Exercise VI Numerical linear algebra

Randomized numerical linear algebra

November 16, 2023

1 Nystrom approximation

Efficiently computing low-rank approximations has been a major area of research, with applications in everything from classical problems in computational physics and signal processing to trendy topics like data science.

What is the "best" rank-k approximation to the positive semidefinite matrx A provided only with the matrix–matrix product $A\Omega$, where Ω is a known $N \times k$ matrix ($k \ll N$)?

2 Randomized algorithms for low-rank decomposition

The task of computing a low-rank approximation to a given matrix can be split naturally into two computational stages. The first is to construct a **low-dimensional subspace** that captures the action of the matrix. The second is to restrict the matrix to the subspace and then compute a standard factorization (QR, SVD, etc.) of the reduced matrix. To be slightly more formal, it can be written as algorithm 1.

Algorithm 1 Proto-Algorithm: Solving the Fixed-Rank Problem
Given an $m \times n$ matrix A, a target rank k, and an oversampling parameter p, this procedure computes an $m \times (k + p)$ matrix Q whose columns are orthonormal and whose range approximates the range of A

1.Draw a random $n \times (k + p)$ test matrix Ω.
2.Form the matrix product $Y = A\Omega$.
3.Construct a matrix Q whose columns form an orthonormal basis for the range of Y.

Therefore, the key point of low-rank approximation is to find the low-dimensional subspace.

3 Stage A::Randomized Schemes for Approximating the Range.

The most natural way to implement the proto-algorithm is to draw a random test matrix Ω from the standard Gaussian distribution. That is, each entry of Ω is an independent Gaussian random variable with mean zero and variance one. We formulate the resulting scheme as Algorithm 2.

Algorithm 2 Randomized Range Finder
Given an $m \times n$ matrix A and an integer l, this scheme computes an $m \times l$ orthonormal matrix Q whose range approximates the range of A

1.Draw a random $n \times (k+p)$ test matrix Ω.
2.Form the matrix product $Y = A\Omega$.
3.Construct a matrix Q whose columns form an orthonormal basis for the range of Y.

The goal of Algorithm 1 is to produce an orthonormal matrix Q with few columns that achieves:

$$||(I - Q * Q^*)A|| \leq \epsilon \tag{1}$$

where ϵ is a specified tolerance. The number of columns l that the algorithm needs to reach this threshold is usually slightly larger than the rank k of the smallest basis that verifies (1). We refer to this discrepancy $p = l - k$ as the **oversampling parameter**. The size of the oversampling parameter depends on several factors:

The Matrix Dimensions. Very large matrices may require more oversampling.

The Singular Spectrum. The more rapid the decay of the singular values, the less oversampling is needed. In the extreme case that the matrix has exact rank k, it is not necessary to oversample.

The Random Test Matrix. Gaussian matrices succeed with very little oversampling, but are not always the most cost-effective option.

3.1 A Posteriori Error Estimation

To handle the fixed-precision problem, where the parameter is the computational tolerance, there is a scheme for estimating how well a putative basis matrix Q captures the action of the matrix A.

Theorem 1 Let B be a real $m \times n$ matrix. Fix a positive integer r and a real number $\alpha > 1$. Draw an independent family $\{\omega(i) : i = 1, 2, \ldots, r\}$ of standard Gaussian vectors. Then

$$||B|| \leq \alpha \sqrt{\frac{2}{\pi}} \max_{i=1,\ldots,r} ||B\omega_i|| \tag{2}$$

except with probability α^{-r} .

3.2 Adaptive Randomized Range Finder

In this section, we explain how the error estimator can be incorporated into Algorithm 1 at almost no additional cost. To be precise, let us suppose that A is an $m \times n$ matrix and ϵ is a computational tolerance. We seek an integer l and an $m \times l$ orthonormal matrix $Q^{(l)}$ such that:

$$||(I - Q^{(l)}\big(Q^{(l)}\big)^*)A|| \leq \epsilon \tag{3}$$

The size l of the basis will typically be slightly larger than the size k of the smallest basis that achieves this error.

The adaptive scheme is that we can generate the basis in step 3 of Algorithm 2 incrementally.

Algorithm 3 Adaptive Randomized Range Finder
Given an $m \times n$ matrix A, a tolerance ε, and an integer r (e.g., $r = 10$), the following scheme computes an orthonormal matrix Q such that (2) holds with probability at least $1 - \min\{m, n\}10^{-r}$.

1.Draw standard Gaussian vectors $\omega^{(1)}, \ldots, \omega^{(r)}$ of length n.
2.For $i = 1, 2, \ldots, r$, compute $y^{(i)} = A\omega^{(i)}$.
3.$j = 0$.
4.$Q^{(0)} = []$, the $m \times 0$ empty matrix.
while $\max\left\{\left\|y^{(j+1)}\right\|, \left\|y^{(j+2)}\right\|, \ldots, \left\|y^{(j+r)}\right\|\right\} > \varepsilon/(10\sqrt{2/\pi})$ **do**,
  $j = j + 1$.
  Overwrite $y^{(j)}$ by $\left(\mathbf{I} - Q^{(j-1)}\left(Q^{(j-1)}\right)^*\right) y^{(j)}$.
  $q^{(j)} = y^{(j)} / \left\|y^{(j)}\right\|$.
  $Q^{(j)} = \left[Q^{(j-1)} q^{(j)}\right]$.
  Draw a standard Gaussian vector $\omega^{(j+r)}$ of length n.
  $y^{(j+r)} = \left(\mathbf{I} - Q^{(j)}\left(Q^{(j)}\right)^*\right) A\omega^{(j+r)}$.
  for $i = (j+1), (j+2), \ldots, (j+r-1)$ **do**,
    Overwrite $y^{(i)}$ by $y^{(i)} - q^{(j)}\left\langle q^{(j)}, y^{(i)}\right\rangle$.
  end for
end while

3.3 An Accelerated Technique for General Dense Matrices

This section describes a set of techniques that allow us to compute an approximate $rank - k$ factorization of a general dense m × n matrix in roughly $O(mnlog(l))$ flops, in contrast to the asymptotic cost $O(mnl)$ required by earlier methods. When the test matrix Ω is standard Gaussian, the cost of this multiplication is $O(mnl)$. The key idea of this technique is to use a structured random matrix that allows us to compute the product in $O(mnlog(l))$ flops.

The subsampled random Fourier transform, or SRFT, is perhaps the simplest example of a structured random matrix that meets our goals. An SRFT is an

$n \times l$ matrix of the form:

$$\Omega = \sqrt{\frac{n}{\ell}} DFR \tag{4}$$

where

D is an $n \times n$ diagonal matrix whose entries are independent random variables uniformly distributed on the complex unit circle,

F is the $n \times n$ unitary discrete Fourier transform (DFT), whose entries take the values $f_{pq} = n^{-1/2} \mathrm{e}^{-2\pi \mathrm{i}(p-1)(q-1)/n}$ for $p, q = 1, 2, \ldots, n$,

R is an $n \times l$ matrix that samples ℓ coordinates from n uniformly at random; i.e., its ℓ columns are drawn randomly without replacement from the columns of the $n \times n$ identity matrix. When Ω is defined by (4), we can compute the sample matrix $Y = A\Omega$ using $O(mnlog(l))$ flops via a subsampled FFT.

Algorithm 4 Fast Randomized Range Finder
Given an $m \times n$ matrix A and an integer ℓ, this scheme computes an $m \times \ell$ orthonormal matrix Q whose range approximates the range of A.

---

1. Draw an $n \times \ell$ SRFT test matrix Ω, as defined by (4.6).
2. Form the $m \times \ell$ matrix $Y = A\Omega$ using a (subsampled) FFT.
3. Construct an $m \times \ell$ matrix Q whose columns form an orthonormal basis for the range of Y, e.g., using the QR factorization $Y = QR$.

---

4 Stage B: Construction of Standard Factorizations

The algorithms for Stage A described in section 4 produce an orthonormal matrix Q whose range captures the action of an input matrix A. This section describes methods for approximating standard factorizations of A using the information in the basis Q. One way to compute the factorization is directly using SVD.

Algorithm 5 Direct SVD
Given matrices A and Q such that (5.1) holds, this procedure computes an approximate factorization $A \approx U\Sigma V^*$, where U and V are orthonormal, and Σ is a nonnegative diagonal matrix.

Form the matrix $B = Q^*A$.
Compute an SVD of the small matrix: $B = \widetilde{U}\Sigma V^*$.
Form the orthonormal matrix $U = Q\widetilde{U}$.

The cost of Algorithm 5 is generally dominated by the cost of the product QA in step 1, which takes $O(kmn)$ flops for a general dense matrix.

4.1 Postprocessing via Row Extraction.

In this section, we will introduce a method which can be faster than Algorithm 4 but less accurate.
Given a matrix Q such that $||(I-Q*Q^*)A|| \leq \epsilon$ holds, we can obtain a $rank-k$ factorization.

$$A \approx XB \tag{5}$$

where B is a $k \times n$ matrix consisting of k rows extracted from A. The approximation (4) can be produced without computing any matrix-matrix products, which makes this approach to post-processing very fast. The drawback comes because the error $\|A-XB\|$ is usually larger than the initial error $\|A - QQ^*A\|$, especially when the dimensions of A are large. **why?**

We simply construct the interpolative decomposition of the matrix Q:

$$Q = XQ_{(J,:)}$$

The index set J marks k rows of Q that span the row space of Q, and X is an $m \times k$ matrix whose entries are bounded in magnitude by two and contains the $k \times k$ identity as a submatrix: $X_{(J,:)} = \mathbf{I}_k$. Then, we reach

$$A \approx QQ^*A = XQ_{(J,:)}Q^*A. \tag{6}$$

Since $X_{(J,:)} = \mathbf{I}_k$, equation (6) implies that $A_{(J,:)} \approx Q_{(J,:)}Q^*A$. Therefore, (5) follows when we put $B = A_{(J,:)}$.

Algorithm 6 illustrates an SVD calculation.This procedure requires $O(k^2(m+n))$ flops. The following lemma guarantees the accuracy of the computed factors.

Lemma 1. Let A be an $m \times n$ matrix and let Q be an $m \times k$ matrix that satisfy $||A - QQ^*A|| \leq \epsilon$. Suppose that U, Σ, and V are the matrices constructed by Algorithm 6. Then:

$$\|A - U\Sigma V^*\| \leq [1 + \sqrt{1 + 4k(n-k)}]\varepsilon.$$

Algorithm 6 SVD VIA Row Extraction
Given matrices A and Q such that (5.1) holds, this procedure computes an approximate factorization $A \approx U\Sigma V^*$, where U and V are orthonormal, and Σ is a nonnegative diagonal matrix.

Compute an **ID** $Q = XQ_{(J,:)}$.
Extract $A_{(J,:)}$, and compute a QR factorization $A_{(J,:)} = R^*W^*$..
Form the product $Z = XR^*$.
Compute an SVD $Z = U\Sigma\widetilde{V}^*$.
Form the orthonormal matrix $V = W\widetilde{V}$.

The final factorization identifies a collection of k columns from a rank- k matrix A that span the range of A. **The Interpolative Decomposition**

(ID). To be precise is to compute an index set $J = [j_1, \ldots, j_k]$ such that

$$A = A_{(:,J)}X$$

where X is a $k \times n$ matrix that satisfies $X_{(:,J)} = \mathbf{I}_k$. Furthermore, no entry of X has magnitude larger than two. In other words, this decomposition expresses each column of A using a linear combination of k fixed columns with bounded coefficients. Stable and efficient algorithms for computing the ID appear in the papers [26, 68]. It is also possible to compute a two-sided ID,

$$A = WA_{(J',J)}X,$$

where J' is an index set identifying k of the rows of A, and W is an $m \times k$ matrix that satisfies $W_{(J',:)} = \mathbf{I}_k$ and whose entries are all bounded by two.

5 Exercise

1. randomly generate matrix $A = rand(400, 10) * rand(10) * rand(10, 500)$, computing its rank-10 approximation with the methods as follows:

Methods	Stage A	Stage B
1	-	TSVD(matlab function)
2	Algorithm 2	Algorithm 5
3	Algorithm 3	Algorithm 5
4	Algorithm 4	Algorithm 5
5	Algorithm 2	Algorithm 6
6	Algorithm 3	Algorithm 6
4	Algorithm 4	Algorithm 7

comparing the speed and the accuracy of these methods.

6 Reference

References

[1] Halko, N. and Martinsson, P. G. and Tropp, J. A., Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions,.