Scientific computing project

Arthur Bongiovanni
Léa Houot

May 3, 2024

Contents

1 Introduction

Matrix reduction is a technique that allows one to reduce the data in matrix in order to simplify it. One of the ways to do a matrix reduction is a Principal Component Analysis (PCA) that is a dimension reduction. This technique uses the spectral decomposition of the symmetric variance/covariance matrix. However, you don't need all of it, you only need the leading eigenpairs.

We saw in CTD the power method to compute the leading eigenpairs but it has some limits that we will see. We will also see an other algorithm that is more effective, the subspace iteration method.

2 Limitations of the power method

Question 1 : We started by comparing the calculation time of the power method and the `eig` function of MATLAB.

Type	size	eig function	Power method v11
Type 1	100×100	$0ms$	$260ms$
	200×200	$20ms$	$1640ms$
	400×400	$60ms$	$8340ms$
Type 2	100×100	$0ms$	$20ms$
	200×200	$10ms$	$20ms$
	400×400	$40ms$	$60ms$
Type 3	100×100	$0ms$	$20ms$
	200×200	$20ms$	$40ms$
	400×400	$30ms$	$330ms$
Type 4	100×100	$0ms$	$180ms$
	200×200	$10ms$	$1320ms$
	400×400	$60ms$	$8420ms$

Table 1: Computing time of the eig function and power method v11

Question 2 : We changed the method by doing only one matrix $\times$ vector product in the loop, we obtained the algorithm 1.

Algorithm 1 Vector power method v2

Input : Matrix ($A \in \mathbb{R}^{n\times n}$)
Output :(λ_1, v_1) eigenpair associated to the largest (in module) eigenvalue.
$v \in \mathbb{R}^n$ given
$z = A \cdot v$
$\beta = v^T \cdot z$
repeat
    $y = A \cdot v$
    $v = y / \|y\|$
    $\beta_{old} = \beta$
    $\beta = v^T \cdot z$
until $|\beta - \beta_{old}| / |\beta_{old}| < \varepsilon$
$\lambda_1 = \beta$ and $v_1 = v$

Type	size	Power method v11	Power method v12
Type 1	100×100	$260ms$	$200ms$
	200×200	$1640ms$	$680ms$
	400×400	$8340ms$	$4340ms$
Type 2	100×100	$20ms$	$20ms$
	200×200	$20ms$	$10ms$
	400×400	$60ms$	$40ms$
Type 3	100×100	$20ms$	$20ms$
	200×200	$40ms$	$30ms$
	400×400	$330ms$	$300ms$
Type 4	100×100	$180ms$	$160ms$
	200×200	$1320ms$	$740ms$
	400×400	$8420ms$	$4370ms$

Table 2: Computing time of the power method v11 and power method v12

We can see that the computing time of the power method v12 is two times faster. And we did not put it in the table but the quality of the eigenpairs and of the eigenvalues do not change.

Question 3 : The main drawback of the deflated power method in terms of computing time is that it varies greatly depending of the type of matrix. Even when we use a type 1 matrix it fails to converge if the matrix is too big.

3 Extending the power method to compute dominant eigenspace vectors

Question 4 : If we apply algorithm 1 to a set of m vectors instead of applying on one vector it converges towards a matrix which each column is an eigenvector associated with the same eigenvalue and not m vectors associated to different eigenvalues.

3.1 Rayleigh quotient

Question 5 : Computing the spectral decomposition of H does not cause a problem because $H \in \mathbb{R}^{m \times m}$. Indeed $A \in \mathbb{R}^{n \times n}$ and $m \leqslant n$ because m is the number of required eigenpairs and there is at most n.

3.2 subspace iter_v1: improved version making use of Raleigh-Ritz projection

Question 7 : This algorithm is used in the `subspace_iter_v1` file. On the **Algorithm 2**, the line refers to the localization of the different steps and operations in the file.

4 subspace_iter_v2 and subspace_iter_v3: toward an efficient solver

4.1 Block approach

Question 8 : In order to compute A^2 for each element we make the dot product of a row of A with a column of A, it means n multiplication and n-1 additions, so for an element of A^2 we have (2n-1) flops

So for A^p we have a computation of $(p-1) \times n^2 \times (2n-1) = 2(p-1)n^3$ flops For $A^p \cdot V$ this is $(p-1) \times n^2 \times (2n-1) + n \times m \times (2n-1) = 2(p-1)n^3$ flops

If we begin by doing $V = A \cdot V$ and after we compute $A^p \cdot V$ we will have $(p-1) \times n \times m \times (2n-1) + n \times m \times (2n-1) = 2(p-1)mn^2 << 2(p-1)n^3$ because $m < n$

Algorithm 2 Subspace iteration method v1 with Rayleigh-Ritz projection

Input : Symmetric matrix $A \in \mathbb{R}^{n\times n}$, tolerance ε, $MaxIter$ (max nb of iterations) and $PercentTrace$ the target percentage of the trace of A
Output : n_{ev} dominant eigenvectors V_{out} and the corresponding eigenvalues Λ_{out}.
Generate a set of m orthonormal vectors $V \in \mathbb{R}^{n\times m}$ ▷ Lines 48-49
$k = 0$ ▷ Line 38
$PercentReached = 0$ ▷ Line 45
repeat
    $k = k + 1$ ▷ Line 54
    Compute Y such that $Y = A \cdot V$ ▷ Line 56
    $V \leftarrow$ orthonormalisation of the columns of Y ▷ Line 58
    $Rayleigh - Ritz\ projection$ applied on matrix A and orthonormal vectors V ▷ Line 61
    Convergence analysis step: save eigenpairs that have converged ▷ Lines 70-112
    and update $PercentReached$ ▷ Line 115
until ($PrecentReached > PercentTrace$ or $n_{ev} = m$ or $k > MaxIter$)

Question 10 : The computing time is reduced as p is greater, as the algorithm converges faster (the number of iterations falls). This is because the eigenvalues of A^P are raised to to the power of p as well, heightening the differences in magnitude of the eigenvalues. This in turn makes the convergence much faster than before, as the conditionment of the matrix is much greater. There is little to no loss on precision, unless p is unreasonably high.

4.2 Deflation method (subspace_iter_v3)

Question 11 : In `subspace_iter_v1` like in `subspace_iter_v2` the vectors with the largest eigenvalue converge faster and so the vectors considered as having converged continue to be updated. Like that we finish with some vectors refined with a high accuracy and some vectors with less accuracy.

Question 12 : In subspace_iter_v3 once a vector of V converged we isolate him and we stop doing operation on him. Like that we reduce the number of operation and we have less chances to make him more precise because it doesn't update again in the loop.

5 Numerical experiments

Question 14 : For the difference between the types of matrix we have :

- Type 1 : the eigenvalues range form 1 to n with a step of 1;
- Type 2 : the eigenvalues are random and between 0 and 1 not included ;
- Type 3 : the eigenvalues are random and between 0 and 1, and one of the eigenvalues is 1 ;
- Type 4 : the eigenvalues are the eigenvalues of the type 1 but divided by n.

Question 15 : We have compared the performances of the algorithms implemented as well as those provided (eig) for different types and sizes of matrix. The results, presented on Table 3, show that the versions v1 and v2 increases the computing's speed (the version v3's results aren't showed because there seems to be a mistake in our matlab program causing the algorithm to almost never achieve convergence)

Figure 1: Distribution of eigenvalues for a type 1 matrix of size 200 $\times$ 200

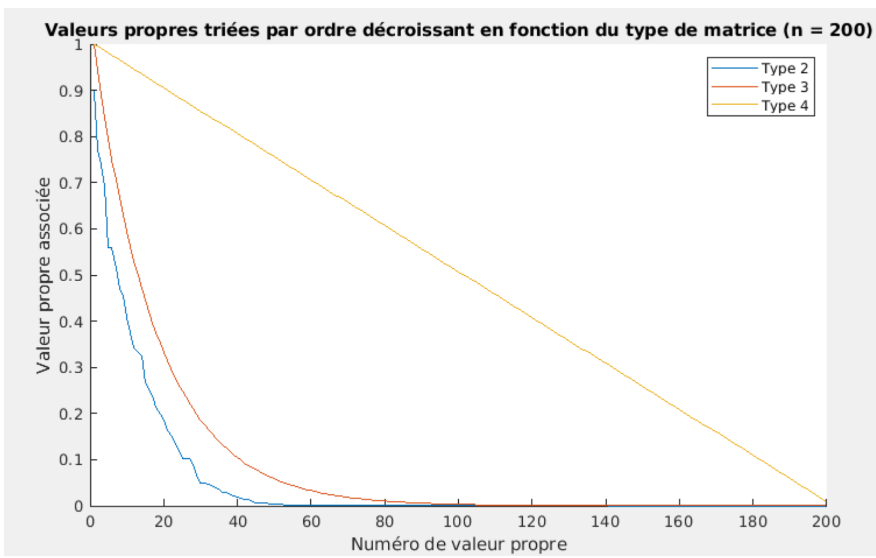

Figure 2: Distribution of eigenvalues for a type 2, 3 and 4 matrix of size 200 $\times$ 200

Type	size	eig	Subspace iteration V0	Subspace Iteration V1	Subspace Iteration V2
Type 1	100×100	$10ms$	$590ms$	$100ms$	$10ms$
	200×200	$10ms$	$1420ms$	$200ms$	$20ms$
	1000×1000	$540ms$	$24550ms$	no convergence	no convergence
Type 2	100×100	$30ms$	$20ms$	$10ms$	$10ms$
	200×200	$0ms$	$70ms$	$60ms$	$50ms$
	1000×1000	$340ms$	$3940ms$	$180ms$	$230ms$
Type 3	100×100	$0ms$	$80ms$	$20ms$	$10ms$
	200×200	$0ms$	$160ms$	$20ms$	$10ms$
	1000×1000	$480ms$	$4030ms$	$780ms$	$370ms$
Type 4	100×100	$0ms$	$620ms$	$70ms$	$10ms$
	200×200	$10ms$	$1070ms$	$210ms$	$40ms$
	1000×1000	$430ms$	$25590ms$	no convergence	no convergence

Table 3: Computing time of the eig function and the subspace iteration v0, v1, v2 and v3

6 Application to Image Compression

An image can be describe by a matrix I of size $q \times p$. To perform an image compression we can use the k-low-rank approximation of I, I_k. In order to generate the matrix I_k :

- We create a matrix $M = I \times I'$ (or $M = I' \times I$ if $p < q$)
- We find the k eigenpairs of M
- We create the Σ_k with the k eigenvalues
- We create the U_k with the k eigenvectors (or V_k)
- We create the matrix V_k by using the relation between the vectors of U_k and those of V_k (or the matrix U_k with V_k
- $I_k = U_k$ x Σ_k x V_k'

Question 1 : The size of the elements the triplet (Σ_k, U_k, V_k) are :

- Σ_k : $\mathbb{R}^{k \times k}$;
- U_k : $\mathbb{R}^{q \times k}$;
- V_k : $\mathbb{R}^{p \times k}$.