

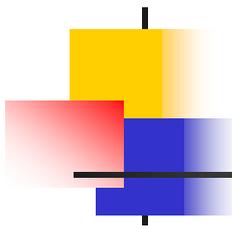
Extracting Structure from Matrices & Tensors by Random Sampling

Michael W. Mahoney

Yale University
michael.mahoney@yale.edu

(joint work with **Petros Drineas** and **Ravi Kannan**)

@ ARCC Workshop on Tensor Decompositions



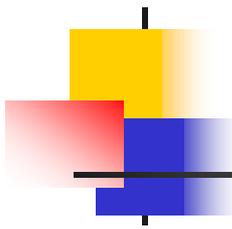
Motivation

Goal: To develop and analyze fast Monte-Carlo algorithms for performing useful computations on large matrices (and tensors):

- Matrix Multiplication
- Computation of the SVD
- 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 are useful in applications where the datasets are modeled by *matrices (or tensors)* and are *extremely large*.

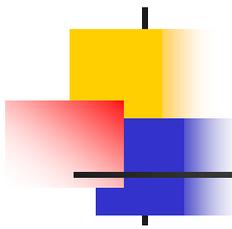


Motivation (cont'd)

In many applications **large matrices** appear (**too large to store in RAM**).

- We can make a **few "passes"** (sequential READS) through the matrices.
- We can create and store a **small "sketch"** of the matrices in RAM.
- Computing the "sketch" should be a very **fast** process.

Discard the original matrix and work with the "sketch".



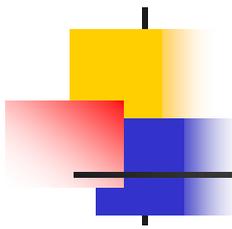
Our approach & our results

1. A “sketch” consisting of a **few rows/columns** of the matrix is adequate for efficient approximations.
2. We draw the rows/columns randomly, using **non-uniform sampling**; rows/columns are picked with probability proportional to their lengths.

- Create an approximation to the original matrix which can be stored in much less space.

$$\begin{pmatrix} A \end{pmatrix} \approx \begin{pmatrix} C \end{pmatrix} \cdot \begin{pmatrix} U \end{pmatrix} \cdot \begin{pmatrix} R \end{pmatrix}$$

- Generalize CUR to tensors.

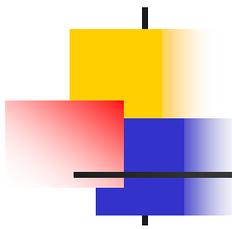


Approximating A by CUR

Given a **large** m-by-n matrix A (stored on disk), compute an approximation $A' = CUR$ to A such that:

1. $A' = CUR$ is stored in $O(m+n)$ space, after making **two** passes through A, and using $O(m+n)$ additional space and time.
2. $A' = CUR$ satisfies, with high probability,

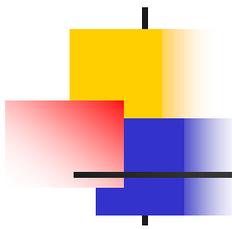
$$\|A - CUR\|_{F,2}^2 \leq \|A - A_k\|_{F,2}^2 + \varepsilon \|A\|_F^2$$



Computing C and R

$$\begin{pmatrix} A \\ m \times n \end{pmatrix} \approx \begin{pmatrix} C \\ m \times c \end{pmatrix} \cdot \begin{pmatrix} U \\ c \times r \end{pmatrix} \cdot \begin{pmatrix} R \\ r \times n \end{pmatrix}$$

- C consists of $c = O(k/e^2)$ columns of A.
- R consists of $r = O(k/e^2)$ rows of A.
- C and R are created using **non-uniform sampling**: $p_i = \frac{|A^{(i)}|^2}{\|A\|_F^2}$.



Computing U

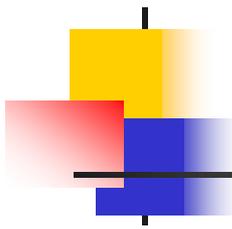
The CUR algorithm essentially expresses every row of the matrix A as a linear combination of a small subset of the rows of A .

- This small subset consists of the rows in R .
- Given a row of A – say $A_{(i)}$ – the algorithm computes the “best fit” for the row $A_{(i)}$ using the rows in R as the basis.

$$\text{i.e. } \min_u \left\| \begin{pmatrix} A_{(i)} \\ \phantom{A_{(i)}} \end{pmatrix} - \begin{pmatrix} u \\ \end{pmatrix} \cdot \begin{pmatrix} R \\ \end{pmatrix} \right\|_2$$

$1 \times n$ $1 \times r$ $r \times n$

Notice that only $c = O(1)$ elements of the i -th row are given as input. However, a vector of coefficients u can still be computed. The whole process runs in $O(m)$ time.



Main Theorem

Assume A_k is the “best” rank k approximation to A (through SVD). Then

$$\|A - CUR\|_F^2 \leq \|A - A_k\|_F^2 + \varepsilon \|A\|_F^2 \quad (1)$$

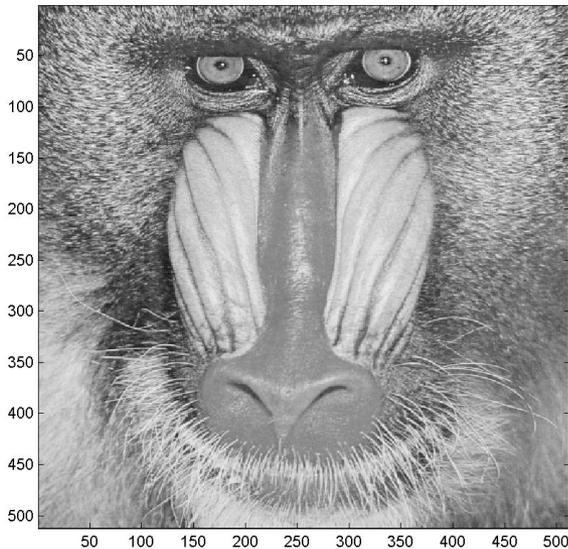
$$\|A - CUR\|_2^2 \leq \|A - A_k\|_2^2 + \varepsilon \|A\|_F^2 \quad (2)$$

To achieve (1), we set $r = O(k/\varepsilon^2)$ rows and $c = O(k/\varepsilon^2)$ columns.

To achieve (2), $r = O(1/\varepsilon^2)$ rows and $c = O(1/\varepsilon^2)$ columns suffice.

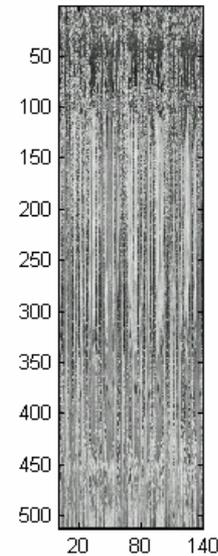
A starting point for CUR

A



Original matrix

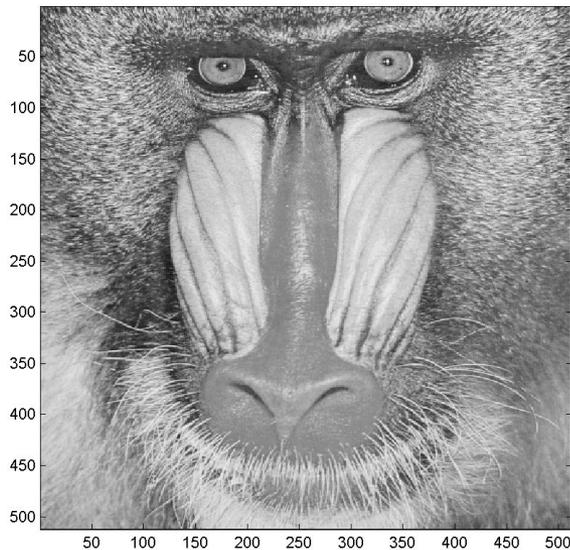
C



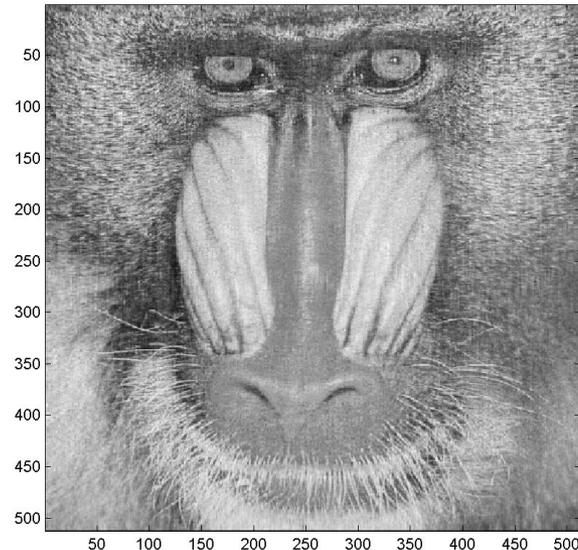
After sampling columns

1. Compute the top k left singular vectors of the matrix C and store them in the 512 -by- k matrix H_k .
2. A and $H_k H_k^T A$ are close.

A starting point for CUR (cont'd)



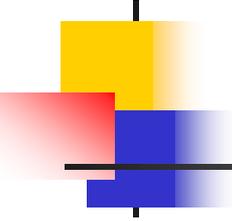
A



$H_k H_k^T A$

Question: How is $H_k H_k^T A$ related to CUR?

Answer: If we replace A by a matrix which consists of a carefully scaled copy of the rows "revealed" in R, then we get CUR!



A straightforward extension to tensors

Let $\mathcal{A} \in \mathcal{R}^{n_1 \times \dots \times n_d}$.

(1) (for all $i = 1 \dots d$) Let the $n_i \times n_i$ matrix U_i denote the left singular vectors of the $n_i \times (n_1 \dots n_{i-1} n_{i+1} \dots n_d)$ matrix $A_{[i]}$.

Running time (for a fixed i):

$$O(n_1 \dots n_{i-1} n_i^2 n_{i+1} \dots n_d)$$

(2) If $\mathcal{Z} = \mathcal{A} \times_1 U_1^T \dots \times_d U_d^T$, then

$$\mathcal{A} = \mathcal{Z} \times_1 U_1 \dots \times_d U_d.$$

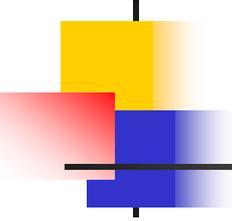
(1') (for all $i = 1 \dots d$) Approximate the left singular vectors of $A_{[i]}$ by sampling c_i columns of $A_{[i]}$, thus computing \tilde{U}_i .

Running time (for a fixed i):

$$O(\min\{c_i^2 n_i, c_i n_i^2\})$$

We can bound the error $\|\mathcal{A} - \mathcal{A} \times_1 (\tilde{U}_1 \tilde{U}_1^T) \dots \times_d (\tilde{U}_d \tilde{U}_d^T)\|_F^2$.

(Preliminary results - work in progress)



A recursive extension of CUR to tensors

Let $\mathcal{A} \in \mathcal{R}^{n_1 \times n_2 \times n_3}$ be a 3-way tensor...

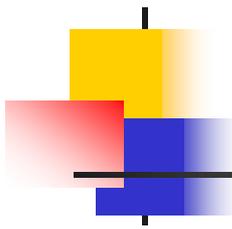
(1) Randomly sample “pages” $\mathcal{A}(:, :, i_t)$, $t = 1 \dots c = O(1)$.

These c “pages” form a “basis”.

(2) Approximate each of the above “pages” by sampling a constant number of their rows and columns via the CUR approximation algorithm.

(3) Express the “pages” of the tensor that were not sampled as a linear combination of the above “basis pages”.

(Preliminary results - work in progress)



Thank you!

For more details:

<http://www.cs.rpi.edu/~drinep>

<http://www.cs.yale.edu/homes/mmahoney>