TD4 - Valeurs singulières et analyse en composantes principales

Alexandra Bac

Méthodes numériques
Polytech Marseille - IRM 3A

1 TD

Algorithme 1 Algorithme des puissances itérées (sans normalisation).

Entrées: Matrice $\boldsymbol{A} \in \mathbb{R}^{d \times d}$, nombre d'itérations N
Sorties: Plus grande valeur propre λ de $\boldsymbol{A}$ et vecteur propre $\boldsymbol{u}$ associé.

Initialiser $\boldsymbol{x}_0 \in \mathbb{R}^d$ aléatoirement
pour n de 1 à N **faire**
  $\boldsymbol{x}_n \leftarrow \boldsymbol{A}\boldsymbol{x}_{n-1}$
fin pour
$\boldsymbol{u} \leftarrow \frac{\boldsymbol{x}_N}{\|\boldsymbol{x}_N\|_2}$
$\lambda \leftarrow \frac{\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{u} \rangle}{\|\boldsymbol{u}\|_2^2}$
renvoyer $\lambda, \boldsymbol{u}$

Algorithme 2 Algorithme des puissances itérées (avec normalisation).

Entrées: Matrice $\boldsymbol{A} \in \mathbb{R}^{d \times d}$, nombre d'itérations N
Sorties: Plus grande valeur propre λ de $\boldsymbol{A}$ et vecteur propre $\boldsymbol{u}$ associé.

Initialiser $\boldsymbol{y}_0 \in \mathbb{R}^d$ aléatoirement
$\boldsymbol{x}_0 \leftarrow \frac{\boldsymbol{y}_0}{\|\boldsymbol{y}_0\|_2}$
pour n de 1 à N **faire**
  $\boldsymbol{y}_n \leftarrow \boldsymbol{A}\boldsymbol{x}_{n-1}$
  $\boldsymbol{x}_n \leftarrow \frac{\boldsymbol{y}_n}{\|\boldsymbol{y}_n\|_2}$
fin pour
$\boldsymbol{u} \leftarrow \boldsymbol{x}_N$
$\lambda \leftarrow \langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{u} \rangle$
renvoyer $\lambda, \boldsymbol{u}$

Exercice 1 (Méthode des puissances itérées)**.** On considère la méthode des puissances itérées, qui permet de calculer rapidement la plus grande valeur propre d'une matrice $\boldsymbol{A}$ de dimensions $d \times d$ et un vecteur propre associé. Une version simplifiée est donnée par l'Algorithme 1. Elle permet de comprendre mathématiquement le fonctionnement de cette méthode. La version utilisée en pratique est donnée par l'Algorithme 2. On s'intéresse ici à la complexité et à la preuve de ces algorithmes. On supposera que $\boldsymbol{A}$ admet d valeurs propres $|\lambda_1| > |\lambda_2| \geq \ldots > |\lambda_d|$ rangées dans l'ordre décroissant en valeur absolue et on note $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d$ les vecteurs propres associés. On suppose que $|\lambda_1|$ est strictement plus grande que les autres $|\lambda_i|$, ce qui n'est pas toujours le cas.

(i) Quelle est la complexité de l'Algorithme 1 en fonction du nombre d'itération N et de la dimension d?

(ii) En écrivant $\boldsymbol{x}_0$ sous la forme $\boldsymbol{x}_0 = \sum_{i=1}^{n} \alpha_i \boldsymbol{u}_i$, montrez par récurrence que pour tout $n \geq 0$, on a

$$\boldsymbol{x}_n = \sum_{i=1}^{d} \alpha_i \lambda_i^n \boldsymbol{u}_i$$

(iii) Montrez que $\frac{\boldsymbol{x}_n}{\lambda_1^n}$ tend vers $\alpha_1 \boldsymbol{u}_1$ quand n tend vers $+\infty$.

(iv) Montrez qu'à convergence, on a $\frac{\boldsymbol{x}_N}{\|\boldsymbol{x}_N\|_2} = \boldsymbol{u}_1$ ou $\frac{\boldsymbol{x}_N}{\|\boldsymbol{x}_N\|_2} = -\boldsymbol{u}_1$ et $\frac{\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{u} \rangle}{\|\boldsymbol{u}\|_2^2} = \lambda_1$.

(v) Pourquoi le nombre d'itérations N nécessaire pour atteindre la convergence est-il faible? Vous pouvez prendre par exemple $\lambda_2 = 0.5\lambda_1$ pour vous en persuader.

(vi) En considérant la limite de λ_1^n quand n tend vers $+\infty$, expliquez pourquoi en pratique, l'Algorithme 1 ne fonctionne pas bien (séparez les cas $|\lambda_1| < 1$ et $|\lambda_1| > 1$).

(vii) (optionnel) Pour prouver l'Algorithme 2, montrez par récurrence que pour tout $n \geq 0$, on a

$$\boldsymbol{x}_n = \frac{1}{\sqrt{\sum_{i=1}^{d} \alpha_i^2 \lambda_i^{2n}}} \sum_{i=1}^{d} \alpha_i \lambda_i^n \boldsymbol{u}_i$$

En factorisant λ_1^n au numérateur et au dénominateur, montrez ensuite que la limite de $\boldsymbol{x}_n$ quand n tend vers $+\infty$ est $\pm \boldsymbol{u}_1$.

Exercice 2 (Algorithme des puissances itérées pour la SVD et l'ACP)**.** On considère une matrice rectangulaire quelconque $\boldsymbol{B} \in \mathbb{R}^{n \times m}$ et on note $p = \min(m, n)$. Vous allez ici réutiliser l'algorithme des puissances itérées pour le calcul de la SVD $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. On rappelle que l'algorithme des puissances itérée vu au TD/TP 1 prend en argument une matrice carrée $\boldsymbol{A}$ et renvoie sa valeur propre dominante λ_1 ($\forall i \geq 2, |\lambda_1| > |\lambda_i|$) et un vecteur propre associé.

(i) Rappelez la relation entre la SVD de $\boldsymbol{B}$ et la décomposition en valeur propre.

(ii) En utilisant l'algorithme des puissances itérées, donnez le pseudo-code d'un algorithme qui calcule la valeur singulière dominante σ_1 de $\boldsymbol{B}$ et un vecteur singulier à droite associé.

(iii) Comment compléter l'algorithme pour calculer ensuite un vecteur singulier à gauche associé à σ_1?

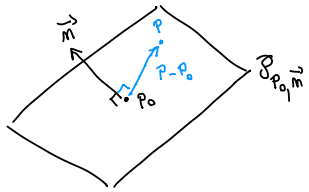
(iv) En écrivant $\boldsymbol{B} = \sum_{i=1}^{p} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ et en isolant le terme $i = 1$, trouvez un algorithme permettant de calculer la 2e valeur singulière dominante σ_2 et des vecteurs singuliers à droite et à gauche associés?

(v) En déduire un algorithme permettant de calculer la SVD de $\boldsymbol{B}$ tronquée aux s valeurs singulières dominantes, où s est un entier donné tel que $1 \leq s \leq p$.

(vi) En déduire un algorithme permettant de calculer l'ACP d'un ensemble de points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$, en notant k le nombre de composantes principales à extraire; les sorties de l'algorithme seront les vecteurs principaux $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ ainsi que le vecteur moyen $\overline{\boldsymbol{x}}$ des points.

Exercice 3 (ACP et calcul d'un plan approximant aux moindres carrés (3D))**.** Dans cet exercice, on s'intéresse au plan approximant aux moindres carrés un nuage de points 3D $\{P_i : i = 1 \ldots N\}$. On recherche donc le plan $\mathcal{P}$ minimisant $\sum_i \mathrm{d}(P_i, \mathcal{P})^2$. Tout plan (affine) est déterminé par un point P_0 lui appartenant et sa normale ($\vec{n}$ dont on peut supposer qu'elle est de norme 1). Etant donné un point P_0 et un vecteur $\vec{n}$,

on notera donc $\mathcal{P}_{P_0,\vec{n}}$ le plan passant par P_0 et de normale $\vec{n}$.



$P \in \mathcal{P}_{P_0,\vec{n}} \iff P - P_0 \perp \vec{n}$
$\iff \langle (P - P_0), \vec{n} \rangle = 0$

On rappelle que la distance (signée) d'un point P à un plan $\mathcal{P}_{P_0,\vec{n}}$ est donnée par

$$\mathrm{d}(P, \mathcal{P}_{P_0,\vec{n}}) = \frac{\langle (P - P_0), \vec{n} \rangle}{\|\vec{n}\|}$$

et on rappelle, pour deux vecteurs colonnes U, V, les identités :

$$\langle U, V \rangle = U^t \times V = V^t \times U$$

avec lesquelles nous jouerons dans la suite.

Le plan approximant aux moindres carrés est donné par le point P_0 et le vecteur $\vec{n}$ minimisant :

$$f(P_0, \vec{n}) = \sum_{i=1}^{N} \frac{(\langle (P_i - P_0), \vec{n} \rangle)^2}{\|\vec{n}\|^2} \tag{1}$$

(i) On note $\bar{P}$ le barycentre des points, ie : $\bar{P} = \frac{1}{N} \sum_i P_i$. Pour l'instant, on supposera que $\|\vec{n}\| = 1$. En utilisant $P_i - P_0 = (P_i - \bar{P}) + (\bar{P} - P_0)$, montrer que

$$f(P_0, \vec{n}) = f(\bar{P}, \vec{n}) + N \cdot (\mathrm{d}(\bar{P}, \mathcal{P}_{P_0,\vec{n}}))^2$$

(ii) En déduire que le plan aux moindres carrés approximant les points passe par $\bar{P}$. On notera donc simplement $f(\vec{n}) = f(\bar{P}, \vec{n})$ la fonction à minimiser.

(iii) Montrer que :

$$f(\vec{n}) = \vec{n}^t \underbrace{\left(\sum_{i=1}^{N} (P_i - \bar{P}) (P_i - \bar{P})^t \right)}_{Z \text{ matrice de corrélation}} \vec{n}$$

(iv) Pourquoi Z est-elle diagonalisable ? En se plaçant dans la base de diagonalisation, montrer que le vecteur $\vec{n}$ minimisant f est le vecteur propre associé à la plus petite valeur propre de Z.

2 TP

Exercice 4 (Implémentation de la SVD et de l'ACP)**.** Vous allez ici implémenter des algorithmes de calcul de la décomposition en valeurs singulières et d'analyse en composantes principales. Vous pouvez tester ces algorithmes en vérifiant qu'ils donnent les mêmes résultats que des implémentations existantes (à une petite erreur numérique près).

(i) Récupérez votre fonction qui implémente l'algorithme des puissances itérées du TP 1 et terminez-la éventuellement.

(ii) Écrivez une fonction `svd1(A, n_iterations)` qui prend en argument une matrice `A` quelconque et un entier `n_iterations` qui implémente l'algorithme des questions 2 et 3 de l'exercice 2 du TD 4, en faisant appel à l'algorithme des puissances itérées sur `n_iterations` itérations.

(iii) Écrivez une fonction `svds(A, s, n_iterations)` qui renvoie la SVD tronquée aux `s` valeurs propres dominantes d'une matrice `A` quelconque en implémentant l'algorithme de la question 5 de l'exercice 2 du TD 2, en faisant appel à la fonction `svd1(... , n_iterations)`.

(iv) Écrivez une fonction `acp(X, k, n_iterations)` qui calcule l'analyse en `k` composantes principales d'un ensemble de points qui forment les lignes de la matrice `X`, en utilisant la fonction `svds` et en implémentant l'algorihme de la question 6 du TD 2.

Exercice 5 (Théorème d'Eckart-Young et approximation d'images)**.** Le théorème d'Eckart-Young, basé sur la SVD, fournit, étant donnée une matrice M de rang n, la matrices de rang $k \leqslant n$ approximant au mieux M (en norme 2, mais aussi en norme de Frobenius).

Or nous manipulons chaque jour, sans le savoir, quantité de matrices : les images. En effet, une image n'est rien d'autre qu'une matrice $n \times m$ de valeurs (entiers compris entre 0 et 255 pour les images en niveaux de gris, triplets d'entiers pour les images couleur).

Le but de cet exercice est d'approximer une image (ie. une matrice d'image) par Eckart-Young pour en construire une simplification (donc débruiter ou extraire les grandes lignes de la forme) ou une compression.

On rappelle qu'étant donnée la SVD d'une matrice A :

$$A = U\Sigma V^t \quad U, V \text{ matrice orthogonales}, \Sigma \text{ diagonale}$$

Pour tout $k \leqslant \text{rang}(A)$, la matrice :

$$A_k = \sum_{i=1}^{k} \sigma_{ii} u_i v_i^t$$

est la matrice de rang k la plus proche de A en norme 2 et en norme de Frobenius.

(i) En utilisant les fonction `imread` et `im2double` (cf. tutorial Matlab/Python), charger la matrice A d'une image niveaux de gris (deux exemples sont fournis dans le matériel de TP, répertoire *data*). Puis convertissez-la en une matrice de flottants.

(ii) Pour k croissant, calculer la matrice A_k et l'afficher en utilisant `imshow`.

(iii) Si la matrice initiale est de taille $n \times m$ et que vous arrêtez la décomposition au rang k, quelle taille est nécessaire pour stocker les informations nécessaires à calculer A_k ? Qu'en pensez-vous ?

Exercice 6 (Application de la SVD à la séparation d'objets en mouvement et du fond fixe dans des vidéos.)**.** On considère qu'une image est une matrice et qu'une vidéo est une séquence d'images. Lorsqu'une vidéo montre une scène fixe (la caméra ne bouge pas) avec des objets en mouvement, une méthode simple permet de séparer le fond fixe et les objets en mouvement de façon assez efficace. Elle consiste à remarquer que le fond fixe est une image dont les pixels sont présents dans la plupart des images de la vidéo, à l'exception des instants où il y a occlusion par un objet. Cette image peut être estimée comme la composante principale de la séquence d'image. La méthode de séparation est donné par l'algorithme 3.

(i) Chargez une des séquences d'images proposées sur Ametice

(ii) Écrivez une fonction qui implémente l'algorithme 3.

(iii) Appliquez cette fonction à la séquence d'image et sauvegardez les images produites.

(iv) Vous pouvez également reconstruire les trois vidéos (original, fond, objets en mouvement) à l'aide du code fourni et les regarder.

(v) Question de réflexion: est-ce que la vidéo obtenue pour le fond est une image fixe ou bien quel genre de variation peut-il y avoir? Expliquez votre réponse mathématiquement.

Commandes Matlab pour générer une vidéo :

```
video_file= VideoWriter('nom_fichier.avi'); %create the video objects
open(video_f); %open the files for writing

writeVideo(video_f,image1); %write the images to files
writeVideo(video_f,image2); %write the images to files
...
writeVideo(video_f,imagen); %write the images to files
close(video_f); %close the files
```

Algorithme 3 Séparation d'objets en mouvement et du fond fixe dans des vidéos par SVD.

Entrées: Une vidéo sous forme d'une séquence d'images $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n \in \mathbb{R}^{d_1 \times d_2}$.

Sorties: Deux vidéos sous forme de séquences d'images $\boldsymbol{F}_1, \ldots, \boldsymbol{F}_n \in \mathbb{R}^{d_1 \times d_2}$ (vidéo du fond) et $\boldsymbol{O}_1, \ldots, \boldsymbol{O}_n \in \mathbb{R}^{d_1 \times d_2}$ (vidéo des objets en mouvement).

1: Pour $i \in \{1, \ldots, n\}$, vectoriser chaque matrice $\boldsymbol{Y}_i$ en un vecteur $\boldsymbol{y}_i \in \mathbb{R}^d$ avec $d = d_1 d_2$ et construire la matrice $\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1^T \\ \vdots \\ \boldsymbol{y}_n^T \end{bmatrix} \in \mathbb{R}^{n \times d}$.

2: Calculer la valeur singulière dominante σ_1 de $\boldsymbol{Y}$ et des vecteurs singuliers à gauche et à droite $\boldsymbol{u}_1$ et $\boldsymbol{v}_1$.

3: $\boldsymbol{F} \leftarrow \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T$

4: $\boldsymbol{O} \leftarrow \boldsymbol{Y} - \boldsymbol{F}$

5: Pour $i \in \{1, \ldots, n\}$, construire les matrices $\boldsymbol{F}_i$ et $\boldsymbol{O}_i$ en redimensionnant respectivement la ligne i des matrices $\boldsymbol{F}$ et $\boldsymbol{O}$ au format $d_1 \times d_2$.

6: **renvoyer** $\{\boldsymbol{F}_i\}_{i=1}^n$, $\{\boldsymbol{O}_i\}_{i=1}^n$

Exercice 7 (ACP et réduction de dimension.)**.** Pour cet exercice, vous pouvez utiliser, au choix, les jeux de données Wine ou Breast Cancer fournis sur Ametice. Wine regroupe 178 échantillons de 3 vins italiens différents (catégories dans `y`) avec 13 descripteurs pour chaque échantillon (matrice `X`). Breast Cancer regroupe 569 mesures sur des patientes du cancer du sein de deux catégories tumeur bénigne/maligne (dans `y`) avec 30 descripteurs pour chaque patiente. Dans les deux cas, on ne peut pas facilement visualiser les données qui sont en dimension 13 ou 30, supérieure à 3.

(i) Chargez les données et affichez les dimensions de la matrice `X` et du vecteur `y` pour contrôler ces valeurs.

(ii) Appliquez l'ACP à `X` puis tracez en 2D les points dans le sous-espace défini par les 2 premières composantes principales; vous distinguerez les points de chaque catégorie par une couleur et/ou un marqueur différents; (optionnel) faites la même chose en 3D.

(iii) les descripteurs sont de nature différente; calculez la variance de chaque descripteur: vous devriez constater que les descripteurs ont des variances différentes; un descripteur avec une petite variance pèse ainsi très peu dans le choix des composantes principales, alors qu'il peut contenir de l'information importante;

(iv) dans chaque colonne de `X`, soustrayez la moyenne et divisez par l'écart-type de cette colonne pour que chaque descripteur ait une variance égale à 1; puis appliquez à nouveau l'ACP, visualisez le résultat et comparez-le à votre première visualisation.