EPFL

Volume maximization for cross approximation

Prof. Dr. D. Kressner
A. Cortinovis

▶ Setting

The goal of this project is to implement and analyze an algorithm for cross approximation based on volume maximization. We will consider the case of symmetric positive semidefinite (SPSD) matrices.

▶ Tasks

1. Implement the Adaptive Cross Approximation algorithm `ACA_SPSD`(A, k) that takes as input an SPSD matrix A and an integer k and returns an index set I of cardinality k, as seen in the lecture, with complete (diagonal) pivoting. Show that this greedy algorithm does not always output the maximum volume submatrix, by applying it to the 7×7 matrix A_1 defined as

```
theta = 0.3;
L = -tril(ones(n),-1)*cos(theta) + eye(n);
d = sin(theta).^(0:2:(2*n-2));
A = L*diag(d)*L';
```

   for rank $k = 6$.

2. Consider the following algorithm, that attempts to improve the volume of the submatrix selected by `ACA_SPSD` in the hope to obtain a better cross approximation.

Algorithm 1

```
1:  $I \leftarrow$ ACA_SPSD$(A, k)$
2:  repeat
3:      for $i \in I$ do
4:          for $j \in \{1, \dots, n\} \setminus I$ do
5:              if swapping $i$ and $j$ improves volume of selected submatrix then
6:                  $I \leftarrow (I \cup \{j\}) \setminus \{i\}$
7:                  Go to line 3
8:              end if
9:          end for
10:     end for
11: until no changes are done anymore
```

   Implement this algorithm and apply it to the matrix A_1. Does the volume improve? Does the accuracy of the cross approximation improve?

3. Consider a 200×200 matrix A_2 constructed in the following way:

```
d = (1:n).^(-2);
[Q, ~] = qr(randn(n));
A = Q*diag(d)*Q';
```

   Apply Algorithm 1 to A_2 and plot, for ranks $k = 1, \dots, 100$, the error of the cross approximation obtained by `ACA_SPSD`, the one obtained by Algorithm 1, and the singular values of A. Do the same for the Hilbert matrix $A_3(i, j) = \frac{1}{i+j-1}$.

4. Explain why the following statements hold:

   - Algorithm 1 will always stop.
   - Algorithm 1 is not guaranteed to output an index set corresponding to a maximum volume submatrix.
   - Algorithm 1 is guaranteed to output an index set I satisfying the bound $\|A - A(:, I)A(I, I)^{-1}A(I, :)\|_{\max} \leq (k+1)\sigma_{k+1}(A)$ as in the theorem by Goreinov and Tyrtyshnikov for maximum volume submatrices.

EPFL

Sketching for low-rank matrix approximation

Prof. Dr. D. Kressner
A. Cortinovis

▶ Setting

Skecthing is a popular method for low-rank approximation (that we did not cover in the lectures); see e.g. [1]. The basic idea is that instead of working with a large matrix $A \in \mathbb{R}^{n\times n}$, we use a *skecth* $A\Omega$ where Ω is a suitable random matrix with $m \ll n$ columns; an advantage of this representation is that if A is updated by a low-rank matrix B, the sketch can be updated by multiplying B with Ω without accessing A again. The goal of this project is to implement and analyze a sketch-based algorithm for low-rank approximation, taken from [2].

▶ Tasks

1. Given a matrix $A \in \mathbb{R}^{n\times m}$ and integers $0 < r \leq \ell$, the following algorithm computes a rank-r approximation $A \approx BC^T$ for $B \in \mathbb{R}^{n\times r}$ and $C \in \mathbb{R}^{m\times r}$.

Algorithm 1

```
1: procedure SKECTHING($A, r, \ell$)
2:     Draw Gaussian random matrices $X \in \mathbb{R}^{m\times r}$ and $Y \in \mathbb{R}^{n\times(r+\ell)}$
3:     Compute the products $AX$ and $Y^TA$
4:     Compute an economy QR factorization $QR = Y^TAX$
5:     Set $B \leftarrow AXR^\dagger$ and $C \leftarrow A^TYQ$
6: end procedure
```

   $R^\dagger$ denotes the Moore-Penrose pseudo-inverse of the matrix R.

   Implement this algorithm. What is the asymptotic time complexity of the algorithm in function of n, m, r, ℓ and $\mathrm{nnz}(A)$?

2. Prove that if A has rank exactly r, then `SKETCHING`(A, r, r) returns an exact low-rank factorization of A with probability 1.

   Hint: The inverse of a square Gaussian random matrix is invertible with probability 1.

3. Consider the three following 200×200 matrices:

   - The Hilbert matrix $A_1(i,j) = \frac{1}{i+j-1}$;
   - A matrix A_2 constructed in the following way:

   ```
   d = 0.8.^(1:n);
   [Q, ~] = qr(randn(n));
   [V, ~] = qr(randn(n));
   A = Q*diag(d)*V;
   ```

   - A matrix A_3 constructed similarly as before, with `d = (1:n).^(-1.5)`.

   Plot the error of the low-rank approximations that you obtain with ranks $1, \dots, 100$, for $\ell = 0, \ell = 3, \ell =$ `floor`$(r/2)$. What do you notice?

4. Explore the effect of randomness in the algorithm: For a fixed instance of matrix A_3, for $r = 1, \dots, 100$, apply the algorithm 1000 times and plot the average error and the 95% confidence interval, in the case $\ell = 3$ and then in the case $\ell =$ `floor`$(r/2)$.

▶ References

[1] Woodruff, David P. Sketching as a tool for numerical linear algebra. arXiv preprint arXiv:1411.4357 (2014). https://arxiv.org/pdf/1411.4357

[2] Nakatsukasa, Yuji. Fast and stable randomized low-rank matrix approximation. arXiv preprint arXiv:2009.11392 (2020). https://arxiv.org/pdf/2009.11392

Leverage score sampling

Prof. Dr. D. Kressner
A. Cortinovis

▶ Setting

The goal of this project is to explore the leverage score sampling for low-rank approximation. Let $A \in \mathbb{R}^{m\times n}$ and denote by $a_1, \dots, a_n$ its columns. The *leverage scores* are defined as

$$\ell_i(A) := a_i^T (AA^T)^\dagger a_i, \tag{1}$$

where $\dagger$ denotes the Moore-Penrose inverse of a matrix. The *ridge leverage scores* are defined as

$$\ell_{i,\lambda}(A) := a_i^T (AA^T + \lambda^2 I)^{-1} a_i \tag{2}$$

for a suitable regularization parameter $\lambda > 0$.

Sampling columns of A proportionally to the ridge leverage scores gives good approximation results in theory and in pratice; however, computing them is expensive, so algorithms to approximate them have been developed, see e.g. [1].

▶ Tasks

1. Prove the following properties of the leverage scores (1):

   a. $\ell_i(A) \le 1$;

   b. $\sum_{i=1}^n \ell_i(A) = \text{rank}(A)$;

   c. If a column a_i is orthogonal to all other columns then $\ell_i(A) = 1$;

   d. If A has rank k then considering a thin SVD of $A = U_k\Sigma_k V_k^T$, the leverage scores are the squared norms of the rows of V_k (recall that sampling with respect to squared norms of the rows of V_k was analyzed in Lecture 7).

2. From what you proved in point 1, it follows that if A has full column rank then all the leverage scores are equal, which corresponds to uniform column sampling; however, in many situations this is not a good idea, so in practice it is useful to consider the ridge leverage scores (2). Work out an expression of $\ell_{i,\lambda}(A)$ in function of the singular values of A and the matrix V of the right singular vectors of A.

   To get an intuition of why ridge leverage scores make sense, consider the following situation. An $m \times n$ matrix A (with $m \ge n$) has a large gap between the kth and $(k+1)$th singular values, and we choose λ such that

   $$\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_k \gg \lambda \gg \sigma_{k+1} \ge \cdots \ge \sigma_n > 0.$$

   Argue that the leverage scores are all ones, but the ridge leverage scores are *close* to the squared norms of the rows of the matrix V_k from the truncated SVD of A.

3. Implement the following sampling strategies:

   a. Ridge leverage score sampling with $\lambda = 10^{-8}$;

   b. Uniform sampling;

   c. Sampling proportionally to the squares of the norms of columns of A.

   Consider the 100×100 Hilbert matrix A. For ranks $k = 1, \dots, 50$, for each of the strategies above, sample k columns C (independently, with replacement) and plot the errors $\|A - CC^\dagger A\|$. Compare these errors with the $(k+1)$th singular value of A. As these strategies are randomized, plot the average error that you obtain running each sampling strategy for 20 times.

▶ References

[1] Cohen, M. B., Musco, C., and Musco, C. Input sparsity time low-rank approximation via ridge leverage score sampling. In Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1758–1777 (2017)

Prof. Dr. D. Kressner
A. Cortinovis

The tensor rank

▶ Setting

The goal of this project is to gain a better understanding of the tensor rank.

▶ Tasks

1. Implement the ALS method for approximating a tensor in CP decomposition.

   Apply it to the Strassen tensor $\mathcal{T}_2$ (which is associated to matrix-matrix multiplication) with tensor rank 7. Choose the initial values by taking the explicit CP decomposition from the slides (Lecture 8–9) and peturb each factor by a random matrix of norm $\varepsilon > 0$. How large can you choose ε such that the ALS method still reliably converges to the global minimum?

2. Develop a damped Gauss-Newton method for solving

   $$\min_{A,B,C} \left\{ \|\mathcal{X} - [\![A, B, C]\!]\|^2 : A \in \mathbb{R}^{n_1 \times R}, B \in \mathbb{R}^{n_2 \times R}, C \in \mathbb{R}^{n_3 \times R} \right\},$$

   with $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. Apply it to the setting from Point 1.

3. Test the algorithm from Point 2 on the $2 \times 2 \times 2$ tensor

   $$\mathcal{X} = e_1 \circ e_1 \circ e_2 + e_1 \circ e_2 \circ e_1 + e_2 \circ e_1 \circ e_1,$$

   where e_1 and e_2 are the first and second canonical basis vectors, with tensor rank 2.

ALS for CP decomposition

Prof. Dr. D. Kressner
A. Cortinovis

▶ Setting

The goal of this project is to develop a CP-based method for solving a linear system of the form

$$(I \otimes I \otimes A + I \otimes A \otimes I + A \otimes I \otimes I)x = b,$$

where A is a tridiagonal symmetric positive definite matrix of the form

$$A = (n+1)^2 \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

Letting $\mathcal{X}, \mathcal{B} \in \mathbb{R}^{n \times n \times n}$ be such that $x = \mathrm{vec}(\mathcal{X})$ and $b = \mathrm{vec}(\mathcal{B})$, this linear system is equivalent to

$$A \circ_1 \mathcal{X} + A \circ_2 \mathcal{X} + A \circ_3 \mathcal{X} = \mathcal{B}. \tag{1}$$

We will look at two different right-hand sides:

1. $\mathcal{B}_1(i_1, i_2, i_3) = \sin(\xi(i_1) + \xi(i_2) + \xi(i_3))$,
2. $\mathcal{B}_1(i_1, i_2, i_3) = \sqrt{\xi(i_1)^2 + \xi(i_2)^2 + \xi(i_3)^2}$,

with $\xi(i) = i/(n+1)$.

▶ Tasks

1. Implement the ALS method for approximating a tensor in CP decomposition. Apply it to $\mathcal{B}_1$ and $\mathcal{B}_2$ for $n = 200$ with the tensor rank chosen such that the norm of the error stays below $10^{-4}\|\mathcal{B}_i\|$.
2. Formulate a conjecture about the tensor rank of $\mathcal{B}_1$. Prove your conjecture.
3. Implement a reference method for solving (1) for small n, e.g., using `kron` and backslash. What is the largest value of n you can handle this way?
4. Develop and implement a low-rank solver for (1) by applying ALS to the optimization problem

$$\min_{\mathcal{X} \text{ in CP decomposition}} \frac{1}{2}\langle \mathcal{X}, A \circ_1 \mathcal{X} + A \circ_2 \mathcal{X} + A \circ_3 \mathcal{X} \rangle - \langle \mathcal{X}, \mathcal{B} \rangle,$$

   where $\mathcal{B}$ is in CP decomposition. Develop a heuristics for adapting the tensor rank of $\mathcal{X}$. Apply your method with $n = 200$ for $\mathcal{B}_1$ and $\mathcal{B}_2$. What is the largest value of n you can handle this way?

EPFL

HOSVD with randomized algorithm

Prof. Dr. D. Kressner
A. Cortinovis

▶ Setting

The goal of this project is to develop a modification of the HOSVD algorithm which is based on the randomized algorithm.

The modified HOSVD should be of the following form:

```
1: procedure mHOSVD($\mathcal{X}$)
2:     for $\mu = 1, 2, 3$ do
3:         $\tilde{U}_\mu \leftarrow$ orthonormal basis for approximation of range of $X^{(\mu)}$
4:     end for
5:     $\tilde{\mathcal{C}} \leftarrow \tilde{U}_1^T \circ_1 \tilde{U}_2^T \circ_2 \tilde{U}_3^T \circ_3 \mathcal{X}$
6:     $[\mathcal{C}, V_1, V_2, V_3] = \text{HOSVD}(\tilde{\mathcal{C}})$
7:     for $\mu = 1, 2, 3$ do
8:         $U_\mu \leftarrow \tilde{U}_\mu V_\mu$
9:     end for
10: end procedure
```

Lines 6-9 are only necessary if the basis matrices from Line 3 and the core tensor from Line 5 are unnecessary large, for example when there is an oversampling parameter involved in their calculation. If their size is optimal, Lines 6-9 can be skipped.

▶ Tasks

1. Implement the randomized range finder, both for given rank (use oversampling parameter) and for given precision – Algorithms 4.1 and 4.2 from [1].

2. Implement the modified HOSVD algorithm such that the Line 3 is obtained from the randomized range finder from Point 1. Adjust the algorithm to work with given rank and with given precision. Test the algorithm on a function related tensor – e.g. evaluate the function

$$f(x, y, z) = \frac{1}{\sqrt{x + y + z}},$$

on a grid $\{0.1, 0.2, \dots, n/10\}^3$ for $n = 50$.

3. Create a random tensor of size $200 \times 200 \times 200$ and approximate it by Tucker decomposition with multilinear ranks (R, R, R) for $R = 5, 10, \dots, 50$. Report the times needed for the modified HOSVD and the HOSVD to obtain such approximations and the norms of the errors of the resulting tensors.

4. Derive an error bound for the modified HOSVD algorithm – if each basis matrix is obtained such that

$$\|(I - U_\mu U_\mu^T) X^{(\mu)}\|_F \le \varepsilon,$$

find a bound for

$$\|\mathcal{X} - U_1 \circ_1 U_2 \circ_3 U_3 \circ_3 \mathcal{C}\|_F.$$

▶ References

[1] Halko, N., Martinsson, P. G., Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM review, 53(2), 217-288.