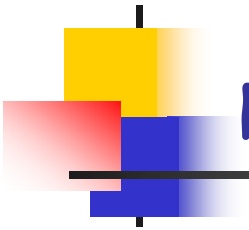
Fast Monte Carlo Algorithms for Matrix Operations & Massive Data Set Analysis

Michael W. Mahoney

Yale University
Dept. of Mathematics
http://cs-www.cs.yale.edu/homes/mmahoney

Joint work with:
P. Drineas and R. Kannan
and also with:
S. Muthukrishnan, M. Maggioni, R. Coifman, O. Alter, and F. Meyer



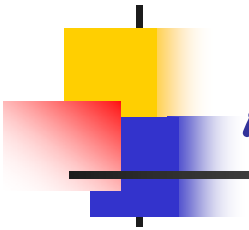
Randomized Linear Algebra Algorithms

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., $O(n^3)$ 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 an m x n matrix A.

- Covariance Matrices
- Latent Semantic Indexing
- DNA Microarray Data
- Eigenfaces and Image Recognition
- Similarity Query
- Matrix Reconstruction

- Linear Programming Applications
- Approximation Algorithm Applications
- Statistical Learning Theory Applications

Review of Linear Algebra

F-norm: $\|A\|_F^2 = \sum_{ij} A_{ij}^2$

2-norm: $\|A\|_2 = \sup_{x \in R^n,\ x \neq 0} \frac{\|Ax\|}{\|x\|}$

SVD: $A = U \Sigma V^T$

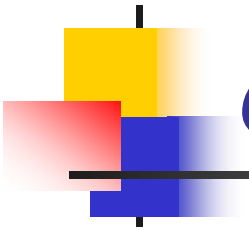
$A_k = U_k \Sigma_k V_k^T$

SPSD: $x^T A x \geq 0 \ \forall x \neq 0$

MPGI: $A^+ = V \Sigma^{-1} U^T$

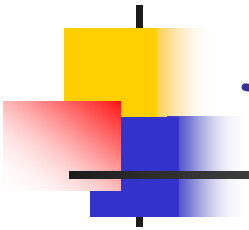
$$\max_{t:1 \leq t \leq n} |\sigma_t(A+E) - \sigma_t(A)| \leq \|E\|_2$$

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



Overview and Summary

- Pass-Efficient Model and Random Sampling
- Matrix Multiplication
- Singular Value Decomposition
- Lower Bounds
- CUR Decomposition
- Kernel-based data sets and KernelCUR
- Tensor-based data sets and TensorCUR
- Large scientific (e.g., chemical and biological) data



The Pass Efficient Model

Motivation: Amount of disk/tape space has increased enormously; RAM and computing speeds have increased less rapidly.

- Can store large amounts of data.
- **Cannot** process these data with traditional algorithms.

In the **Pass-Efficient Model**:

- Data are assumed to be stored on disk/tape.
- Algorithm has access to the data via a pass (a pass is a sequential read of the entire input from disk).
- 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.

Note: If data are an m x n matrix A, then algorithms which require additional time and space that is $O(m+n)$ or $O(1)$ are pass-efficient.

Approximating Matrix Multiplication

(See: Drineas & Kannan FOCS '01 and Drineas, Kannan, & Mahoney TR '04, SICOMP '05)

Problem : Given an m-by-n matrix A and an n-by-p matrix B:

Approximate the product A·B,

OR

Approximate the sum of *n* rank-one matrices.

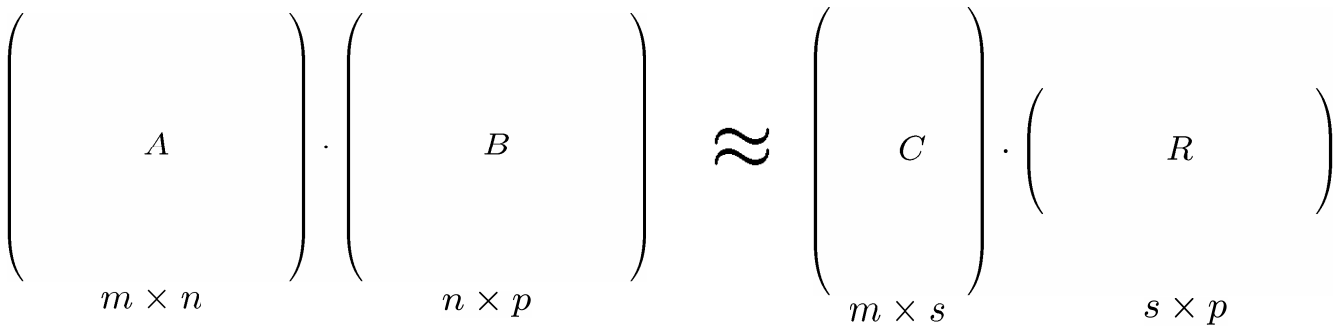
$$A \cdot B = \sum_{i=1}^{n} \underbrace{\left(A^{(i)} \right) \cdot \left(B_{(i)} \right)}_{\in \mathcal{R}^{m \times p}}$$

i-th column of A

i-th row of B

Each term in the summation is a rank-one matrix

Matrix multiplication algorithm



- Sample s columns of A to form an m-by-s matrix C and the corresponding s rows of B to form an s-by-p matrix R in s i.i.d. trials.
- Sample a column $A^{(i)}$ and a row $B_{(i)}$ with nonuniform probability $\{p_i\}$.
- Include $A^{(j_t)}/(sp_{j_t})^{1/2}$ as a column of C, and $B_{(j_t)}/(sp_{j_t})^{1/2}$ as a row of R.

Note: C and R consist of rescaled copies of the sampled columns and rows.

Notes about the algorithm

- The matrix A is given in "sparse unordered representation"; non-zero entries of A are presented as unordere triples (i, j, A_{ij}).
- Can implement the sampling in two passes and $O(n)$ (or $O(1)$ if $B=A^T$) RAM space.
- Can implement the algorithm with $O(sm+sp)$ RAM space and time.

- The expectation of CR is AB (element-wise) for any $\{p_i\}$.
- If we sample with the nonuniform probabilities:

$$p_i \geq \beta_i \, ||A^{(i)}||_2 ||B_{(i)}||_2 / \Sigma_i \, ||A^{(i)}||_2 ||B_{(i)}||_2$$

(with $\beta = 1$ now, but not later) then the variance is minimized.

Error bounds for the algorithm

For this sampling-based matrix multiplication algorithm (with β=1):

$$E\left(\|AB - CR\|_{2,F}\right) \leq \frac{1}{\sqrt{s}}\|A\|_F\|B\|_F$$

If B = A^T, then the sampling probabilities are $p_i = ||A^{(i)}||^2/||A||_F^2$ and:

$$E\left(\left\|AA^T - CC^T\right\|_{2,F}\right) \leq \frac{1}{\sqrt{s}}\|A\|_F^2$$

- Can prove tight concentration results via a martingale argument.
- If $||AB||_F = \Omega(||A||_F\,||B||_F)$, (i.e., if there is "not much cancellation") then this is a relative error bound.
- (Slight β-dependent loss if $\beta \neq 1$.)
- Vershynin improves the spectral norm bound for the case B = A^T.

Fast - O(n) - SVD computations

(See: Frieze, Kannan & Vempala FOCS '98, Drineas, Frieze, Kannan, Vempala & Vinay SODA '99, Etc. and Drineas, Kannan, & Mahoney TR '04, SICOMP '05)

Given: m x n matrix *A*

- Sample c columns from *A* and rescale to form the m x c matrix *C*.
- Compute the m x k matrix H_k of the k left singular vectors of *C*.

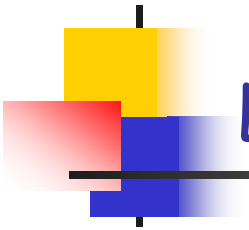
Structural Theorem: *For any probabilities* and number of columns:

$$||A-H_kH_k^TA||_{2,F}^2 \leq ||A-A_k||_{2,F}^2 + 2\sqrt{k}||AA^T-CC^T||_F$$

Algorithmic Theorem: *If* $p_i = |A^{(i)}|^2/||A||_F^2$ and $c \geq 4\,\eta^2 k/\varepsilon^2$, then:

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

Proof: Matrix multiplication.



Lower Bounds

Question: 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.

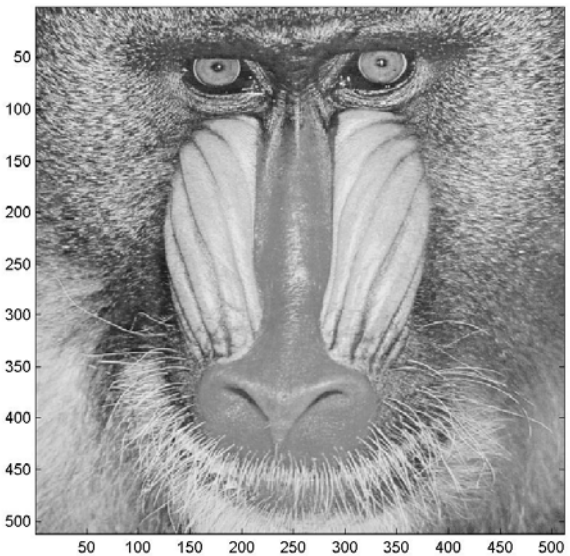
- Any sampling algorithm that w.h.p. finds a good low rank approximation requires $\Omega(m+n)$ queries.
- Even if the algorithm is given the exact weight distribution over the columns of a matrix it will still require $\Omega(k/\varepsilon^4)$ queries.
- Finding a matrix D such that $||A-D||_F \leq \varepsilon ||A||_F$ requires $\Omega(mn)$ queries and that finding a D such that $||A-D||_2 \leq \varepsilon ||A||_2$ requires $\Omega(m+n)$ queries.

Applied to our results:

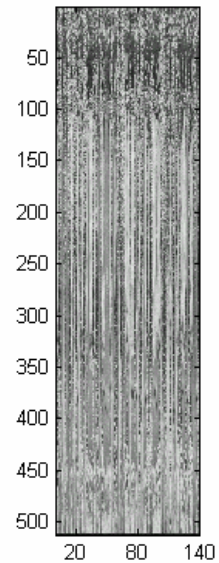
- The **LinearTimeSVD** algorithm is optimal w.r.t. $||\bullet||_F$ bounds; see also DFKVV99.
- The **ConstantTimeSVD** algorithm is optimal w.r.t. $||\bullet||_2$ bounds up to poly factors; see also FKV98.
- The **CUR** algorithm is optimal for constant ε.

Example of randomized SVD

A



C

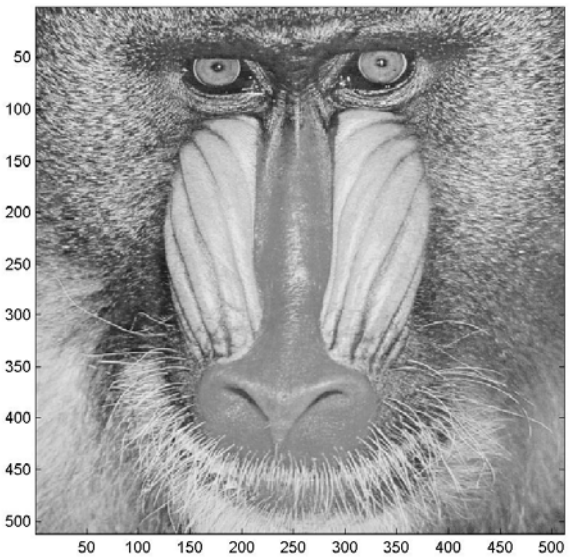


Original matrix

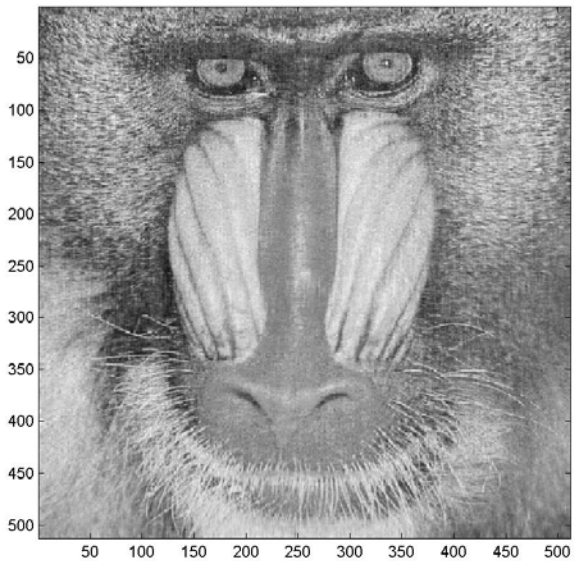
After sampling columns

Compute the top k left singular vectors of the matrix C and store them in the 512-by-k matrix H_k.

Example of randomized SVD (cont'd)



A

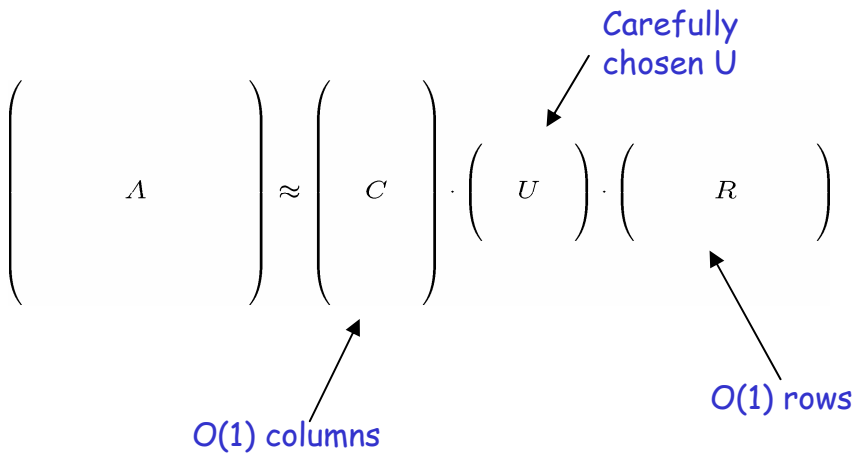


$H_k H_k^T A$

A and $H_k H_k^T A$ are close.

A novel CUR matrix decomposition

1. A "sketch" consisting of a few rows/columns of the matrix is adequate for efficient approximations.
2. Create an approximation to the original matrix of the following form:

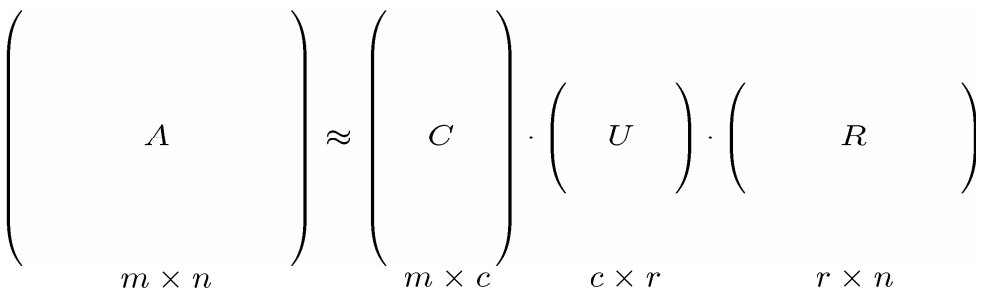


3. Given a query vector x, instead of computing A · x, compute CUR · x to identify its nearest neighbors.

$$\max_{x:|x|=1} \|Ax - CURx\| = \|A - CUR\|_2 \leq \epsilon \|A\|_F$$

The CUR decomposition

Given a **large** m-by-n matrix A (stored on disk), compute a decomposition CUR of A such that:



1. C consists of c = O(k/ε^2) columns of A.
2. R consists of r = O(k/ε^2) rows of A.
3. C (R) is created using importance sampling, e.g. columns (rows) are picked in i.i.d. trials with respect to probabilities

$$p_i = |A^{(i)}|^2 / \sum_i |A^{(i)}|^2$$

The CUR decomposition (cont'd)

Given a **large** m-by-n matrix A (stored on disk), compute a decomposition CUR of A such that:

4. C, U, R can be stored in O(m+n) space, after making **two** passes through the entire matrix A, using O(m+n) additional space and time.

5. The product CUR satisfies (with high probability)

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

$$\|A - CUR\|_2 \leq \epsilon\|A\|_F$$

Computing U

Intuition (which can be formalized):

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 a good fit for the row $A_{(i)}$ using the rows in R as the basis, by approximately solving

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

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

Notice that only c = O(1) element of the i-th row are given as input.

However, a vector of coefficients *u* can still be approximated.

Error bounds for CUR

Assume A_k is the "best" rank k approximation to A (through SVD). Then, if we pick $O(k/\varepsilon^2)$ rows and $O(k/\varepsilon^2)$ columns,

$$\|A - CUR\|_F^2 \leq \|A - A_k\|_F^2 + \varepsilon \|A\|_F^2$$

If we pick $O(1/\varepsilon^2)$ rows and $O(1/\varepsilon^2)$ columns,

$$\begin{aligned} \|A - CUR\|_2^2 &\leq \|A - A_k\|_2^2 + \varepsilon \|A\|_F^2 \\ &\leq \left(\frac{1}{k+1} + \varepsilon\right) \|A\|_F^2 \\ &\leq 2\varepsilon \|A\|_F^2 \end{aligned}$$

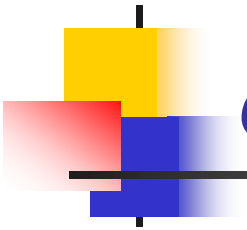
Other (randomized) CUR decompositions

Computing U in constant (O(1) instead of O(m+n)) space and time:

- (Drineas, Kannan, & Mahoney TR '04, SICOMP '05)
- samples O(poly(k,ε)) rows and columns of A & needs an extra pass.
- significantly improves the error bounds of Frieze, Kannan, and Vempala, FOCS '98.

Computing U and R for any C:

- (Drineas, Mahoney, and Muthukrishnan '05)
- For any subset of the columns, denoted C (e.g., chosen by the practitioner)
- Obtain bounds of the form: $|| A - CUR ||_F \leq (1 + \varepsilon) || A - CC^+ A ||_F$
- Uses ideas from approximating L2 Regression problems by random sampling.
- Can combine with recent algorithms/heuristics for choosing columns.



Other (non-randomized) CUR decompositions

Stewart:

- Compute sparse low-rank approximations to sparse matrices.
- Develop the quasi-Gram-Schmidt (variant of QR) algorithm:

  Input: m x n matrix A.

  Output: m x k matrix C (k columns of A) and upper-triangular k x k matrix S_C (that orthogonalizes these columns).
- Apply this algorithm to A and A^T: construct U to minimize $||A-CUR||_F^2$.

Goreinov, Tyrtyshnikov, and Zamarashkin:

- Scattering applications with large matrices with low-rank blocks.
- Relate to maximum volume concept in interpolation theory.
- They call the CUR decomposition the pseudoskeleton component of A.
- Provable error bounds for $||A-CUR||_2$.

Fast Computation with Kernels

Q. SVD has been used to identify/extract linear structure from data. What about non-linear structures, like multi-linear structures or non-linear manifold structure?

A. Kernel-based learning algorithms.

Data $\Psi = \{\Psi_{(1)}, \ldots, \Psi_{(m)}\} \in R^{m \times n}$

Mapping $\phi : \Psi \rightarrow \Phi$ (feature space)

Gram Matrix $G_{ij} = G\left(\Psi_{(i)}, \Psi_{(j)}\right) = \left\langle \phi\left(\Psi_{(i)}\right), \phi\left(\Psi_{(j)}\right) \right\rangle$

PSD matrix — inner product

Algorithms extracting linear structure can be applied to G without knowing ϕ !

Isomap, LLE, Laplacian Eigenmaps, SDE, are all Kernel PCA for special Gram matrices.

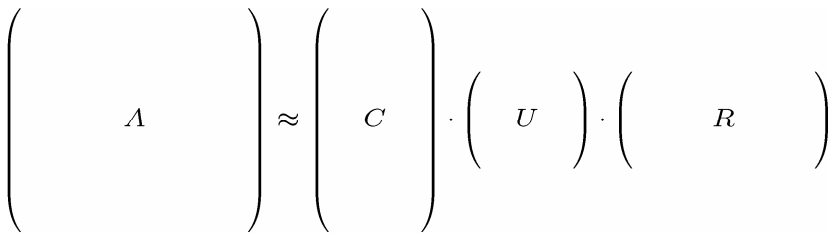
However, running, e.g., SVD to extract linear structure from the Gram matrix still requires $O(m^3)$ time.

We can **apply CUR decompositions** to speed up such calculations.

Fast Computation with Kernels (cont'd)

Note: The CUR decomposition of an SPSD matrix is not SPSD.

- Even if $R=C^T$, due to the form of U.



Goal: Obtain provable bounds for a $CW_k^+C^T$ decomposition.

- C consists of a small number of representative data points.
- W consists of the induced subgraph defined by those points.

$$\begin{pmatrix} & G & \end{pmatrix} \approx \begin{pmatrix} & \tilde{G} & \end{pmatrix} = \begin{pmatrix} C \end{pmatrix} \begin{pmatrix} W \end{pmatrix}^{+} \begin{pmatrix} & C^T & \end{pmatrix}$$

Fast Computation with Kernels (cont'd)

For an SPSD kernel matrix $G = XX^T$, if we use the "optimal" U ($=W_k^+$), then:

$$\|G - CUC^T\|_F^2 \leq \|G - G_k\|_F^2 + \epsilon\|X\|_F^4$$

$\|X\|_F^4$ vs. $\|G\|_F^2 = \|XX^T\|_F^2$

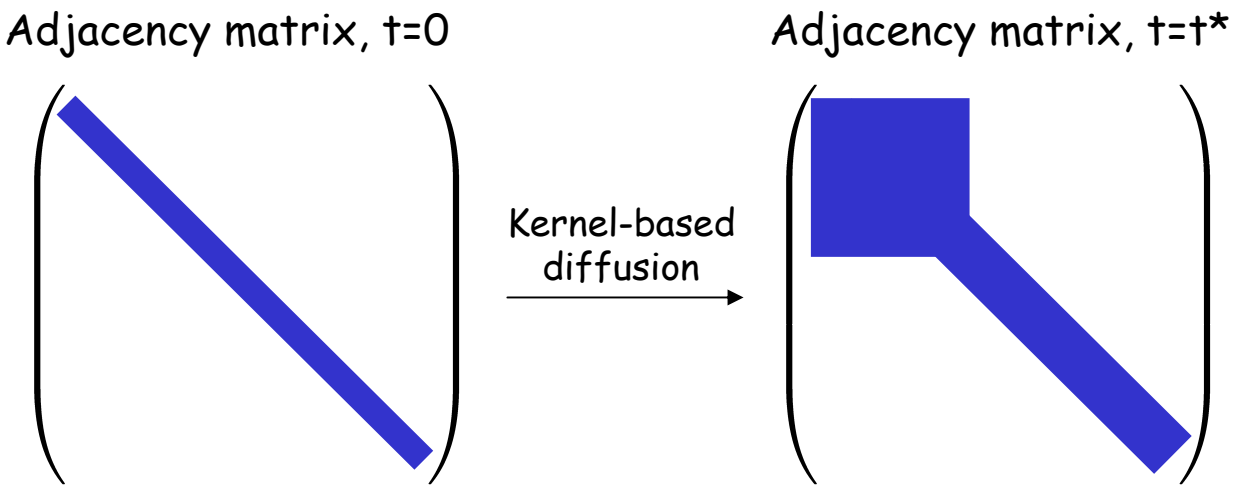
If the sampling probabilities were: $\mathbf{p_i} = \mathbf{||G^{(i)}||^2/||G||_F^2}$

- they would provide a bias towards data points that are more ``important'' - longer and/or more representative.
- the additional error would be $\varepsilon||G||_F$ and not $\varepsilon\ \Sigma_i\ G_{ii}^2 = \varepsilon||X||_F^2$.

Our (**provable**) sampling probabilities ignore correlations:

$$\mathbf{p_i = G_{ii}^2 / \Sigma_i\ G_{ii}^2 = ||X^{(i)}||^2/||X||_F^2}$$

Fast Computation with Kernels (cont'd)



To construct a coarse-grained version of the data graph:

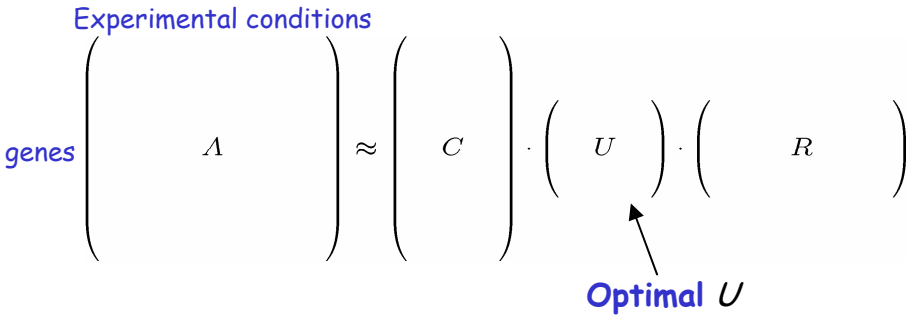
- Construct landmarks,
- Partition/Quantization,
- Diffusion wavelets.

To construct landmarks, randomly sample with the "right" probabilities:

- $p_i = |A^{(i)}|^2 / \|A\|_F^2$
- $p_i \sim 1/|A^{(i)}|$ for outliers,
- uniform sampling.

CUR and gene microarray data

Exploit **structural property of CUR** (or kernel-CUR) in biological applications:



Find a "good" set of genes and arrays to include in C and R?

Provable and/or heuristic strategies are acceptable.

Common in Biological/Chemical/Medical applications of PCA:

- Explain the singular vectors, by mapping them to meaningful biological processes.
- This is a "challenging" task (think: **reification**)!

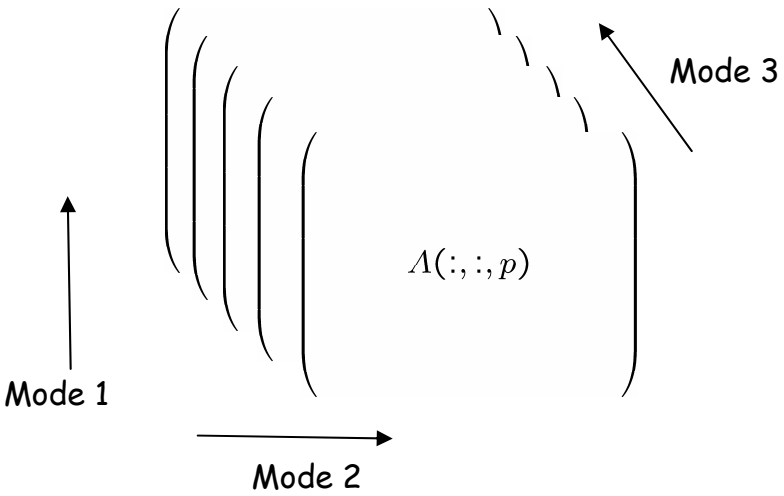
CUR is a low-rank decomposition in terms of the data that practitioners understand.

- Use it to explain the data and do dimensionality reduction, classification, clustering.

Gene microarray data: Mahoney, Drineas, and Alter (UT Austin) (sporulation and cell cycle data).

Datasets modeled as tensors

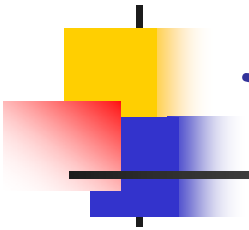
Goal: Extract structure from a tensor dataset A (naively, a dataset subscripted by multiple indices) using a small number of samples.



m x n x p tensor *A*

Q. What do we know about tensor decompositions?

A. Not much, although tensors arise in numerous applications.



Tensors in Applications

Tensors appear both in Math and CS.

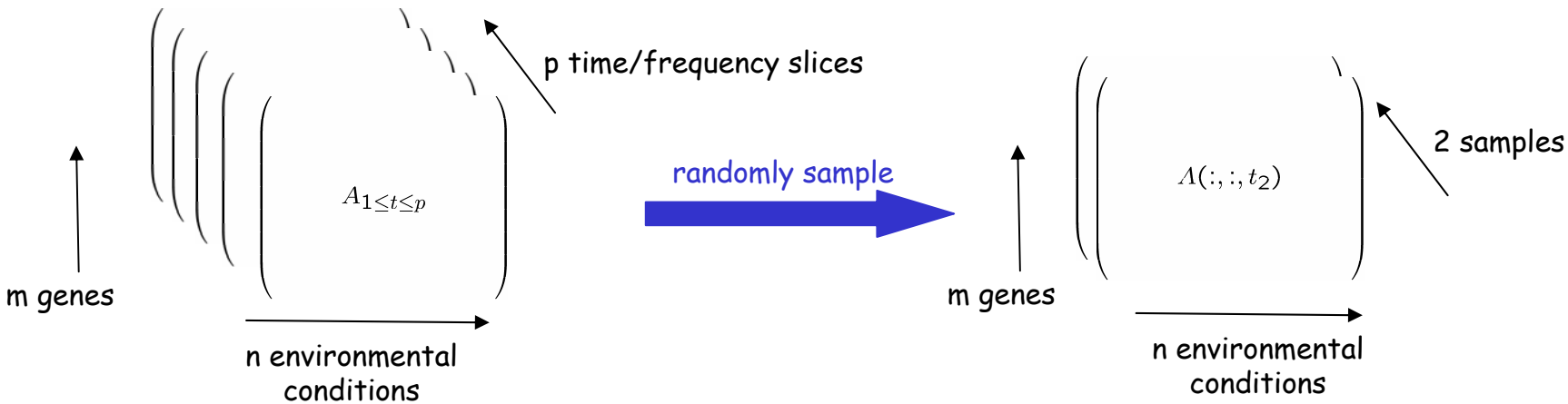
- Represent high dimensional functions.
- Connections to complexity theory (i.e., matrix multiplication complexity).
- Statistical applications (i.e., Independent Component Analysis, higher order statistics, etc.).
- Large data-set applications (e.g., Medical Imaging & Hyperspectral Imaging)

Problem: However, there does not exist a definition of tensor rank (and associated tensor SVD) with the – nice – properties found in the matrix case.

Heuristic solution: "unfold" the tensor along a mode and apply Linear Algebra.

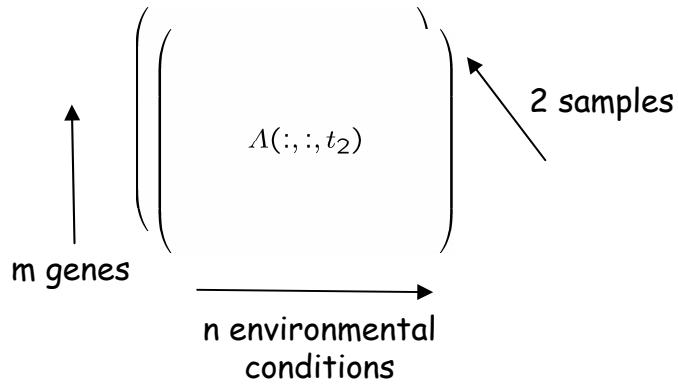
The TensorCUR algorithm (3-modes)

- Choose the preferred mode α (time)
- Pick a few representative snapshots: $p_t = \frac{\|A(:,:,t)\|_F^2}{\sum_{t=1}^{m} \|A(:,:,t)\|_F^2}$
- Express all the other snapshots in terms of the representative snapshots.



The TensorCUR algorithm (cont'd)

- Let R denote the tensor of the sampled snapshots.
- Express the remaining images as linear combinations of the sampled snapshots.

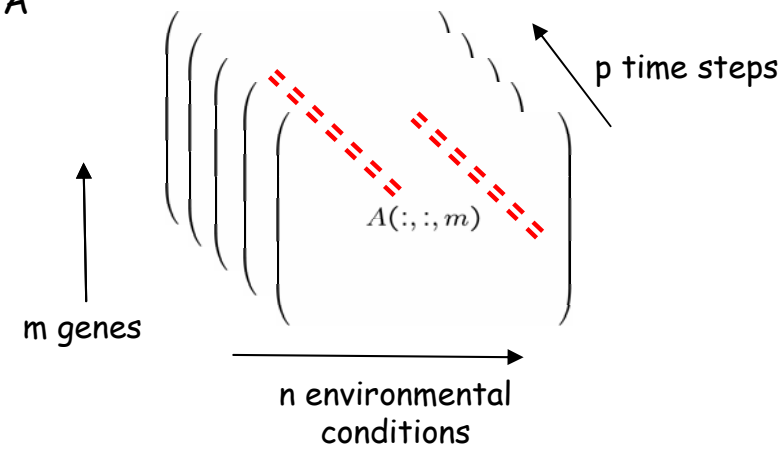


- First, pick a constant number of "fibers" of the tensor A (the red dotted lines).
- Express the remaining snapshots as linear combination of the sampled snapshots.

$$\min_u \sum_{i,j} \left(A(i,j,s) - \sum_{s \in R} u_s A(i,j,s)\right)^2$$

sampled fibers

sampled snapshots



The TensorCUR algorithm (cont'd)

Theorem: $\|A - CU \times_\alpha R\|_F^2 \leq \left\|A_{[\alpha]} - \left(A_{[\alpha]}\right)_{k_\alpha}\right\|_F^2 + \epsilon\|A\|_F^2$

Unfold R along the α dimension and pre-multiply by CU

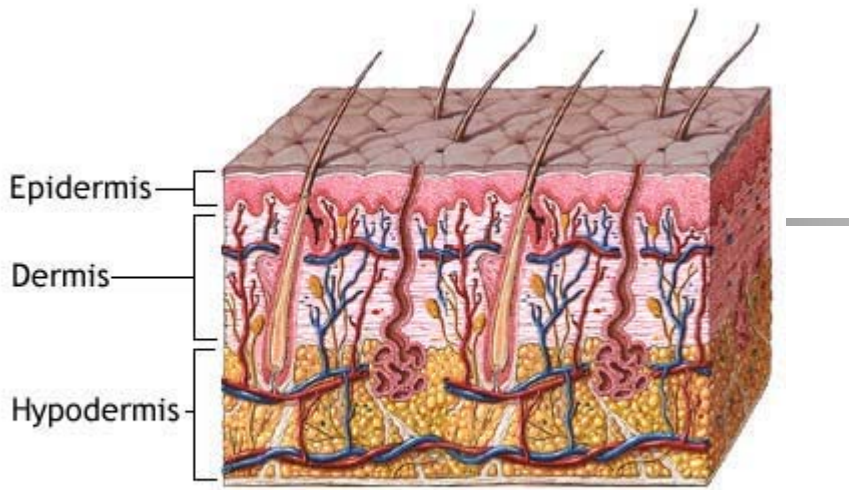
Best rank k_a approximation to $A_{[a]}$

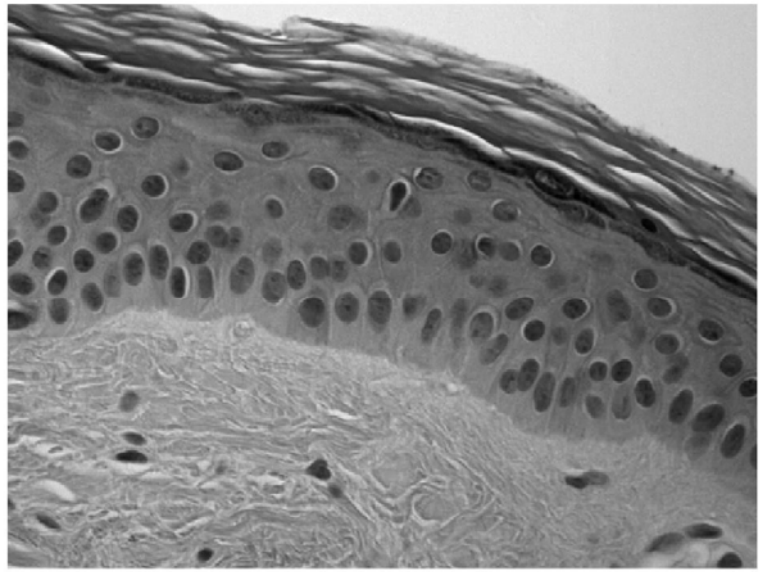
How to proceed:

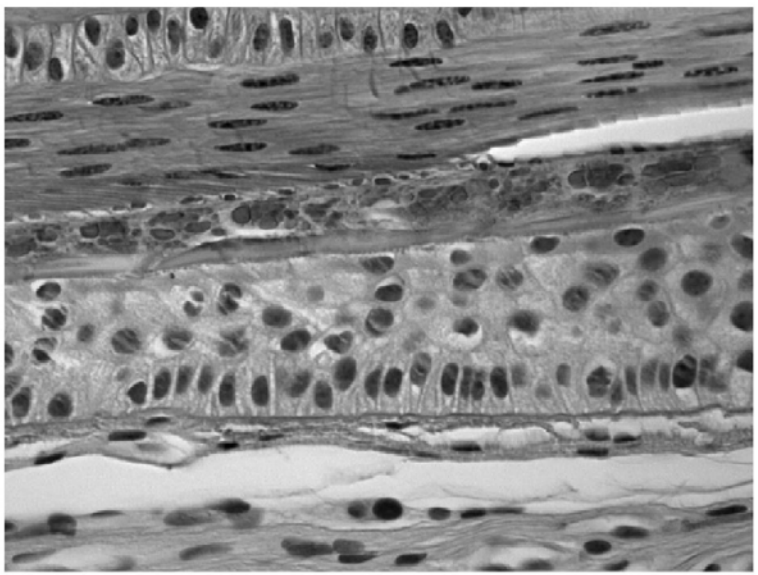
- Can recurse on each sub-tensor in R,
- or do SVD, exact or approximate,
- or do kernel-based diffusion analysis,
- or do wavelet-based methods.

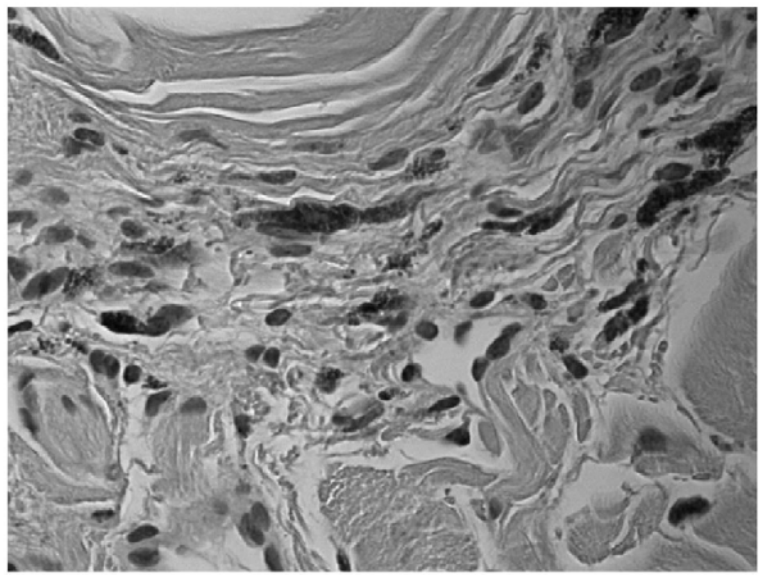
TensorCUR:

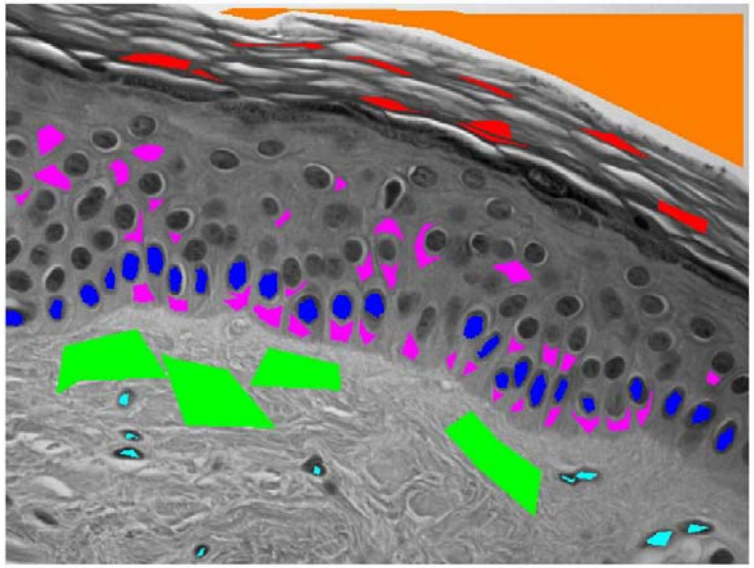
- Framework for dealing with very large tensor-based data sets,
- to extract a "sketch" in a principled and optimal manner,
- which can be coupled with more traditional methods of data analysis.

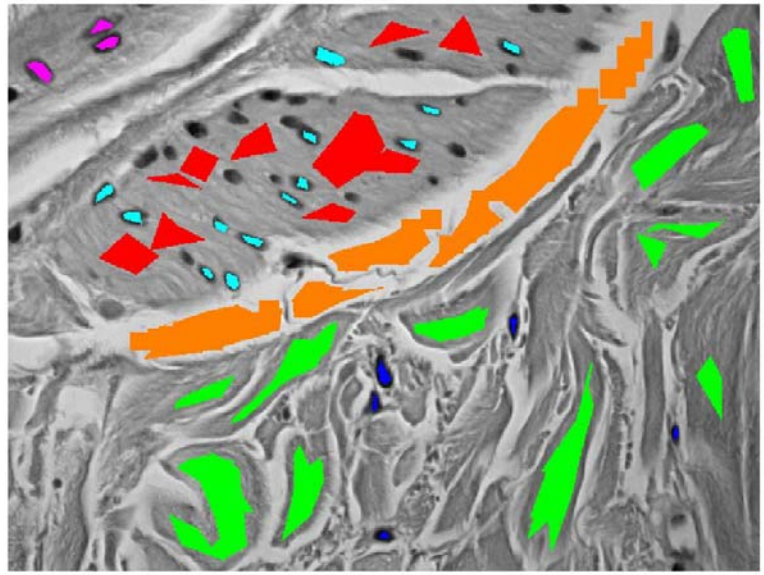

Epidermis
Dermis
Hypodermis

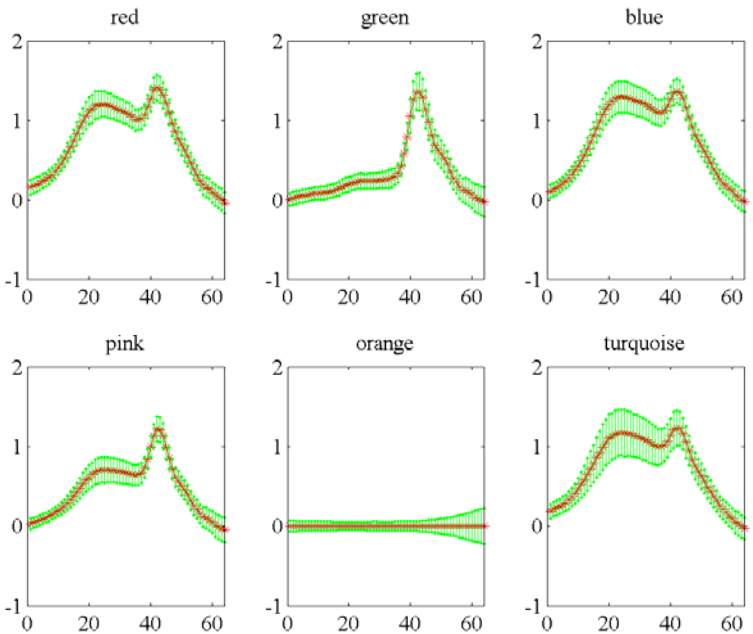


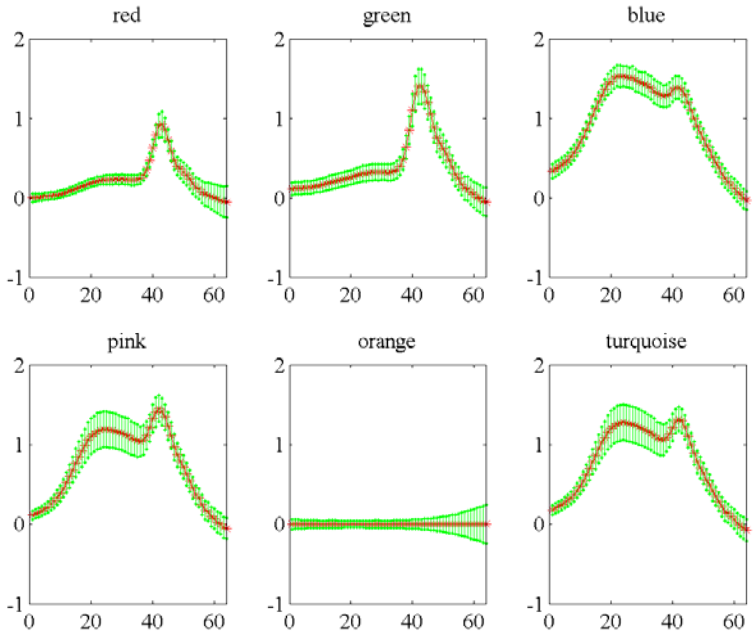


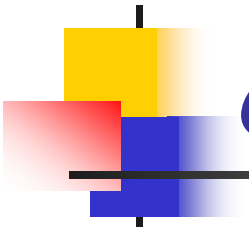







red
green
blue
pink
orange
turquoise
2
1
0
-1
0
20
40
60


red
green
blue
pink
orange
turquoise
2
1
0
-1
0
20
40
60



Coupling sampling with finer methods

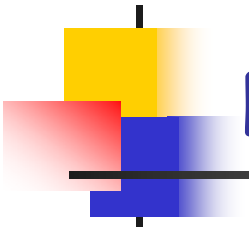
Apply random sampling methodology and kernel-based Laplacian methods to large physical and chemical and biological data sets.

Common application areas of large data set analysis:

- telecommunications,
- finance,
- web-based modeling, and
- astronomy.

Scientific data sets are quite different:

- with respect to their size,
- with respect to their noise properties, and
- with respect to the available field-specific intuition.



Data sets being considered

Sequence and mutational data from G-protein coupled receptors
- to identify mutants with enhanced stability properties,

Genomic microarray data
- to understand large-scale genomic and cellular behavior,

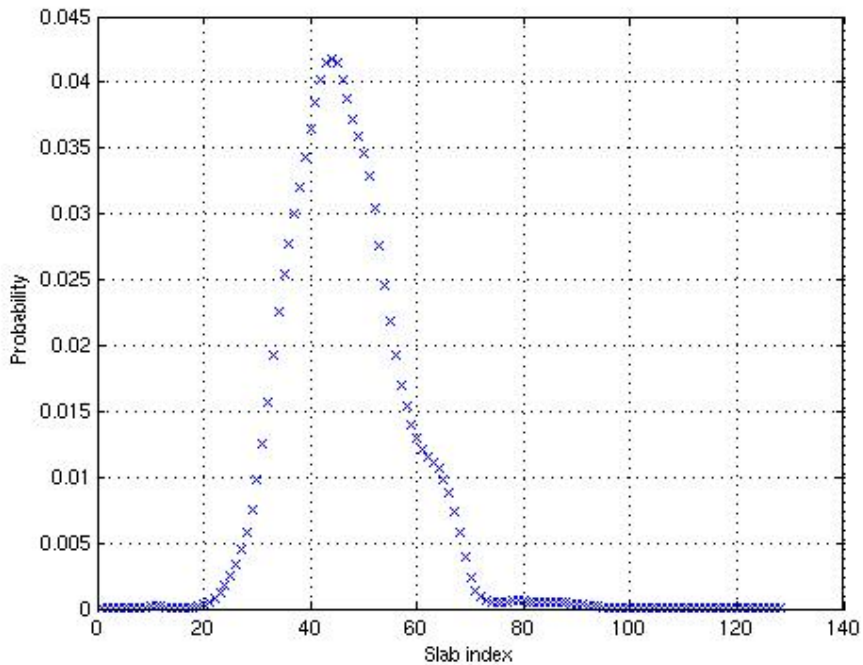
Hyperspectral colon cancer data
- for improved detection of anomalous behavior, and

Time-resolved fMRI data
- to better represent large, complex visual brain-imaging data, and

Simulational data
- to more efficiently conduct large scale computations.

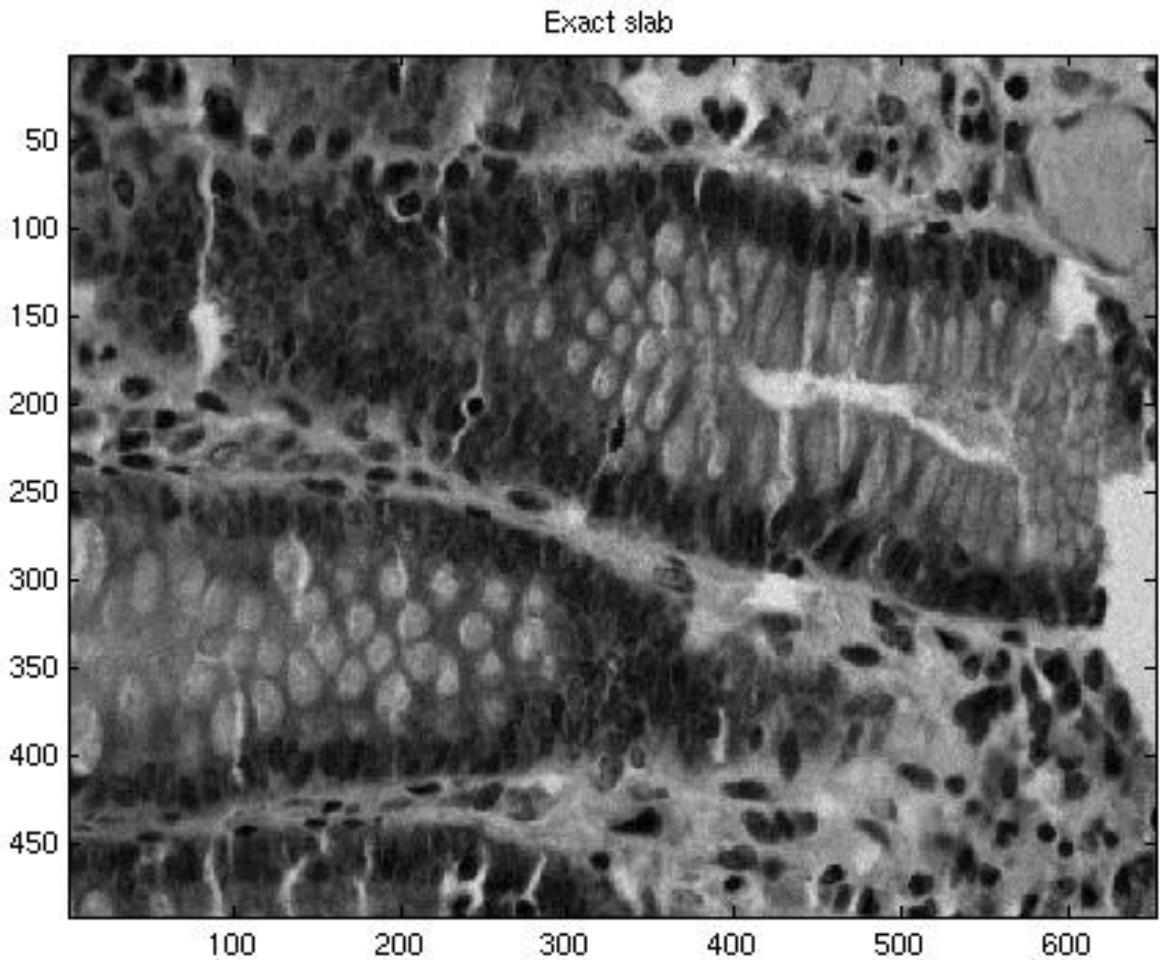
Sampling the hyperspectral data

Sample slabs depending on total absorbtion:

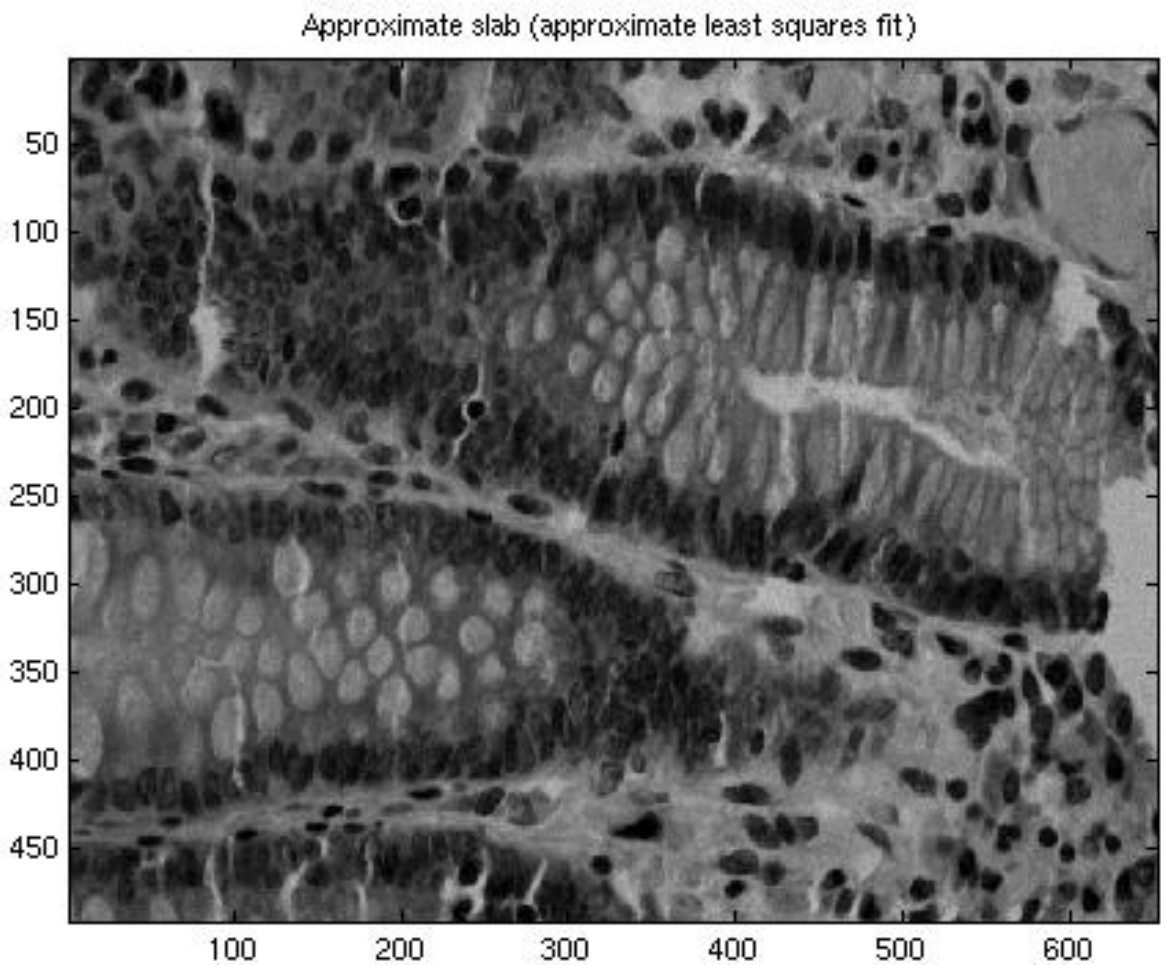


Sample fibers uniformly (since intensity depends on stain).

Look at the exact 65-th slab.

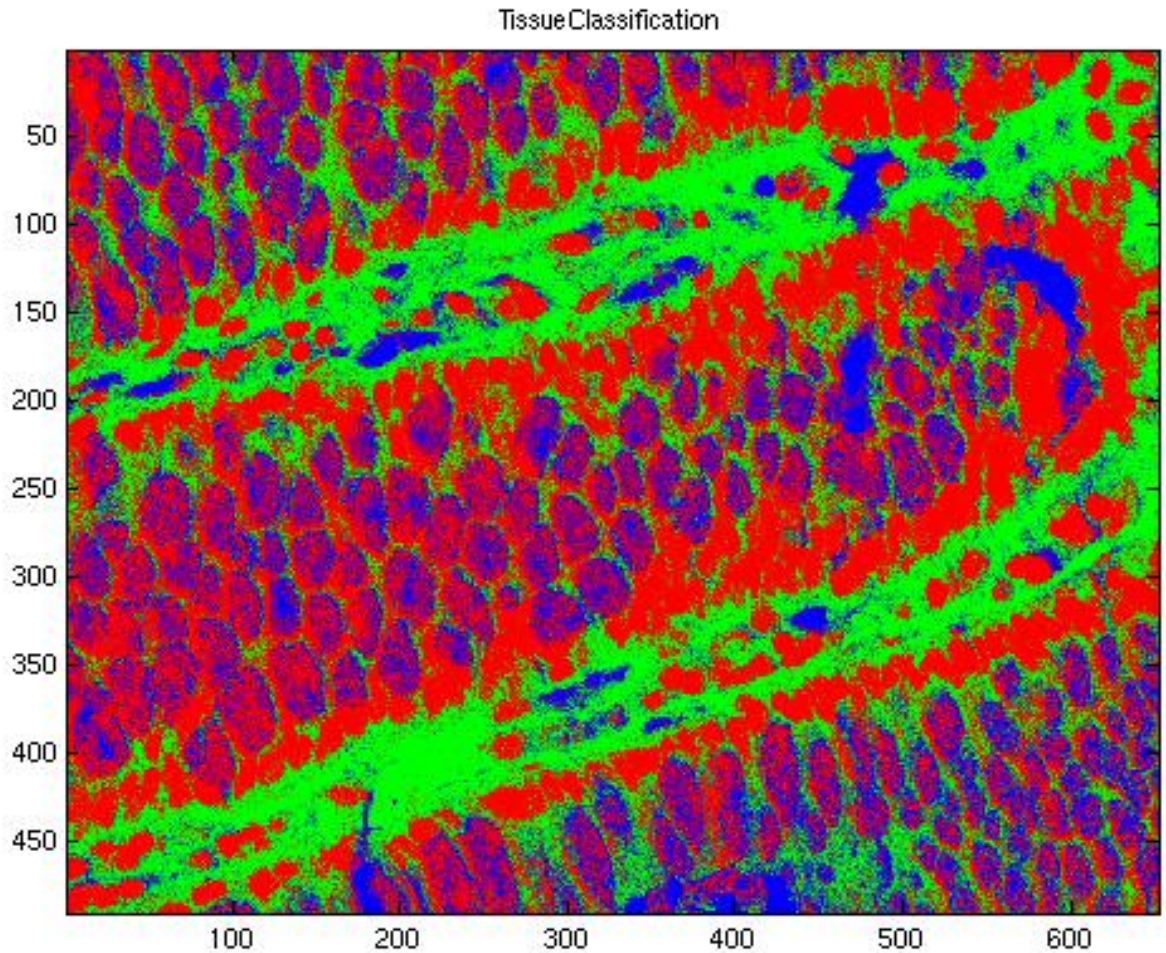


The 65-th slab approximately reconstructed

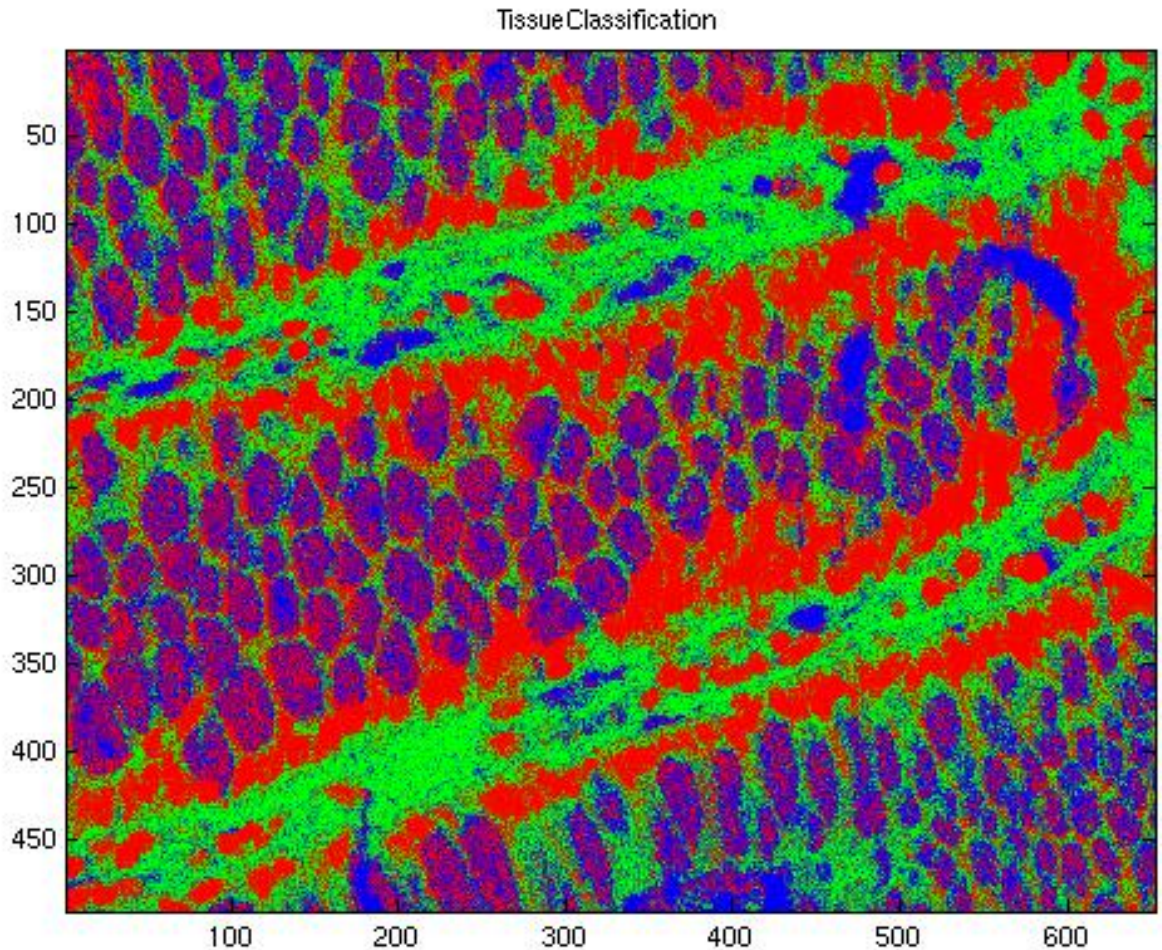


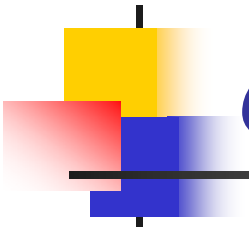
This slab was reconstructed by approximate least-squares fit to the basis from slabs 41 and 50, using 1000 (of 250K) pixels/fibers.

Tissue Classification - Exact Data



Tissue Classification - N_s=12 & N_f=1000





Overview and Summary

- Pass-Efficient Model and Random Sampling
- Matrix Multiplication
- Singular Value Decomposition
- Lower Bounds
- CUR Decomposition
- Kernel-based data sets and KernelCUR
- Tensor-based data sets and TensorCUR
- Large scientific (e.g., chemical and biological) data