Complexity of Matrix Computations discussion series, June 9, 2021

What does it mean to compute a low rank approximation of a matrix?

1. What is a low rank matrix?
2. What does approximation mean in this context?
3. What is a good algorithm for this problem?
   How should the running time depend on the approximation parameter?

Per-Gunnar Martinsson, University of Texas at Austin

Low rank approximation — a problem formulation

Given a matrix $\mathbf{A}$, find a matrix $\mathbf{B}$ of “low” rank such that $\dfrac{\|\mathbf{A}-\mathbf{B}\|}{\|\mathbf{A}\|}$ is “small”.

- Sometimes, the numerical rank of $\mathbf{A}$ is given in advance.
  More often, a *tolerance* is given, and determining the rank is part of the problem.

- Sometimes, it is important that the approximation is close to optimal, sometimes not.
  Is the cost to factorize the matrix a bottleneck, or not?
  If the matrix was expensive to build (say measured data), then prioritize optimality.
  If the purpose is to accelerate a computation, then prioritize speed.

What is a low rank matrix and what does approximation mean in this context?

Depends on the computational environment.

Matrices with rapid decay in singular spectrum. Common in “science and engineering”:
Reasonable to ask that $\|\mathbf{A} - \mathbf{B}\| < \varepsilon\|\mathbf{A}\|$, for ε *small*, say 10^{-5} or 10^{-10}.
Choice of norm is less important when singular values decay rapidly.

Matrices with slow decay in singular spectrum. Common in “data applications”:
$\varepsilon = 0.1$ often all you can ask for .
The choice of norm becomes important, in particular when matrix dimensions are high.

There are applications where $\|\mathbf{A} - \mathbf{B}\|$ does not even need to be small!
The answer could be, if I replace the original matrix **A** by some low rank matrix **B**,
then the algorithm still outputs a useful answer.
— Very challenging situation to analyze!

The notion of a singular value “gap” is less often useful in applications, I believe.

How do we measure “running time” of an algorithm?

Classical metrics in the literature:

- Number of flops.
- Number of matrix-vector multiplications.

Very cleanly defined and useful. Likely to stick around.

However, the correlation between classical metrics and “wall clock time” is weakening.

The reason: The ever tightening communication bottleneck.

- Multilevel memory structures (levels of cache / RAM / disk / network / ...).
- Heterogeneous architectures (GPUs and other accelerators).
- “Edge computing”, “cloud computing”, ... argh.

Various metrics for communication have been advanced. Useful and interesting!

However, landscape is complicated, so universally accurate metrics are elusive.

In many ways, this is an engineering problem.

My short term solution: Emphasize the matrix-matrix product.

What is a good algorithm for low rank approximation?

The kernel of the poodle!

What is a good algorithm for low rank approximation?

Excellent well established algorithms include:

- Various $O(n^3)$ complexity methods. Essentially optimal in terms of precision. Continual improvement in practical speed.
- Powering methods (Krylov, block Krylov, subspace iteration, etc.). Essentially optimal in terms of "accuracy per matvec".
- Pivoting based methods — column pivoted QR, LU with complete pivoting (cross approximation), etc.

What is a good algorithm for low rank approximation?

Excellent well established algorithms include:

- Various $O(n^3)$ complexity methods. Essentially optimal in terms of precision. Continual improvement in practical speed.
- Powering methods (Krylov, block Krylov, subspace iteration, etc.). Essentially optimal in terms of “accuracy per matvec”.
- Pivoting based methods — column pivoted QR, LU with complete pivoting (cross approximation), etc.

A natural answer in 2021: A new algorithm is good if it decisively outperforms (either faster, or closer to optimal, or both) existing methods.

- Better theoretical performance guarantees.
- Faster in practice. Actual applications, existing hardware.

What is a good algorithm for low rank approximation?

Excellent well established algorithms include:

- Various $O(n^3)$ complexity methods. Essentially optimal in terms of precision. Continual improvement in practical speed.
- Powering methods (Krylov, block Krylov, subspace iteration, etc.). Essentially optimal in terms of "accuracy per matvec".
- Pivoting based methods — column pivoted QR, LU with complete pivoting (cross approximation), etc.

A natural answer in 2021: A new algorithm is good if it decisively outperforms (either faster, or closer to optimal, or both) existing methods.

- Better theoretical performance guarantees.
- Faster in practice. Actual applications, existing hardware.

Randomized algorithms have in the past 10 – 20 years proven that they can do both.

What is a good algorithm for low rank approximation?

Randomized algorithms come with very different computational profiles!

Methods that compete directly with traditional techniques:

Techniques like the “randomized SVD” are reliable and robust.

Can come close to SVD in terms of optimality (at a cost, of course).

High practical speed; in particular when you accept some suboptimality.

Structured random maps — compelling theoretical results, *and* high practical speed (cannot be combined with powering, unfortunately).

Streaming (“single-view”) algorithms:

Can compute a low rank approximation in a single pass over the matrix.

Not possible with traditional algorithms.

Methods targeting problems outside the range of traditional techniques:

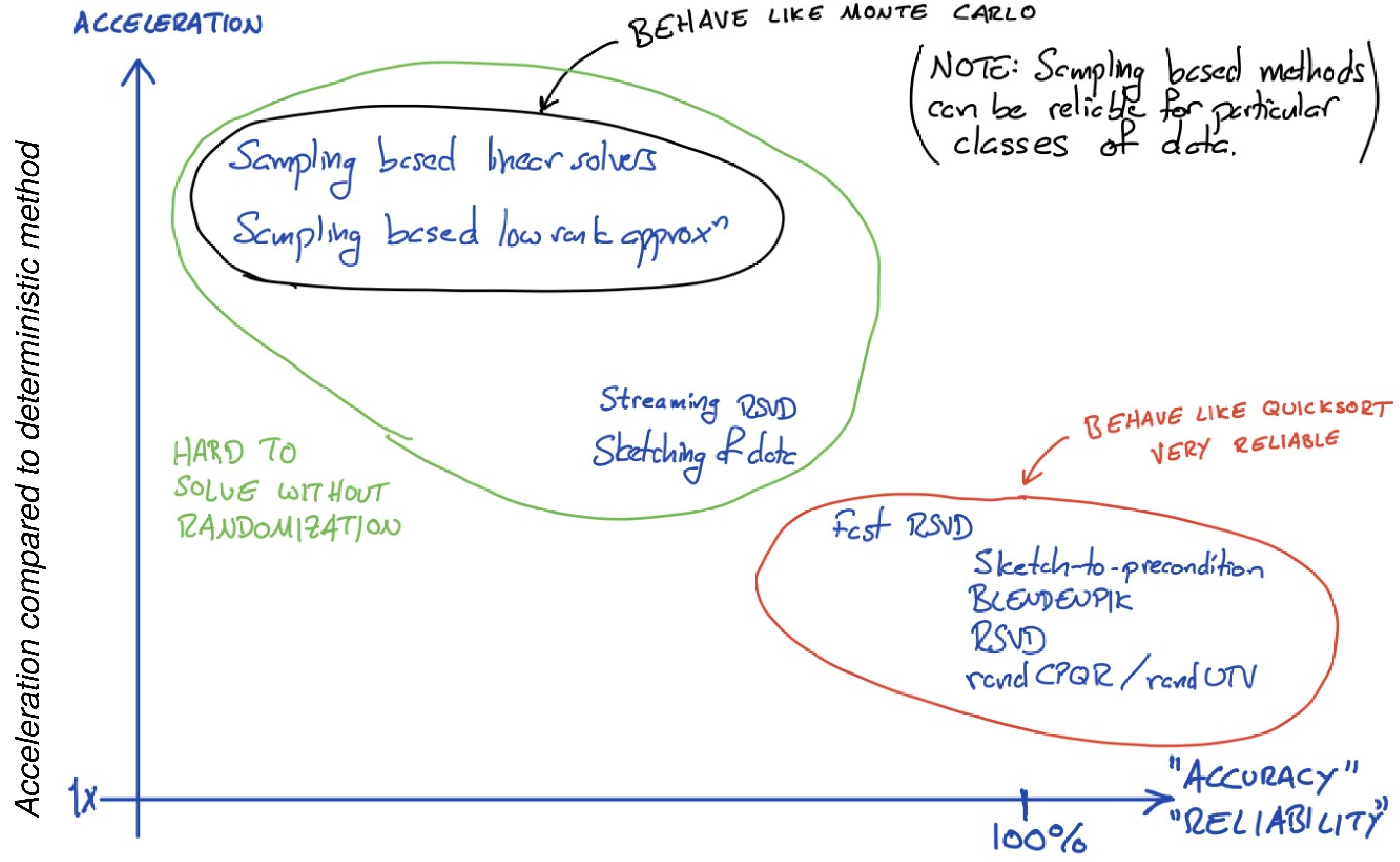
Randomization has enabled low rank approximation of humongous matrices.

Can be faster than matrix-vector multiplication.

Prototypical examples: Sampling based methods for matrices with low “coherence”.

Accuracy/reliability can suffer (behavior is similar to Monte Carlo methods).

Randomized algorithms come with different degrees of acceleration and reliability


ACCELERATION
Acceleration compared to deterministic method
BEHAVE LIKE MONTE CARLO
(NOTE: Sampling based methods can be reliable for particular classes of data.)
Sampling based linear solvers
Sampling based low rank approx^n
Streaming RSVD
Sketching of data
HARD TO SOLVE WITHOUT RANDOMIZATION
BEHAVE LIKE QUICKSORT VERY RELIABLE
fast RSVD
Sketch-to-precondition
BLENDENPIK
RSVD
rand CPQR / rand UTV
1x
100%
"ACCURACY"
"RELIABILITY"

Accuracy or reliability compared to deterministic method

This graphic is not to be taken *too* seriously…