its singular value decomposition. Say $W^T W = \sum_{t=1}^{c} \sigma_t^2(W) z^t z^{t^T}$.
- If a $\|\cdot\|_F$ bound is desired
  - Set $\gamma = \Theta(\epsilon^2/k)$,

  Else if a $\|\cdot\|_2$ bound is desired
  - Set $\gamma = \Theta(\epsilon^2)$.
- Let l be the index of the smallest singular value such that $\sigma_t^2(W) \geq \gamma \|W\|_F^2$.
- Return $min\{l, k\}$ singular values $\sigma_t(W)$ and their corresponding singular vectors $\left\{z^t\right\}_{t=1}^{l}$.
- (If an explicit solution is to be computed then) compute $\tilde{h}^t = C z^t / |\sigma_t(W)|$ for $t = 1, \ldots, l$.
- (If an explicit solution is to be computed then) return $\tilde{H}_l$, where $\tilde{H}_l^{(i)} = \tilde{h}^i$, and $\{\lambda_i\}_{i=1}^{l}$, where $\lambda_i = \sigma_i(W)$.

The ConstantTimeSVD Algorithm, Cont.

Theorem. [DKM] *With the* ConstantTimeSVD *algorithm construct* $\tilde{H}_l$ *by sampling* c *columns of* A *and then* w *rows of* C *with nearly optimal probabilities. Let* $\epsilon > 0$. *If* $\gamma = \Theta(\epsilon^2/k)$, $c = \Omega(k^3/\epsilon^8)$, *and* $w = \Omega(k^3/\epsilon^8)$ *then with probability at least* $1 - \delta$

$$\left\| A - \tilde{H}_l \tilde{H}_l^T A \right\|_F \leq \| A - A_k \|_F + \epsilon \| A \|_F .$$

If $\gamma = \Theta(\epsilon^2)$, $c = \Omega(1/\epsilon^4)$, *and* $w = \Omega(1/\epsilon^6)$ *then with probability at least* $1 - \delta$

$$\left\| A - \tilde{H}_l \tilde{H}_l^T A \right\|_2 \leq \| A - A_k \|_2 + \epsilon \| A \|_F .$$

Proof. Similar to that for the LinearTimeSVD algorithm; more complicated due to second level of sampling since, e.g., $\tilde{H}_l$ is not orthogonal. □

Summary of the SVD Results

In order to compute a matrix D (e.g., $H_k H_k^T A$) s.t.

$$\|A - D\|_\xi \leq \|A - A_k\|_\xi + \epsilon \|A\|_F \,,$$

for $\xi = 2, F$, we need a sampling complexity of:

	LinearTimeSVD	ConstantTimeSVD
$\|A - D^*\|_2$	$1/\epsilon^4$	$1/\epsilon^6$
$\|A - D^*\|_F$	k/ϵ^4	k^3/ϵ^8

- FKV98 did original work on low rank approximations.
  - Worked with randomly-chosen and constant-sized submatrix to compute low rank approximations to a matrix.
  - Sampled k^4/ϵ^6 rows and columns.
  - Construction of submatrix required sampling probabilities and thus $O(m + n)$ additional space and time.
- Linear time results from DFKVV99.
- AM01 do elementwise sampling to discretize and/or zero out elements; motivation is to accelerate orthogonal iteration and Lanczos iteration.
- DKM04
  - Improvement for constant-time $\|\cdot\|_F$ bound; k^3/ϵ^8 suffice.
  - Improvement for constant-time $\|\cdot\|_2$ bound; $1/\epsilon^6$ suffice.
  - Construct sample *and* compute in constant additional space and time.

Summary of SVD Results of AM01

- Achlioptas and McSherry.
- Goal: speed up computation of low-rank approximations by reducing the number of nonzero elements and/or their description length.
- Idea: independently sample and/or quantize the entries of A.
- Sampling and quantization: adding a random matrix N to A, whose entries are independent random variables with zero-mean and bounded variance.
- Obtain bounds of the form:

$$\begin{aligned}\left\|A - D^*\right\|_2 &\leq \|A - A_k\|_2 + O(1)\sqrt{n/p} \\ \left\|A - D^*\right\|_F &\leq \|A - A_k\|_F + O(1)\,\|A_k\|_F^{1/2}\,(kn/p)^{1/4}\end{aligned}$$

- Proofs use bounds on the eigenvalues of random matrices from FK81 and AKV02.

Summary of the SVD Results

- Two algorithms to compute a description of a low-rank approximation D^* to a matrix matrix A which are qualitatively faster than the SVD.
- In the first algorithm, $c = O(1)$ columns of A are randomly chosen and used to form C; from $C^T C$ a description of an approximation to the top singular values and corresponding singular vectors of A may be computed such that rank$(D^*) \leq k$ and such that

$$\left\|A - D^*\right\|_\xi \leq \|A - A_k\|_\xi + poly(k, 1/c) \, \|A\|_F$$

  holds with high probability for both $\xi = 2, F$.
- Implementable without storing A in RAM, provided two passes over the matrix and $O(m + p)$ additional RAM memory.
- The second algorithm approximates C by randomly sampling $r = O(1)$ rows of C; additional error, three passes, and constant additional RAM memory.
- To achieve additional error $\leq \epsilon \|A\|_F$, both take time $poly(k, 1/\epsilon, \log(1/\delta)$; the first takes time linear in max(m, n) and the second takes time independent of m and n.

The CUR Approximate Decomposition

Given a matrix $A \in \mathbb{R}^{m \times n}$, we want an $A' \approx A$ s.t.:

1. $A' = CUR$, where C is an $m \times c$ matrix consisting of c randomly picked columns of A, R is an $r \times n$ matrix consisting of r randomly picked rows of A and U is a $c \times r$ matrix computed from C, R,
2. C, U, and R can be constructed after making a small constant number of passes through the whole matrix A from disk,
3. U can be constructed using additional RAM space and time that is either $O(m+n)$ or $O(1)$, independent of m and n,
4. for every $\epsilon > 0$ and every k such that $1 \leq k \leq$ rank(A) we can choose c and r such that A' satisfies

$$\left\|A - A'\right\|_F \leq \left\|A - A_k\right\|_F + \epsilon \left\|A\right\|_F ,$$

5. for every $\epsilon > 0$ and every k such that $1 \leq k \leq$ rank(A) we can choose c and r such that A' satisfies

$$\left\|A - A'\right\|_2 \leq \left\|A - A_k\right\|_2 + \epsilon \left\|A\right\|_F$$

   and thus we can choose c and r such that

$$\left\|A - A'\right\|_2 \leq \epsilon \left\|A\right\|_F .$$

The LinearTimeCUR Algorithm

Goal: Given a matrix $A \in \mathbb{R}^{m \times n}$ we wish to compute an approximate CUR decomposition in a constant number of passes through the data and additional space and time that is either $O(m+n)$ or $O(1)$, independent of m and n.

LINEARTIMECUR Algorithm Summary.

- Randomly sample c columns of A according to $\{q_j\}_{j=1}^{n}$ and rescale by $1/\sqrt{cq_{j_t}}$ to form $C \in \mathbb{R}^{m \times c}$.
- Randomly sample r rows of A according to $\{p_i\}_{i=1}^{m}$ and rescale by $1/\sqrt{rp_{i_t}}$ to form $R \in \mathbb{R}^{r \times n}$; sample the same r rows of C and rescale by $1/\sqrt{rp_{i_t}}$ to form $\Psi \in \mathbb{R}^{r \times c}$.
- Compute the SVD of $C^T C \in \mathbb{R}^{c \times c}$; say $C^T C = \sum_{t=1}^{c} \sigma_t^2(C) y^t {y^t}^T$.
- Let $\Phi = \sum_{t=1}^{k} \frac{1}{\sigma_t^2(C)} y^t {y^t}^T$ and define $U = \Phi \Psi^T \in \mathbb{R}^{c \times r}$.

$$\left(\quad A \quad \right) \approx \left(\quad C \quad \right) \left(\quad U \quad \right) \left(\quad R \quad \right)$$

The LinearTimeCUR Algorithm, Cont.

LinearTimeCUR Algorithm

Input: $A \in \mathbb{R}^{m \times n}$, $r, c, k \in \mathbb{Z}^+$ s.t. $1 \le r \le m$, $1 \le c \le n$, and $1 \le k \le \min(r, c)$, $\{p_i\}_{i=1}^m$ s.t. $p_i \ge 0$ and $\sum_{i=1}^m p_i = 1$, and $\{q_j\}_{j=1}^n$ s.t. $q_j \ge 0$ and $\sum_{j=1}^n q_j = 1$.

Output: $C \in \mathbb{R}^{m \times c}$, $U \in \mathbb{R}^{c \times r}$, and $R \in \mathbb{R}^{r \times n}$.

- For $t = 1$ to c,
  - Pick $j_t \in \{1, \ldots, n\}$ with $\mathbf{Pr}[j_t = k] = q_k$, $k = 1, \ldots, n$.
  - Set $C^{(t)} = A^{(j_t)} / \sqrt{c q_{j_t}}$.
- Compute $C^T C$ and its SVD; say $C^T C = \sum_{t=1}^c \sigma_t^2(C) y^t {y^t}^T$.
- If $\sigma_k(C) = 0$ then let $k = max\{k' : \sigma_{k'}(C) \neq 0\}$.
- For $t = 1$ to r,
  - Pick $i_t \in \{1, \ldots, m\}$ with $\mathbf{Pr}[i_t = k] = p_k$, $k = 1, \ldots, m$.
  - Set $R_{(t)} = A_{(i_t)} / \sqrt{r p_{i_t}}$.
  - Set $\Psi_{(t)} = C_{(i_t)} / \sqrt{r p_{i_t}}$.
- Let $\Phi = \sum_{t=1}^k \frac{1}{\sigma_t^2(C)} y^t {y^t}^T$ and define $U = \Phi \Psi^T$.
- Return C, U, and R.

Implementation of Linear (and Constant) Time CUR Algorithms

- Can calculate nearly optimal $\{p_i\}_{i=1}^m$ and $\{q_j\}_{j=1}^n$ in one pass and $O(c+r)$ space and time.

- C and R can then be constructed in one additional pass and $O(mc+nr)$ additional space and time.
  - C and R not explicitly constructed; in second pass compute nearly optimal π_i to construct W in $O(w)$ space and time and construct W in third pass with $O(cw)$ space and time

- Computing C^TC and the SVD of C^TC requires $O(mc^2+c^3)$ additional space and time.
  - Computing W^TW and the its SVD of W^TW requires $O(cw^2+c^3)$.

- Ψ can be constructed in the same second pass and $O(cr)$ space and time.

- Φ can be constructed with $O(c^2k)$ space and time.

- U can be computed using $O(c^2r)$ space and time.

- Overall, $O(m+n)$ additional space and time.
  - Overall, $O(1)$.

Intuition for the CUR algorithms

$H_k H_k^T A$ is an approximation to A, but it can't be computed (in this form) in a small number of passes with sublinear additional space and time.

Can $H_k H_k^T A$ be approximated?

Lemma. [DKM] *Let S_R be the row sampling matrix and D_R the associated diagonal rescaling matrix. Then:*

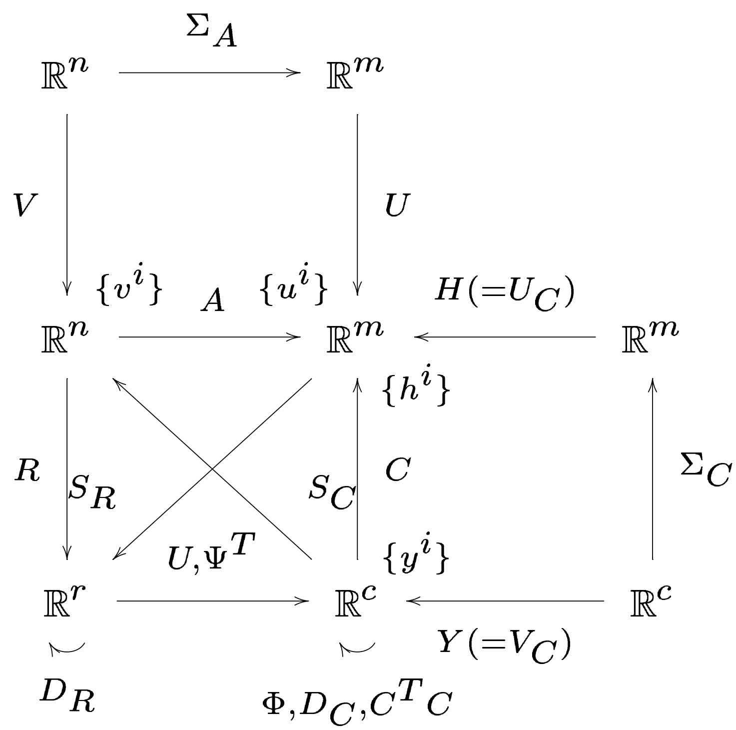
- $CUR = H_k H_k^T (D_R S_R)^T D_R S_R A = H_k \tilde{H}_k^T \tilde{A}$
- $\|A - CUR\|_\xi \leq \|A - H_k H_k^T A\|_\xi + \|H_k H_k^T A - CUR\|_\xi$
- $\left\|H_k H_k^T A - CUR\right\|_F = \left\|H_k^T A - \tilde{H}_k^T \tilde{A}\right\|_F$

Note: Sampling probabilities are **not** nearly optimal with respect to approximating the product $H_k^T A$; thus must use Markov's inequality.

Note: Can view CUR as a "dimensional reduction" technique.

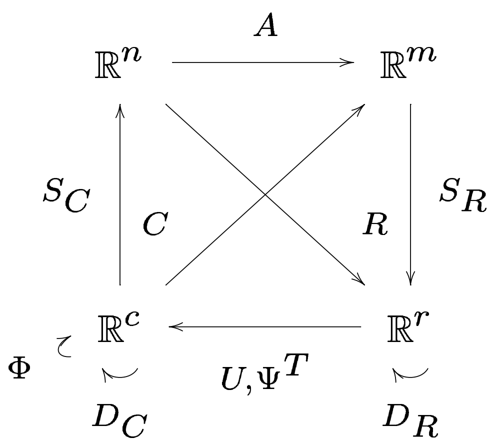
Note: Given A, a database of vectors, and q, a query vector, compute $A'q = CURq$ rather than Aq to identify nearest neighbors.

The LinearTimeCUR Algorithm, Cont.



$$H_k H_k^T A \approx A$$
$$\tilde{H}_k^T \tilde{A} \approx H_k^T A$$

The LinearTimeCUR Algorithm, Cont.



The LinearTimeCUR Algorithm, Cont.

Theorem. [DKM] *Suppose* $A \in \mathbb{R}^{m \times n}$ *and let* C, U, *and* R *be constructed from the* $\textsc{LinearTimeCUR}$ *algorithm by sampling* c *columns of* A *with nice probabilities* $\{q_j\}_{j=1}^n$ *and* r *rows of* A *with nice probabilities* $\{p_i\}_{i=1}^m$. *Let* $\eta_c = 1 + \sqrt{(8/\beta_c)\log(1/\delta_c)}$ *and let* $\delta = \delta_r + \delta_c$. *If* $c = \Omega(k\eta_c^2/\epsilon^4)$ *and* $r = \Omega(k/\delta_r^2\epsilon^2)$, *then*

$$\|A - CUR\|_F \leq \|A - A_k\|_F + \epsilon' \|A\|_F$$

both in expectation and with high probability. If $c = \Omega(\eta_c^2/\epsilon^4)$ *and* $r = \Omega(k/\delta_r^2\epsilon^2)$, *then*

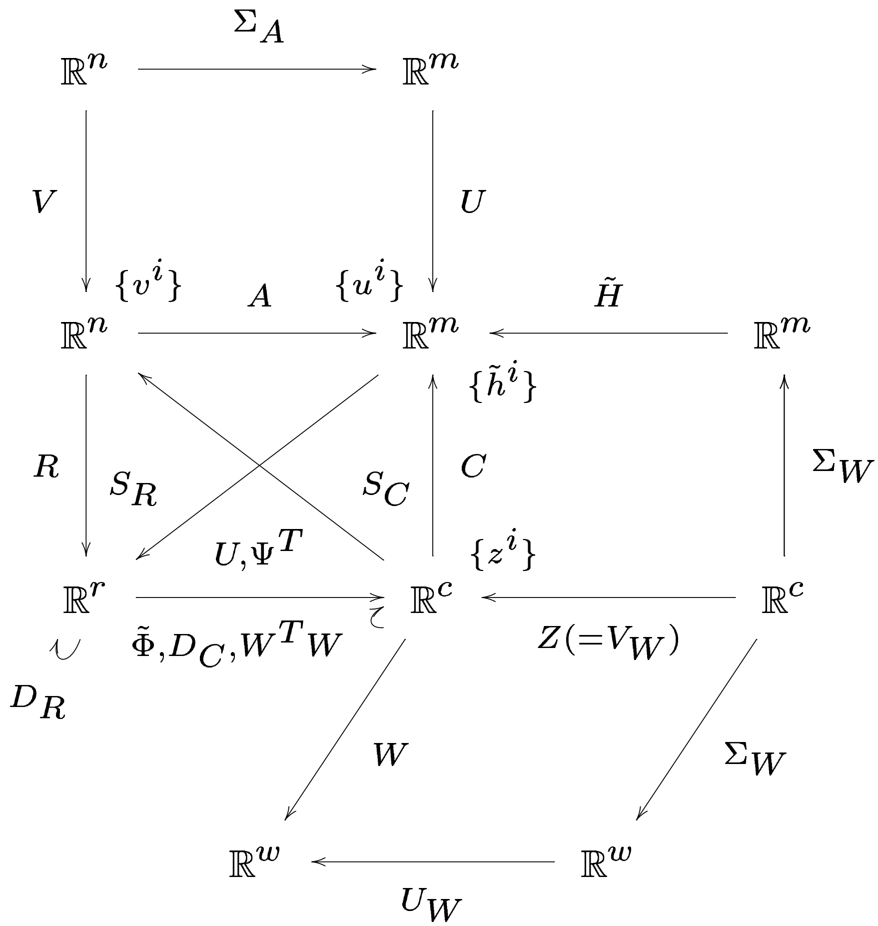
$$\|A - CUR\|_2 \leq \|A - A_k\|_2 + \epsilon' \|A\|_F$$

both in expectation and with high probability.

Proof. Submultiplicitivity; then apply approximate SVD results and approximate matrix multiplication results (for non-optimal probabilities). □

ConstantTimeCUR Algorithm Summary.

- Similar to the LinearTimeCUR algorithm, except that singular values and singular vectors of $C^T C$ are approximated by those of $W^T W$; say $W^T W = \sum_{t=1}^{c} \sigma_t^2(W) z^t {z^t}^T$
- Let $\tilde{\Phi} = \sum_{t=1}^{k} \frac{1}{\sigma_t^2(W)} z^t {z^t}^T$ and define $\tilde{U} = \tilde{\Phi} \Psi^T \in \mathbb{R}^{c \times r}$.



CONSTANTTIMECUR Algorithm

Input: $A \in \mathbb{R}^{m \times n}$, $r, c, k \in \mathbb{Z}^+$ s.t. $1 \leq r \leq m$, $1 \leq c \leq n$, and $1 \leq k \leq \min(r, c)$, $\{p_i\}_{i=1}^m$ s.t. $p_i \geq 0$ and $\sum_{i=1}^m p_i = 1$, and $\left\{q_j\right\}_{j=1}^n$ s.t. $q_j \geq 0$ and $\sum_{j=1}^n q_j = 1$.

Output: $U \in \mathbb{R}^{c \times r}$ and a "description" of $C \in \mathbb{R}^{m \times c}$ and $R \in \mathbb{R}^{r \times n}$.

- For $t = 1$ to c,
  - Pick $j_t \in \{1, \ldots, n\}$ with $\mathbf{Pr}\left[j_t = k\right] = q_k$, $k = 1, \ldots, n$ and save $\{(j_t, q_{j_t}) : t = 1, \ldots, c\}$.
  - Set $C^{(t)} = A^{(j_t)} / \sqrt{c q_{j_t}}$.
- Choose $\{\pi_i\}_{i=1}^m$ s.t. $\pi_i \geq 0$, $\sum_{i=1}^m \pi_i = 1$, and $\pi_i \geq \beta' \left|C_{(i)}\right|^2 / \|C\|_F^2$ with $\beta' < 1$.
- For $t = 1$ to w,
  - Pick $i_t \in 1, ..., m$ with $\mathbf{Pr}\left[i_t = k\right] = \pi_k$, $k = 1, \ldots, m$.
  - Set $W_{(t)} = C_{(i_t)} / \sqrt{w \pi_{i_t}}$.
- Compute $W^T W$ and its SVD; say $W^T W = \sum_{t=1}^c \sigma_t^2(W) z^t z^{t^T}$.
- If a $\|\cdot\|_F$ bound is desired
  - Set $\gamma = \Theta(\epsilon^2/k)$,

  Else if a $\|\cdot\|_2$ bound is desired
  - Set $\gamma = \Theta(\epsilon^2)$.
- Let l be the index of the smallest singular value such that $\sigma_t^2(W) \geq \gamma \|W\|_F^2$.
- Keep $min\{l, k\}$ singular values $\sigma_t(W)$ and their corresponding singular vectors $\left\{z^t\right\}_{t=1}^l$.
- For $t = 1$ to r,
  - Pick $i_t \in \{1, \ldots, m\}$ with $\mathbf{Pr}\left[i_t = k\right] = p_k$, $k = 1, \ldots, m$ and save $\{(i_t, p_{i_t}) : t = 1, \ldots, r\}$.
  - Set $R_{(t)} = A_{(i_t)} / \sqrt{r p_{i_t}}$.
  - Set $\Psi_{(t)} = C_{(i_t)} / \sqrt{r p_{i_t}}$.
- Let $\tilde{\Phi} = \sum_{t=1}^l \frac{1}{\sigma_t^2(W)} z^t z^{t^T}$ and define $\tilde{U} = \tilde{\Phi} \Psi^T$.
- Return U, c column labels $\{(j_t, q_{j_t}) : t = 1, \ldots, c\}$, and r row labels $\{(i_t, p_{i_t}) : t = 1, \ldots, r\}$.

- (If an explicit solution is to be computed then) using the column and row labels compute C and R and return C, U, and R.

The ConstantTimeCUR Algorithm, Cont.

Theorem. [DKM] *Suppose* $A \in \mathbb{R}^{m \times n}$ *and let* C*,* $\tilde{U}$*, and* R *be constructed from the* $\textsc{ConstantTimeCUR}$ *algorithm by sampling* c *columns of* A *with probabilities* $\{q_j\}_{j=1}^{n}$ *(and then sampling* w *rows of* C *with probabilities* $\{\pi_i\}_{i=1}^{m}$ *to construct* W*) and* r *rows of* A *with probabilities* $\{p_i\}_{i=1}^{m}$*. Assume that* $p_i \geq \beta_r \left|A_{(i)}\right|^2 / \|A\|_F^2$*,* $q_j \geq \beta_c \left|A^{(j)}\right|^2 / \|A\|_F^2$*, and* $\pi_i \geq \beta' \left|C_{(i)}\right|^2 / \|C\|_F^2$ *for some positive constants* $\beta_r, \beta_c, \beta' \leq 1$*. Let* $\delta, \epsilon > 0$*. If* $\gamma = \Theta(\epsilon^2/k)$*,* $c = \Omega\left(k^3/\epsilon^8\right)$*,* $w = \Omega\left(k^3/\epsilon^8\right)$*, and* $r = \Omega\left(k^2/\epsilon^2\right)$*, then with probability* $\geq 1 - \delta$*:*

$$\left\|A - C\tilde{U}R\right\|_F \leq \|A - A_k\|_F + \epsilon \|A\|_F .$$

If $\gamma = \Theta(\epsilon^2)$*,* $c = \Omega\left(1/\epsilon^4\right)$*,* $w = \Omega\left(1/\epsilon^6\right)$*, and* $r = \Omega\left(k^2/\epsilon^2\right)$*, then with probability* $\geq 1 - \delta$*:*

$$\left\|A - C\tilde{U}R\right\|_2 \leq \|A - A_k\|_2 + \epsilon \|A\|_F$$

Comparison of SVD and CUR

- The SVD of A is: $A = \sum_{t=1}^{\rho} \sigma_t(A) u^t {v^t}^T$.

- $A_k = \sum_{t=1}^{k} \sigma_t(A) u^t {v^t}^T = U_k \Sigma_k V_k^T$ gives us the "optimal" rank k approximation and requires $O(m+n)$ space if $k = O(1)$.

- CUR achieves weaker bounds which are similar in spirit.

- Think of the SVD as rotation followed by a rescaling followed by a rotation.

- Think of the CUR decomposition as more like A followed by $A^\dagger$ followed by A.

Summary of CUR

- Two algorithms to compute an approximation to A which is the product of three smaller matrices, C, U, and R, each of which may be computed rapidly.
- Let $A' = CUR$; both algorithms have provable bounds for the error matrix $A - A'$.
- In the first algorithm, $c = O(1)$ columns of A and $r = O(1)$ rows of A are randomly chosen to form C and R, repsectively; U calculated from C and R.
- $\left\|A - A'\right\|_\xi \leq \|A - A_k\|_\xi + poly(k, 1/c) \|A\|_F$ holds in expectation and with high probability for both $\xi = 2, F$ and for all $k = 1, \ldots, rank(A)$.
- By appropriate choice of k: $\left\|A - A'\right\|_2 \leq \epsilon \|A\|_F$
- Implementable without storing the matrix A in RAM, provided two passes over the matrix and $O(m + n)$ additional RAM memory.
- The second algorithm is similar except that it approximates the matrix C by randomly sampling $O(1)$ rows of C; additional error, three passes, and constant additional RAM memory.
- To achieve additional error $\leq \epsilon \|A\|_F$, both take time $poly(k, 1/\epsilon, 1/\delta)$; the first takes time linear in max(m, n) and the second takes time independent of m and n.

Lower Bounds

- How many queries does a sampling algorithm need to approximate a given function accurately with high probability?
- ZBY03 proves lower bounds for the low rank matrix approximation problem and the matrix reconstruction problem.
  - Any sampling algorithm that with high probability finds a good low rank approximation requires $\Omega(m+n)$ queries.
  - Even if the algorithm is given the exact weight distribution over the columns of a matrix it will still require $\Omega(k/\epsilon^4)$ queries.
  - Finding a matrix D such that $\|A-D\|_F \leq \epsilon \|A\|_F$ requires $\Omega(mn)$ queries and that finding a D such that $\|A-D\|_2 \leq \epsilon \|A\|_F$ requires $\Omega(m+n)$ queries.
- Applied to our results:
  - The LinearTimeSVD algorithm is optimal with respect to $\|\cdot\|_F$ bounds; see also DFKVV99.
  - The ConstantTimeSVD algorithm is optimal with respect to $\|\cdot\|_F$ bounds up to polynomial factors; see also FKV98.
  - The CUR algorithm is optimal for constant ϵ.

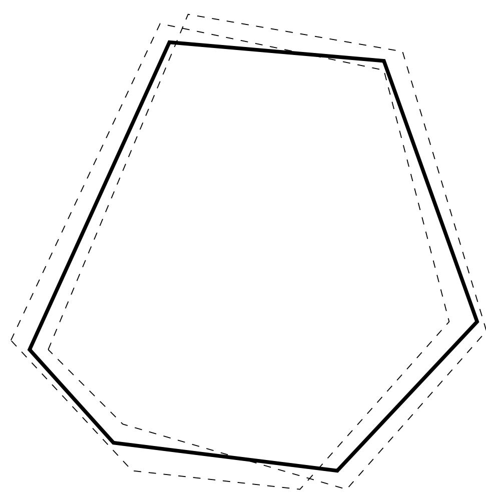
Review of Linear Programming

Primal LP: $\max cx$ s.t. $Px \leq b$, $x \geq 0$.

Def: x is a *feasible solution* if $Px \leq b$ and $x \geq 0$.

Dual LP: $\min by$ s.t. $P^T y \geq c$, $y \geq 0$.

Note: The feasible region is a convex polyhedron.



Sampling Linear Programs

Theorem. [DKM] *Let $P \in \mathbb{R}^{r \times n}$, $b \in \mathbb{R}^r$ and consider:*

$$Px = \sum_{i=1}^{n} P^{(i)} x_i \leq b \qquad 0 \leq x_i \leq c_i. \tag{1}$$

Suppose Q is a random subset of $\{1, 2, \ldots n\}$, with $|Q| = q$, formed by picking elements of $\{1, 2, \ldots n\}$ with probability

$$p_i = \mathbf{Pr}\left[i_t = i\right] = \frac{c_i \left| P^{(i)} \right|}{\mathcal{N}},$$

where $\mathcal{N} = \sum_{i=1}^{n} c_i \left| P^{(i)} \right|$. Let $\eta = 1 + \sqrt{8 \log(1/\delta)}$. If LP (1) is feasible, then with probability at least $1 - \delta$

$$\sum_{i \in Q} \frac{1}{q p_i} P^{(i)} x_i \leq b + \frac{\eta \mathcal{N}}{\sqrt{q}} \vec{\mathbf{1}}_r \qquad 0 \leq x_i \leq c_i$$

is feasible as well. If LP (1) is infeasible, then with probability at least $1 - \delta$

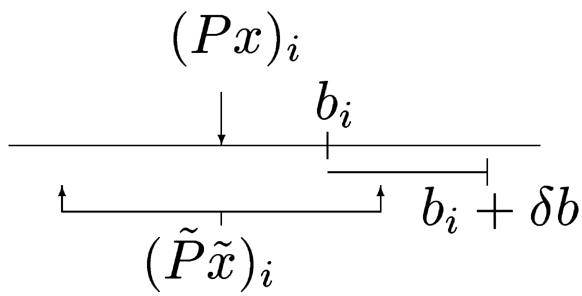
$$\sum_{i \in Q} \frac{1}{q p_i} P^{(i)} x_i \leq b - \frac{\eta \mathcal{N}}{\sqrt{q}} \vec{\mathbf{1}}_r \qquad 0 \leq x_i \leq c_i$$

is infeasible as well.

Proof. Uses matrix multiplication ideas and also LP duality. □

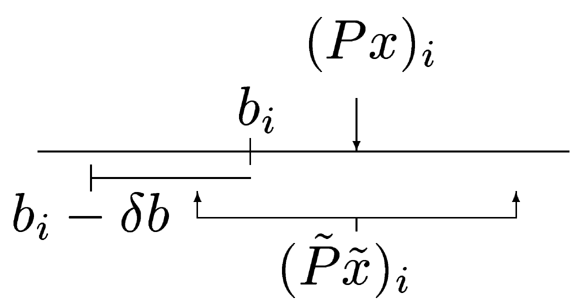
Sampling Linear Programs, Cont.

- If $\{Px \le b,\ 0 \le x_i \le c_i\}$ is *feasible* then $\forall\ i$:



Thus, $\{\tilde{P}\tilde{x} \le b + \delta b \vec{\mathbf{1}}_n,\ 0 \le x_i \le c_i\}$ is also *feasible*.

- If $\{Px \le b,\ 0 \le x_i \le c_i\}$ is *infeasible* then $\exists\ i$:



Thus, $\{\tilde{P}\tilde{x} \le b - \delta b \vec{\mathbf{1}}_n,\ 0 \le x_i \le c_i\}$ is also *infeasible*.

- **Note:** If $c_i = 1$ for all i, then $\sum_{i=1}^{n} \left| P^{(i)} \right| \le \sqrt{n} \left\| P \right\|_F$ and the induced LP becomes

$$\sum_{i \in Q} \frac{1}{q p_i} P^{(i)} x_i \le b \pm \eta \sqrt{\frac{n}{q}} \left\| P \right\|_F \vec{\mathbf{1}}_r \qquad 0 \le x_i \le 1.$$

Other Perturbed LP Results

- Renegar '94, '95:
  - Developing a complexity theory for real data.
  - Customary measures of size were replaced with condition measures.
  - Consider: $Px \leq b$, $x \geq 0$; to decide whether $d = (P, b)$ is a consistent system of constraints, consider the minimal relative perturbation of d.

- Spielman and Teng '01: "smoothed complexity"
  - Studying the performance of algorithms under small random perturbation of their inputs.
  - Consider: $\max z^T x$ s.t. $Px \leq b$.
  - Replace it with: $\max z^T x$ s.t. $(P + \sigma G)x \leq b$.

- We perturb Px to $\tilde{P}\tilde{x}$ and then choose a new b.

- We replace $Px \leq b$ with $\tilde{P}\tilde{x} \leq b \pm \delta b$.

- δb can be quite large.

Review of Max-Cut

- Let $G = (V, E)$ be a graph with $|V| = n$ and edge weights $w : E \to \mathbb{R}$.

- For $S \subset V$ let $cut(S, \overline{S})$ be those edges with exactly one end in S and the weight $w(S, \overline{S})$ be the sum of the weights of the edges.

- The *maximum weight cut problem* or the Max-Cut problem is: Find a cut (or the weight of a cut) with maximum weight over all possible cuts.

- Max-Cut is NP-hard, both in general and for dense graphs.

- There exists a constant α, bounded away from 1, such that (assuming $P \neq NP$) it is not possible to α-approximate Max-Cut; thus, there is no PTAS for Max-Cut.

- A *polynomial time approximation scheme* (PTAS) is an algorithm that for every fixed $\epsilon > 0$ achieves an approximation ratio of $1 - \epsilon$ in time $poly(n)$.

Review of Approximating Max-Cut and Max-2-CSP Problems

- Goemans and Williamson '94:
  - 0.878-approximation algorithm.
- Arora, Karger, and Karpinski '95: ϵn^2 additive error
  - Linear Programming and Randomized Rounding.
  - $O\left(n^{O(1/\epsilon^2)}\right)$ time.
- De La Vega '96: ϵn^2 additive error
  - Combinatorial methods.
  - $O(n^2 2^{1/\epsilon^{2+o(1)}})$ time.
- Frieze and Kannan '96: ϵn^2 additive error (*)
  - Efficient version of Szemerédi's Regularity Lemma.
  - PTAS for dense graph problems like Max-Cut.
  - $O(poly(1/\epsilon)n^{2.6} + \beta(1/\epsilon))$ time.
- Goldreich, Goldwasser, and Ron '96: ϵn^2 additive error
  - Query complexity and property testing methods.
  - $O(1/\epsilon^5)$ sampling complexity.
  - Constructing the Max-Cut takes $O(n)$ time.
- Frieze and Kannan '97: ϵn^2 additive error (*)
  - New method to approximate matrices.
  - PTAS for all dense Max-CSP problems.
  - $2^{O(1/\epsilon^2)}$ time.
- Alon, De La Vega, Kannan, and Karpinski '03: ϵn^2 additive error (*)
  - PTAS for all dense Max-CSP problems.
  - $O\left(\frac{\log 1/\epsilon}{\epsilon^4}\right)$ sampling complexity for Max-Cut, etc.
- De La Vega and Karpinski '04:
  - PTAS in subdense Max-2-CSP problems, e.g., graphs with $\Omega(n^2/\log n)$ edges.

Approximation Algorithm for Max-Cut

Let $G = (V, E)$ be a graph with adjacency matrix A:

- **MAX-CUT** $[A] = \max_{x \in \{0,1\}^n} x^T A(\vec{\mathbf{1}}_n - x)$.
- Reduce this to **MAX-CUT** $\left[C\tilde{U}R\right]$.
- Reduce this to testing a constant number of integer programs **IP**(u, v) , $(u, v) \in \Omega_\Delta$ for feasibility.
- Relax **IP**(u, v) to **LP**(u, v).
- Sample from **LP**(u, v) to construct **LP**$_Q(u, v)$.
  - Use LP Sampling Theorem.
  - Check each of the large but constant number of these LPs for feasibility in constant time.

Approximation Algorithm for Max-Cut, Cont.

Our new result: Approximate the Max-Cut of $G = (V, E)$ up to additive error

$$\epsilon n \left\|A\right\|_F = \epsilon n \sqrt{2\left|E\right|}$$

in constant time and space after reading the graph three times. We need to keep $O(1/\epsilon^{26})$ entries from the adjacency matrix of G.

Note: Particularly useful for graphs with nonuniformities and heterogenities.

- Unweighted graphs: better as $\left|E\right|/n^2$ decreases.
- Weighted graphs: $\epsilon n \left\|A\right\|_F$ versus $\epsilon n^2 W_{max}$.

Also: Approximate Max-2-CSP problems in a similar manner and with similar error.