Randomized algorithms for low rank matrix approximation

Laura Grigori

EPFL and PSI

November 5/12, 2024

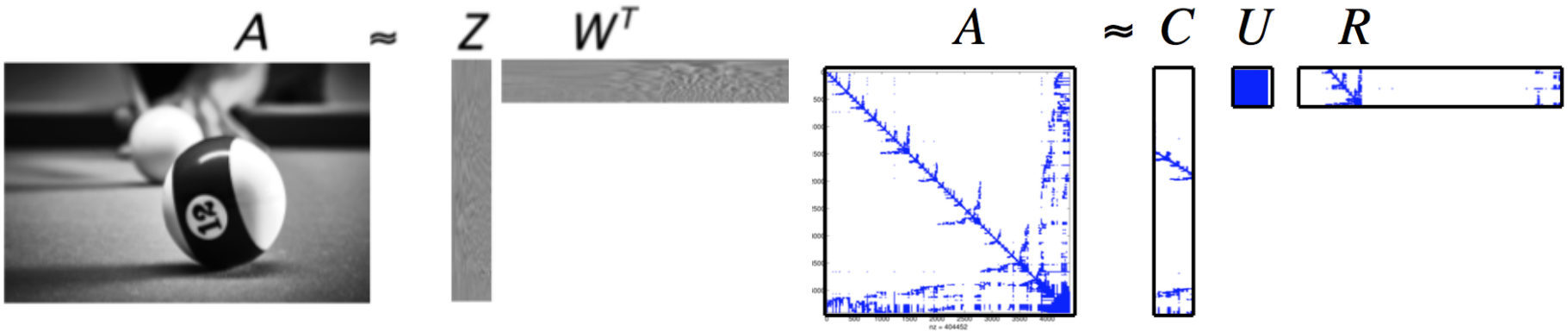
Plan

Low rank matrix approximation

Randomized algorithms for low rank approximation

Low rank matrix approximation

- Problem: given $A \in \mathbb{R}^{m \times n}$, compute rank-k approximation ZW^T, where Z is $m \times k$ and W^T is $k \times n$.



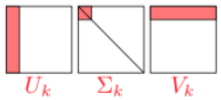
- Problem with diverse applications
  - from scientific computing: fast solvers for integral equations, H-matrices
  - to data analytics: principal component analysis, image processing, ...

$$Ax \to ZW^T x$$

$$\textit{Flops} \quad 2mn \to 2(m+n)k$$

Low rank matrix approximation

- Best rank-k approximation $[\![A]\!]_k = U_k \Sigma_k V_k^T$ is rank-k truncated SVD of A [Eckart and Young, 1936], where U_k, Σ_k, V_k are the first k left singular vectors, leading singular values, right singular vectors respectively,



$$\min_{rank(A_k) \leq k} ||A - A_k||_2 = ||A - [\![A]\!]_k||_2 = \sigma_{k+1}(A) \quad (1)$$

$$\min_{rank(A_k) \leq k} ||A - A_k||_F = ||A - [\![A]\!]_k||_F = \sqrt{\sum_{j=k+1}^{n} \sigma_j^2(A)} \quad (2)$$

Image, size 1190×1920

Rank-10 approximation, SVD

Rank-50 approximation, SVD

- Image source: `https://pixabay.com/photos/billiards-ball-play-number-half-4345870/`

Large data sets

Problems to solve

- Compute low rank approximation of A
- Select a subset of columns of A

Constraints

- Matrix A might not exist entirely at a given time, rows or columns are added progressively.
  - Streaming algorithm: can solve an arbitrarily large problem with one pass over the data (a row or a column at a time).
  - Weakly streaming algorithm: can solve a problem with $O(1)$ passes over the data.
- Matrix A might exist only implicitly, and it is never formed explicitly, e.g. $A = BC$.

Large data sets

Problems to solve

- Compute low rank approximation of A
- Select a subset of columns of A

Constraints

- Matrix A might not exist entirely at a given time, rows or columns are added progressively.
  - Streaming algorithm: can solve an arbitrarily large problem with one pass over the data (a row or a column at a time).
  - Weakly streaming algorithm: can solve a problem with $O(1)$ passes over the data.
- Matrix A might exist only implicitly, and it is never formed explicitly, e.g. $A = BC$.

Goal

Compute efficiently a low rank-k approximation A_k of A satisfying

$$\|A - A_k\|_2 \leq \gamma \sigma_{k+1}(A)$$

for some $\gamma \geq 1$, γ typically a low degree polynomial in k and dimensions of A

- Reduce flops and communication

Properties of the approximations

Definitions and some of the results taken from [Demmel et al., 2023].

Definition 1

[low-rank approximation] A matrix A_k satisfying $\|A - A_k\|_2 \leq \gamma\sigma_{k+1}(A)$ for some $\gamma \geq 1$ will be said to be a (k, γ) *low-rank approximation of* A.

Definition 2

[spectrum preserving] If A_k satisfies

$$\sigma_j(A) \geq \sigma_j(A_k) \geq \gamma^{-1}\sigma_j(A)$$

for $j \leq k$ and some $\gamma \geq 1$, it is a (k, γ) *spectrum preserving*.

Definition 3

[kernel approximation] If A_k satisfies

$$\sigma_{k+j}(A) \leq \sigma_j(A - A_k) \leq \gamma\sigma_{k+j}(A)$$

for $1 \leq j \leq n - k$ and some $\gamma \geq 1$, it is a (k, γ) *kernel approximation of* A.

Idea underlying many algorithms

Compute $A_k = \mathcal{P}A$, where $\mathcal{P} = \mathcal{P}^o$ or $\mathcal{P} = \mathcal{P}^{ob}$ is obtained as:

1. Construct a low dimensional subspace $X = range(A\Omega)$, $\Omega \in \mathbb{R}^{n \times l}$ that approximates well the range of A, e.g.

$$\|A - \mathcal{P}^o A\|_2 \leq \gamma \sigma_{k+1}(A), \text{ for some } \gamma \geq 1,$$

where Q is orth. basis of $(A\Omega)$ and orthogonal projector:

$$\mathcal{P}^o A = A\Omega(A\Omega)^+ A = QQ^T A, \text{ or equiv } \mathcal{P}^o a_j := arg \min_{x \in X} \|x - a_j\|_2$$

2. Select a semi-inner product $\langle \Theta \cdot, \Theta \cdot \rangle_2$, $\Theta \in \mathbb{R}^{l' \times m}$ $l' \geq l$, define (oblique projector):

$$\mathcal{P}^{ob} A = A\Omega(\Theta A\Omega)^+ \Theta A, \text{ or equiv } \mathcal{P}^{ob} a_j := arg \min_{x \in X} \|\Theta(x - a_j)\|_2$$

Idea underlying many algorithms

Compute $A_k = \mathcal{P}A$, where $\mathcal{P} = \mathcal{P}^o$ or $\mathcal{P} = \mathcal{P}^{ob}$ is obtained as:

1. Construct a low dimensional subspace $X = range(A\Omega)$, $\Omega \in \mathbb{R}^{n \times l}$ that approximates well the range of A, e.g.

$$\|A - \mathcal{P}^o A\|_2 \leq \gamma \sigma_{k+1}(A), \text{ for some } \gamma \geq 1,$$

where Q is orth. basis of $(A\Omega)$ and orthogonal projector:

$$\mathcal{P}^o A = A\Omega(A\Omega)^+ A = QQ^T A, \text{ or equiv } \mathcal{P}^o a_j := arg \min_{x \in X} \|x - a_j\|_2$$

2. Select a semi-inner product $\langle \Theta \cdot, \Theta \cdot \rangle_2$, $\Theta \in \mathbb{R}^{l' \times m}$ $l' \geq l$, define (oblique projector):

$$\mathcal{P}^{ob} A = A\Omega(\Theta A\Omega)^+ \Theta A, \text{ or equiv } \mathcal{P}^{ob} a_j := arg \min_{x \in X} \|\Theta(x - a_j)\|_2$$

Low rank approximation and orthogonal projector

Given $A = U\Sigma V^T$, let U_k, Σ_k, V_k be the first k left singular vectors, leading singular values, right singular vectors respectively. Then the best approximation is when $Q = U_k$:

$$\begin{aligned} QQ^T A &= U_k U_k^T U\Sigma V^T \\ ||A - QQ^T A||_2 &= ||diag(0, \ldots, 0, \sigma_{k+1}, \ldots \sigma_n)||_2 = \sigma_{k+1} \end{aligned}$$

Plan

Low rank matrix approximation

Randomized algorithms for low rank approximation

- Randomized SVD
- Randomized Nyström and generalized Nyström approximation
- Parallelism and numerical experiments

Randomized algorithms - main idea

- Construct a low dimensional subspace that captures the action of A.
- Restrict A to the subspace and compute a standard QR or SVD factorization.

Obtained as follows:

1. Compute an approximate basis for the range of A $(m \times n)$
   find Q $(m \times k)$ with orthonormal columns and approximate A by the projection of its columns onto the space spanned by Q:

$$A \approx QQ^T A$$

2. Use Q to compute a standard factorization of A

Good sources for additional information: [Halko et al., 2011, Martinsson and Tropp, 2020]

Typical randomized SVD

1. Compute an approximate basis for the range of $A \in \mathbb{R}^{m \times n}$
   Sample $\Omega \in \mathbb{R}^{n \times l}$, $l = p + k$, with independent mean-zero, unit-variance Gaussian entries.
   Compute $Y = A\Omega$, $Y \in \mathbb{R}^{m \times l}$ expected to span column space of A.
   - Cost of multiplying $A\Omega$: $2mnl$ flops and $O(\log_2 P)$ messages
2. With Q being orthonormal basis of Y, approximate A as:

$$A_{\text{RSVD}} = QQ^T A = \mathcal{P}^o A$$

   - Cost of multiplying $Q^T A$: $2mnl$ flops and $O(\log_2 P)$ messages
   - randomized SVD relies on an orthogonal projection

Source: Halko et al, *Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decomposition*, SIREV 2011.

Typical randomized SVD

Algorithm

Input: matrix $A \in \mathbb{R}^{m \times n}$, desired rank k, $l = p + k$, q.

1. Sample an $n \times l$ test matrix Ω with independent mean-zero, unit-variance Gaussian entries.
2. Compute $Y = (AA^T)^q A\Omega$ /* Y is expected to span the column space of A */
3. Construct $Q \in \mathbb{R}^{m \times l}$ with columns forming an orthonormal basis for the range of Y.
4. Compute $B = Q^T A$, $B \in \mathbb{R}^{l \times n}$
5. Compute the rank-k truncated SVD of B as $\hat{U}\Sigma V^T$, $\hat{U} \in \mathbb{R}^{l \times k}$, $V^T \in \mathbb{R}^{k \times n}$

Return the approximation $[\![A_{\mathrm{RSVD}}]\!]_k = Q\hat{U} \cdot \Sigma \cdot V^T$

See e.g. Algorithm 8 (relying on rangefinder with powering described in section 11.6.1) from [Martinsson and Tropp, 2020]

Randomized SVD ($q = 0$)

Theorem 1.1 from Halko et al. If Ω is chosen to be i.i.d. N(0,1), $k, p \geq 2$, $q = 1$, then the expectation with respect to the random matrix Ω is

$$\mathbb{E}(||A - QQ^T A||_2) \leq \left(1 + \frac{4\sqrt{k+p}}{p-1}\sqrt{min(m,n)}\right)\sigma_{k+1}(A)$$

and the probability that the error satisfies

$$||A - QQ^T A||_2 \leq \left(1 + 11\sqrt{k+p} \cdot \sqrt{min(m,n)}\right)\sigma_{k+1}(A)$$

is at least $1 - 6/p^p$.

For $p = 6$, the probability becomes .99.

Randomized SVD

Theorem 10.6, Halko et al. Average spectral norm. Under the same hypotheses as Theorem 1.1 from Halko et al.,

$$\mathbb{E}(||A - QQ^T A||_2) \leq \left(1 + \sqrt{\frac{k}{p-1}}\right) \sigma_{k+1}(A) + \frac{e\sqrt{k+p}}{p} \left(\sum_{j=k+1}^{n} \sigma_j^2(A)\right)^{1/2}$$

- Fast decay of singular values:
  If $\left(\sum_{j>k} \sigma_j^2(A)\right)^{1/2} \approx \sigma_{k+1}$ then the approximation should be accurate.
- Slow decay of singular values:
  If $\left(\sum_{j>k} \sigma_j^2(A)\right)^{1/2} \approx \sqrt{n-k}\sigma_{k+1}$ and n large, then the approximation might not be accurate.

Source: G. Martinsson's talk

Power iteration $q \geq 1$

The matrix $(AA^T)^q A$ has a faster decay in its singular values:

- has the same left singular vectors as A
- its singular values are:

$$\sigma_j((AA^T)^q A) = (\sigma_j(A))^{2q+1}$$

Cost of randomized truncated SVD

- Randomized SVD requires $2q + 1$ passes over the matrix.

- The last 4 steps of the algorithm cost:
  (2) Compute $Y = (AA^T)^q A\Omega$: $2(2q + 1) \cdot mn \cdot (k + p)$
  (3) Compute QR of Y: $2m(k + p)^2$
  (4) Compute $B = Q^T A$: $2mn \cdot (k + p)$
  (5) Compute SVD of B: $O(n(k + p)^2)$

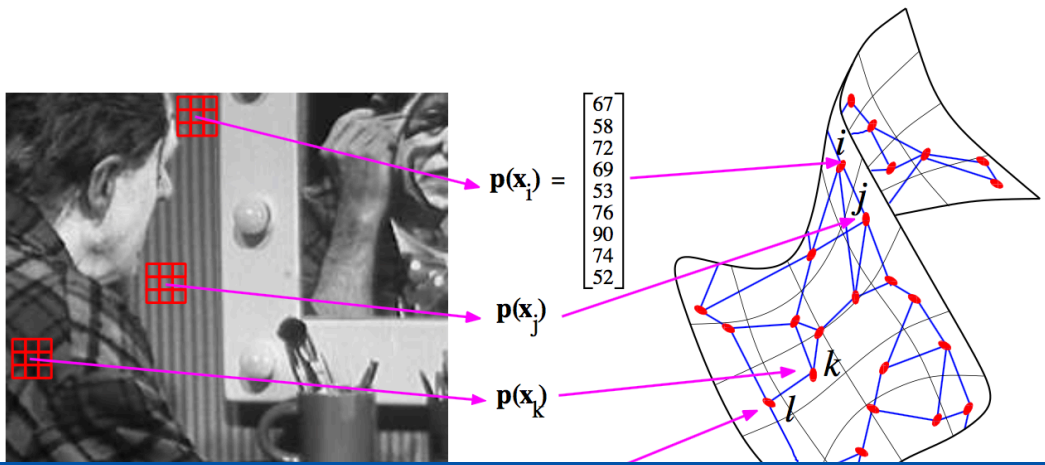
- If $n \geq k + p$ and $q = 1$, then (2) and (4) dominate (3).
- To have a smaller arithmetic cost than deterministic approaches, the cost of (2) and (4) need to be reduced.
- (2) can be reduced by using a fast Johnson-Lindenstrauss transform
- (4) can be reduced by using an oblique projection

Results from image processing (from Halko et al)

- A matrix A of size 9025×9025 arising from a diffusion geometry approach.
- A is a graph Lapacian on the manifold of 3×3 patches.
- 95×95 pixel grayscale image, intensity of each pixel is an integer ≤ 4095.
- Vector $x^{(i)} \in \mathbb{R}^9$ gives the intensities of the pixels in a 3×3 neighborhood of pixel i.
- W reflects similarities between patches, $\sigma = 50$ reflects the level of sensitivity,

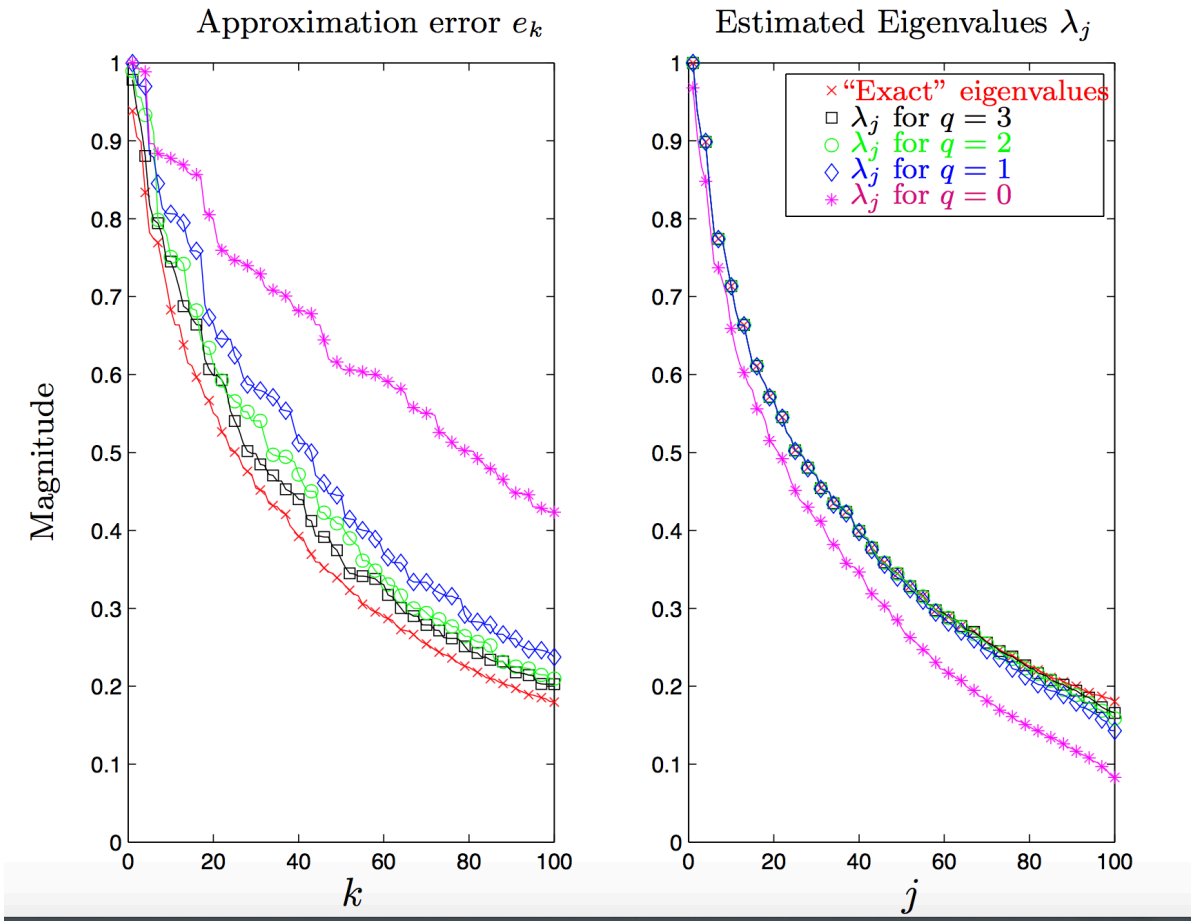
$$w_{ij} = exp\{-||x^{(i)} - x^{(j)}||^2/\sigma^2\},$$

- Sparsify W, compute dominant eigenvectors of $A = D^{-1/2}WD^{-1/2}$.



Experimental results (from Halko et al)

- Approximation error : $||A - QQ^T A||_2$
- Estimated eigenvalues for $k = 100$



Randomized Nyström approximation

Goal: derive an algorithm that uses one pass over the data

For $A \in \mathbb{R}^{n \times n}$ symmetric positive semidefinite (SPSD), sketching $\Omega \in \mathbb{R}^{n \times l}$, randomized Nyström approximation computes

$$A_{Nyst} = (A\Omega)(\Omega^T A\Omega)^+(\Omega^T A) \tag{3}$$

where $(\Omega^T A\Omega)^+$ denotes the pseudoinverse of $\Omega^T A\Omega$

Randomized Nyström Algorithm

Two solutions possible for computing a rank-k approximation from equation (3)

1. compute a rank-k truncation of the Nyström approximation A_{Nyst}
2. compute a rank-k truncation of the matrix $B = \Omega^T A \Omega$

Randomized Nyström Algorithm

- Several solutions possible for computing $B^+ = (\Omega^T A \Omega)^+$ and obtaining a rank-k decomposition
- Since $B = \Omega^T A \Omega$ is SPSD, one can use its truncated eigenvalue decomposition, or Cholesky factorization if B is SPD (symmetric positive definite).
- Special care required if $B = \Omega^T A \Omega$ is badly conditioned or rank deficient, when e.g. A has rank less than l. For one solution see e.g. [Tropp et al., 2017a] or these slides, where the Cholesky factorization of B has to be replaced with a more stable decomposition

Randomized Nyström - rank-k truncation of A_{Nyst}

Let $C = A\Omega$ and $B = \Omega^T A\Omega$, suppose the Cholesky factorization of B exists, then compute:

$$\begin{aligned} B &= \Omega^T A\Omega = LL^T \text{ Cholesky factorization} \\ Z &= CL^{-T} = QR \quad \text{QR factorization} \\ R &= U_k \Sigma_k V_k^T \quad \text{truncated rank-k SVD} \end{aligned}$$

We obtain the factorization:

$$\begin{aligned} (A\Omega)(\Omega^T A\Omega)^+(\Omega^T A) &= CL^{-T}L^{-1}C^T = ZZ^T \\ &= QRR^TQ^T \\ [\![A_{Nyst}]\!]_k &= QU_k\Sigma_k\Sigma_k U_k^T Q^T \\ &= \hat{U}_k \Sigma_k^2 \hat{U}_k^T, \text{ where} \\ \hat{U}_k &= QU_k = CL^{-T}R^{-1}U_k = ZV_k\Sigma_k^{-1} \end{aligned}$$

Remarks:

- $\hat{U}_k$ could be computed more stably as QU_k or more efficiently as $ZV_k\Sigma_k^{-1}$
- If B is rank defficient, replace Cholesky factorization with more stable truncated decomposition (eigenvalue/SVD)

Randomized Nyström (contd)

Using derivations from previous slide, the algorithm becomes:

Algorithm Randomized Nyström with rank-k truncation of A_{Nyst}
Input $A \in \mathbb{R}^{n \times n}$, $\Omega \in \mathbb{R}^{n \times l}$:

1. Compute $C = A\Omega$, $C \in \mathbb{R}^{n \times l}$
2. Compute $B = \Omega^T C$, $B \in \mathbb{R}^{l \times l}$ and its Cholesky factorization $B = LL^T$ (if this fails, one could use the eigenvalue decomposition of B)
3. Compute $Z = CL^{-T}$ with substitution (no explicit computation of the inverse of L^T)
4. Compute the QR factorization $Z = QR$
5. Compute truncated rank-k SVD of R as $U_k \Sigma_k V_k^T$
6. Compute $\hat{U}_k = QU_k$
7. Output factorization $[\![A_{Nyst}]\!]_k = \hat{U}_k \Sigma_k^2 \hat{U}_k^T$

Parallel randomized Nyström

Example of parallelization for computing $C = A\Omega$ and $B = \Omega^T A\Omega$

- Sketching matrix Ω (see previous lecture): consider the case when blocks of Ω can be generated on each processor
- Consider Ω distributed block row-wise over $\sqrt{P}$ processors and A distributed over $\sqrt{P} \times \sqrt{P}$ processors, that is:

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}, \quad \Omega = \begin{pmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{pmatrix}$$

- Compute:

$$C = \begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{pmatrix}, \quad B = \Omega^T A\Omega$$

Parallel randomized Nyström

Compute in parallel $C = A\Omega$, $B = \Omega^T A\Omega$, eg for B

$$B = \Omega^T A\Omega = \begin{pmatrix} \Omega_1^T & \Omega_2^T & \Omega_3^T \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{pmatrix}$$

Computation of $C = A\Omega$, $B = \Omega^T A\Omega$

Root processor broadcasts information for generating blocks of Ω
Let P_{ij} by my processor number that owns block A_{ij}
for all processors P_{ij}, $i = 1 : \sqrt{P}, j = 1 : \sqrt{P}$ in parallel **do**
  Generate blocks Ω_i, Ω_j of Ω
  Compute $C_{ij} = A_{ij}\Omega_j$
  Sum-reduce to compute $C_i = \sum_{j=1}^{\sqrt{P}} C_{ij}$ among procs in same row
  Compute $B_{ij} = \Omega_i^T A_{ij}\Omega_j$
end for
Sum-reduce to compute $B = \sum_{i,j=1}^{\sqrt{P},\sqrt{P}} B_{ij}$

Communication cost: $3log_2 P\alpha + (l^2 + ln/\sqrt{P}) \log_2 P\beta$

Application to kernel methods

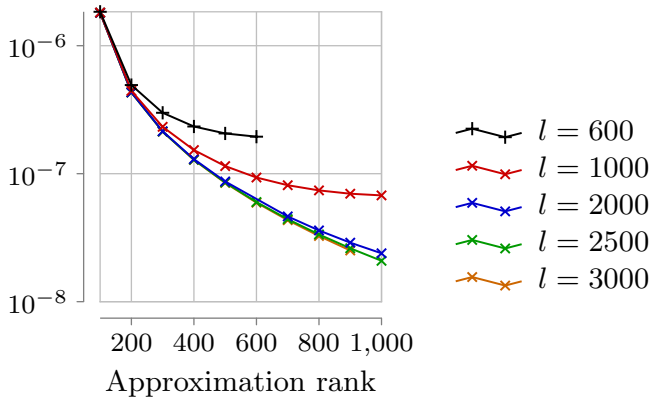
- Consider kernel methods that map data points into a feature space
- Highly used in learning as classification and regression
- Given $X = [x_1, \ldots, x_n]$, $X \in \mathbb{R}^{m \times n}$, each column $x_i, i = 1 \ldots m$, is a data point in $\mathbb{R}^m$
- Kernel based learning maps input data points to a feature space
- Inner products in feature space computed with a nonlinear kernel function $\kappa(\cdot, \cdot)$, used to build a SPSD kernel matrix $K \in \mathbb{R}^{n \times n}$ whose elements are defined as

$$K_{i,j} = \kappa(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle, i, j = 1, \ldots, n \tag{4}$$

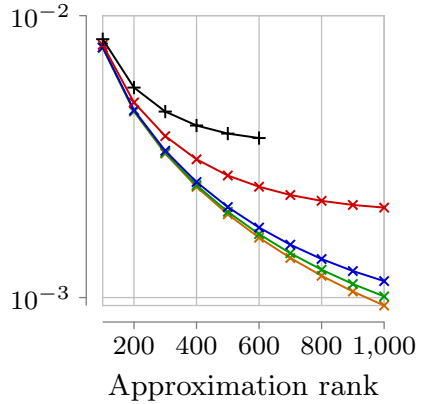
  where $\phi : x \to \phi(x)$ is the kernel-induced feature map
- We consider the radial basis function $\kappa(x_i, x_j) = e^{-\|x_i - x_j\|_2^2 / c^2}$, $c > 0$

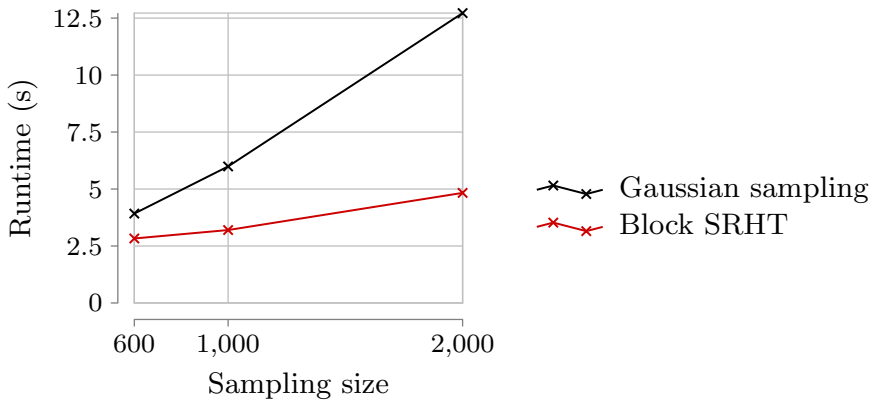
Experiments and performance of Gaussian vs SRHT

Dataset `year` with $c = 10^5$



Dataset `year` with $c = 10^4$





Radial basis function $e^{-\|x_i - x_j\|_2^2/c^2}$ on n rows of input data, $n = 65536$, 8×8 procs (for more details see [Balabanov et al., 2022]

Top graph: nuclear error $\|A - [\![A_{Nyst}]\!]_k\|_*/\|A\|_*$

Bottom graph: Runtime for computing $\Omega^T A \Omega$ (julia)

References (1)

Balabanov, O., Beaupere, M., Grigori, L., and Lederer, V. (2022).

Block subsampled randomized hadamard transform for low-rank approximation on distributed architectures.

Balabanov, O., Beaupere, M., Grigori, L., and Lederer, V. (2023).

Block subsampled randomized hadamard transform for low-rank approximation on distributed architectures.
In *ICML'23: Proceedings of the 40th International Conference on Machine Learning*, number 66, pages 1564–1576.

Demmel, J., Grigori, L., and Rusciano, A. (2023).

An improved analysis and unified perspective on deterministic and randomized low-rank matrix approximation.
SIAM Journal on Matrix Analysis and Applications, 44(2):559–591.

Eckart, C. and Young, G. (1936).

The approximation of one matrix by another of lower rank.
Psychometrika, 1:211–218.

Gittens, A. (2011).

The spectral norm error of the naive nystrom extension.
arXiv preprint arXiv:1110.5305.

Halko, N., Martinsson, P. G., and Tropp, J. A. (2011).

Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.
SIAM Rev., 53(2):217–288.

Martinsson, P.-G. and Tropp, J. A. (2020).

Randomized numerical linear algebra: Foundations and algorithms.
Acta Numerica, 29:403–572.

Tropp, J. A., Yurtsever, A., Udell, M., and Cevher, V. (2017a).

Fixed-rank approximation of a positive-semidefinite matrix from streaming data.
Advances in Neural Information Processing Systems, 30.

References (2)

Tropp, J. A., Yurtsever, A., Udell, M., and Cevher, V. (2017b).
Practical sketching algorithms for low-rank matrix approximation.
SIAM Journal on Matrix Analysis and Applications, 38(4):1454–1485.