Subspace Iteration Methods

Ayoub Bouchama
Oussama ElGuerraoui

Le 19 avril 2023

Contents

List of Figures

1 Introduction

Dans le cadre du cours d'Analyse de Données, nous avons vu que pour réduire la dimension d'un ensemble de données en utilisant l'Analyse en Composantes Principales (ACP), il n'est pas nécessaire d'effectuer la décomposition spectrale complète de la matrice de variance/covariance symétrique. En effet, seuls les premiers paires propres fournissent suffisamment d'informations sur les données.

Lors de la première session du cours de Calcul Scientifique CTD, nous avons introduit la méthode de la puissance pour calculer les premiers paires propres d'une matrice. Couplée à un processus de déflation, les paires propres suivantes peuvent être calculées.

Dans ce projet, nous verrons que cet algorithme spécifique n'est pas efficace en termes de performances. Nous présenterons ensuite une méthode plus efficace appelée méthode d'itération de sous-espaces, basée sur un objet appelé quotient de Rayleigh.

Nous étudierons quatre variantes de cette méthode.

2 Limites de la méthode de puissance

2.1 Question-1

	Temps d'exécution avec eig	Temps d'exécution avec la méthode de la puissance itérée
Matrice 20 x 20 - type 1	0.000e+00	0.000e+00
Matrice 20 x 20 - type 2	0.000e+00	0.000e+00
Matrice 20 x 20 - type 3	0.000e+00	0.000e+00
Matrice 20 x 20 - type 4	0.000e+00	0.000e+00
Matrice 200 x 200 - type 1	0.000e+00	1.922e+00
Matrice 200 x 200 - type 2	1.562e-02	1.562e-02
Matrice 200 x 200 - type 3	0.000e+00	3.125e-02
Matrice 200 x 200 - type 4	0.000e+00	1.875e+00
Matrice 100 x 100 - type 1	0.000e+00	7.812e-02
Matrice 100 x 100 - type 2	0.000e+00	0.000e+00
Matrice 100 x 100 - type 3	0.000e+00	0.000e+00
Matrice 100 x 100 - type 4	1.562e-02	3.125e-02

Figure 1: Tableau des temps d'exécution des méthodes 'eig' et 'puissance itérée'

Nous avons observé que la méthode "eig" est plus performante et plus rapide que la méthode "power iteration" pour calculer les valeurs propres de matrices de différentes tailles et de différents types. Cependant, la qualité des vecteurs propres obtenus avec la méthode "eig" varie davantage

que celle des vecteurs propres obtenus avec la méthode "power iteration", qui reste constante. Cette différence de qualité est particulièrement notable pour les matrices de type 1. Néanmoins, la méthode "power iteration" ne permet pas de calculer les vecteurs propres pour des matrices de grande taille.

2.2 Question-2

Algorithme 1 (version-2)
Input : Matrix $A \in R^{n \times n}$
Output : (λ_1, v_1) eigenpair associated to the largest (in module) eigenvalue.
$v \in R^n$ given
$z = A.v$
$\beta = v^T.z$
repeat

$y = z$
$v = y/||y||$
$z = A.v$
$\beta_{old} = \beta$
$\beta = v^T.z$

until $|\beta - \beta_{old}|/|\beta_{old}| < \epsilon$
$\lambda_1 = \beta$ and $v_1 = v$

Résultats Obtenus Pour Matrice 200 x 200 - type 1

```
Matrice 200 x 200 - type 1

******* calcul avec la méthode de la puissance itérée ******

Temps puissance itérée = 2.516e+00
Nombre de valeurs propres pour attendre le pourcentage = 46
Qualité des couples propres (par rapport au critère d'arrêt) = [9.977e-09 , 1.422e-08]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.990e-14]
```

Figure 2: Tableau des temps d'exécution dee la méthode de puissance itérée type 1

```
Matrice 200 x 200 - type 1

******* calcul avec la méthode de la puissance itérée améliorée ******

Temps puissance itérée = 1.156e+00
Nombre de valeurs propres pour attendre le pourcentage = 46
Qualité des couples propres (par rapport au critère d'arrêt) = [9.977e-09 , 1.422e-08]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.990e-14]
```

Figure 3: Tableau des temps d'exécution dee la méthode de puissance itérée améliorée type 1

2.3 Question-3

Il est constaté que le temps d'exécution a été réduit de moitié. Cette amélioration est due au fait que le calcul du produit matriciel était effectué à chaque itération de la boucle dans la méthode simple, contrairement à l'approche améliorée que nous avons développée.

3 Extension de la méthode de puissance pour calculer les vecteurs d'espace propre dominants

3.1 Question-4

En utilisant l'algorithme subspace iter v0, la matrice V converge vers la matrice de changement de base X qui relie H aux vecteurs propres de A. En d'autres termes, les colonnes de la matrice V convergent vers les vecteurs propres de A, qui sont représentés par les colonnes de la matrice X. Ainsi, la matrice V converge effectivement vers les vecteurs de changement de base qui relient H aux vecteurs propres de A.

3.2 Question-5

Lorsque nous utilisons l'Algorithme 2 pour calculer le sous-espace propre dominant de la matrice A, nous avons besoin de décomposer spectrale de la matrice H. Bien que cela puisse sembler coûteux en termes de calculs, il est en fait plus simple et moins coûteux de le faire plutôt que de décomposer la matrice A. En effet, la matrice H est généralement beaucoup plus petite que la matrice A :

$$m \ll n$$

donc nous pouvons économiser beaucoup de temps et de ressources en calculant la décomposition spectrale de H plutôt que celle de A.
De plus, si nous trouvons les vecteurs propres de H, nous pouvons utiliser une simple transformation linéaire pour retrouver les vecteurs propres de A sans avoir à décomposer spectrale de A.

3.3 Question-6

Voir **subspace_iter_v0** et **test_v0** dans les fichiers matlab rendus !

3.4 Question-7

Input: Symmetric matrix A $\in R^{n\times n}$, 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 an initial set of m orthonormal vectors V $\in$ R n×m; k = 0; $PercentReached = 0$
repeat

l_54 : k = k + 1
l_56 : Compute Y such that Y = A · V
l_58 : V ← orthonormalisation of the columns of Y
l_61 : Rayleigh-Ritz projection applied on matrix A and orthonormal vectors V
l_64 - l_112 : Convergence analysis step: save eigenpairs that have converged and update PercentReached

until (PercentReached > PercentTrace or n_{ev} = m or k > MaxIter)

3.5 Question-8

En general, la complexité associé au produit de deux matrices A $\in R^{m\times n}$ et B $\in R^{n\times p}$ est $\mathcal{O}(2mnp)$. Car ce processus est de l'ordre de mn opérations pour chaque élément de la matrice résultante, qui est de taille $m \times p$. Donc, le nombre total d'opérations nécessaires est de l'ordre de $mn \times p$, ce qui donne une complexité de $\mathcal{O}(mnp)$. Cependant, lors de l'exécution de l'algorithme de multiplication

de matrices, chaque élément de la matrice résultante est le résultat de deux produits scalaires et une addition, soit un total de 2 opérations. Ainsi, la complexité totale de l'algorithme est doublée, d'où le résultat précédent.
Donc la complexité du produit d'un matrice $A \in R^{n \times n}$ par elle meme est :

$$\mathcal{O}(2n^3)$$

D'où la complexité du produit A^p est :

$$\mathcal{O}(2pn^3)$$

La complexité du produit $A^p \in R^{n \times n}$ avec $V \in R^{n \times m}$ est d'après ce qui précède :

$$\mathcal{O}(2mn^2)$$

Donc la complexité total du produit $A^p.V$ est :

$$\mathcal{O}(2mn^2 + 2n^3) = \mathcal{O}(2n^2(m+n)) = \mathcal{O}(2n^3)$$

car :

$$m \ll n$$

Pour minimiser le cout du calcul de $A^p \cdot V$, on doit calculer A^p avant la boucle. Cela permet de ne pas recalculer A^p à chaque itération, ce qui peut être coûteux si p est grand. On peut donc calculer A^p avant d'entrer dans la boucle et ensuite utiliser $V = A^p.V$ à chaque itération, au lieu de $V = A.V$. Cette méthode permet de réduire le nombre de produits matrice-vecteur nécessaires et donc de diminuer le coût de calcul.

3.6 Question-9

Voir le fichier '**subspace_iter_v2.m**' pour identifier le changement effectué !

3.7 Question-10

Nous observons une diminution du nombre d'itérations à mesure que nous augmentons la puissance p de la matrice A. Cependant, nous constatons que la convergence n'est pas atteinte à partir de p = 9.

3.8 Question-11

Lorsque la valeur propre de A est grande, il peut être plus facile d'obtenir une précision suffisante sur les vecteurs propres associés à cette valeur propre en utilisant le critère de convergence. Cela s'explique par le fait que la contribution de cette valeur propre est plus importante que les autres valeurs propres de la matrice A. En s'appuyant sur le critère de convergence $||A.v - \beta.v|| < |\beta|$, si la valeur propre est grande, cela signifie que la matrice A est fortement influencée par cette valeur propre, et donc que la différence entre A · v et · v sera plus importante si le vecteur v est mal approximé. Donc, le critère de convergence sera plus facilement satisfait pour des valeurs propres plus grandes.

3.9 Question-12

La méthode subspace_iter_v3 pour les eigenfaces est probablement plus rapide mais moins précise que les autres méthodes car elle vise uniquement à déterminer les couples propres non encore identifiés, sans chercher à améliorer la précision des couples propres déjà connus.

3.10 Question-13

Voir le fichier '**subspace_iter_v3.m**' pour identifier le changement effectué !

4 Expériences Numériques

4.1 Question-14

Dans cette section, on a tracé les valeurs propres de chaques types de matrice.
- Type-1: valeurs propres varient de 1 à n ($\lambda \in \{1..n\}$)
- Type-2: valeurs propres générées aléatoirement de manière uniforme sur une échelle logarithmique, avec des valeurs comprises entre 1/cond et 1. La valeur de cond est de 1e10.
- Type-3: valeurs propres suivant une décroissance exponentielle déterminée par la formule $D(i) = cond^{(-(i-1)/(n-1))}$, où i est l'indice de la valeur propre et n est le nombre total de valeurs propres. La valeur de cond est de 1e5.
- Type-4: valeurs propres calculées en utilisant la formule $D(i) = 1 - ((i-1)/(n-1)) \times (1-1/cond)$, où i est l'indice de la valeur propre et n est le nombre total de valeurs propres. La valeur de cond est de 1e2.

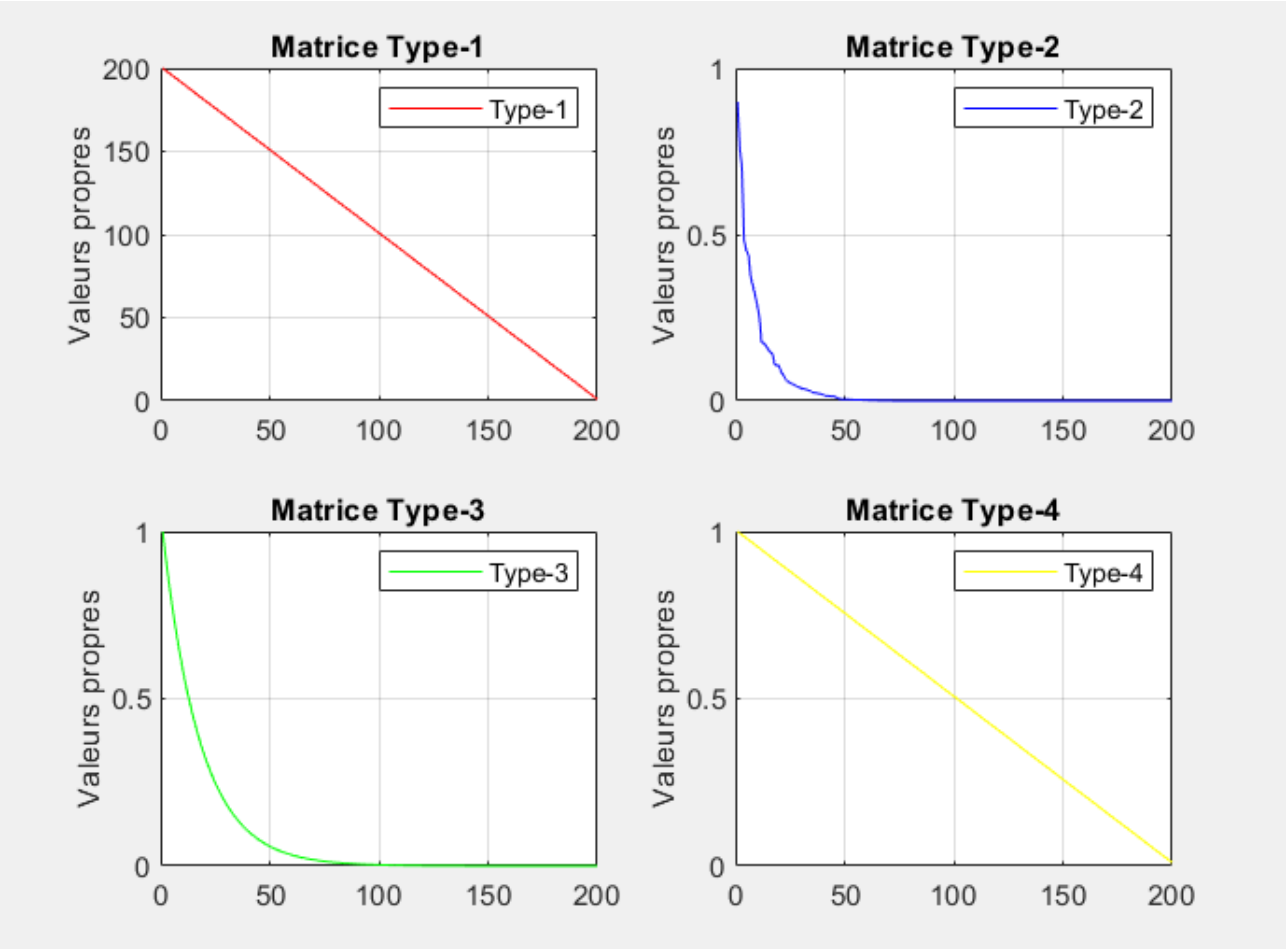

Figure 4: Valeurs propres de chaque type de matrice

4.2 Question-15

Pour des matrices de petite taille, toutes les méthodes ont les memes performances, pourtant il vaut mieux priviligier la méthode subspace_iter_v1 qui est légèrement rapide et moins couteuse.

- Pour le **Type-1** de matrice, la méthode subspace_iter_v3 semble etre la plus rapide mais la méthode subspace_iter_v2 reste la plus précise.

- Pour le **Type-2** de matrice, la méthode subspace_iter_v2 semble etre la plus rapide pourtant la méthode subspace_iter_v3 reste la meilleur quant à une combinaison temps/précision.

- Pour le **Type-3** de matrice, la méthode subspace_iter_v1 présente une bonne combinaison temps/précision mais la méthode subspace_iter_v2 reste aussi un choix à prendre en considération.

- Pour le **Type-4** de matrice, la méthode subspace_iter_v2 est le meilleur choix à faire quand traitant des matrices de grandes tailles, neanmoins, l'utilisation de la méthode subspace_iter_v1 pour celles de petites tailles reste le meilleur choix à considérer.

Finalement, nous avons pu donner une bonne amélioration aux méthodes subspace_iter pour arriver à une version V3 capable de compéter avec la méthode 'eig'.

5 Conclusion

En conclusion, dans ce projet, nous avons étudié plusieurs méthodes de calcul des valeurs propres et vecteurs propres, en particulier la méthode de puissance itérée et ses variantes, ainsi que la méthode de subspace itérative. Nous avons pu constater les avantages et les limites de chaque méthode en fonction de la taille de la matrice et de la structure des valeurs propres. Nous avons également comparé différentes méthodes de calcul des valeurs propres et vecteurs propres, en particulier la méthode de puissance itérée et ses variantes, ainsi que la méthode de subspace itérative. En comparant leurs performances, nous avons pu choisir la méthode la plus performantes capable de compéter avec la fonction 'eig'.