MIAM (ICM1A-ICM2A), UP1, TP1 : Partitionnement

Ce travail pratique numérique sera l'occasion d'implémenter la méthode des K-moyennes, et de l'appliquer au partitionnement d'un ensemble de microstructures représentées par des vecteurs contenant les coordonnées des centres de fibres, ainsi que leur rayon.

Ce TP est évalué. Le travail est à faire individuellement. **Si vous choisissez ce TP** : il faudra déposer, au plus tard deux semaines après la fin de la séance « réseaux de neurones » (dernière séance de TP), le code que vous aurez fait, ainsi qu'un compte-rendu sous format `pdf`. Il est également possible de rendre les deux simultanément dans un notebook jupyter. Dans ce cas, il faut rendre à la fois la version `ipynb` et `pdf`. **Si vous ne choisissez pas ce TP** : vous devrez faire apparaître en préambule du compte-rendu du TP que vous choisissez un paragraphe expliquant la démarche de ce TP.

La forme du code est importante. Dans un contexte « ingénieur », un code mal commenté, mal organisé ou avec des noms de fonctions ou de variables mal choisis a vocation à être une source de stress et d'erreurs pour les contributeurs qui s'y pencheront après vous. Dans le contexte présent, un tel code a simplement vocation à être une source de points en moins pour vous. Il en va de même pour le compte-rendu.

Forme attendue pour le compte-rendu : Il doit pouvoir se lire indépendamment du sujet. Il comprendra une rapide introduction expliquant l'objet d'étude, et surtout présentera les résultats de l'exécution de votre code (images ou matrices si elles sont petites), ainsi qu'une description de ces résultats (paramètres utilisés etc) et un commentaire sur ces résultats. Le compte-rendu est organisé en sections (les mêmes que dans le sujet) et se finit par une conclusion, comme rappelé en fin de document. Si le compte-rendu n'est pas un notebook jupyter, le code ne doit pas y apparaître. Il faut également faire apparaître un court résumé des TPs non-choisis dans le compte-rendu du TP choisi.

Mise en place

En plus de ce sujet, vous aurez besoin des résultats du TD de partitionnement, ainsi que du support de cours pour pouvoir mener à bien le travail demandé. Vous avez également à disposition une bibliothèque de fonctions regroupées dans le fichier `partitionnement_ressources`. Cette bibliothèque devra être importée dans le code que vous créerez, de préférence sous un diminutif bien choisi.

1 Implémentation de l'algorithme des K-moyennes

On rappelle l'algorithme donné en cours.

Algorithme 1 : Kmoyennes($\{\mathsf{x}_n\}_{n=1..N}$, K)

Choisir $\{b_k\}_{k=1..K}$ par exemple par tirage aléatoire (initialisation)
for $iter = 1, 2, \ldots, niter$ (itérations du pt fixe) **do**
  for $i = 1, 2, \ldots, N$ (boucle sur les éléments) **do**
    $\mathcal{E}(n) \leftarrow \underset{k\in[\![1;K]\!]}{\arg\min} \|\mathsf{x}_n - b_k\|^2$ (lier chaque vecteur au barycentre le plus proche)
  end
  for $k = 1, 2, \ldots, K$ (boucle sur les ensembles) **do**
    $b_k \leftarrow \dfrac{1}{\mathrm{Card}(E_k)} \displaystyle\sum_{n|\mathcal{E}(n)=k} \mathsf{x}_n$ (calcul des barycentres)
  end
end
Retourner $\{b_k\}_{k=1..K}$ et $\{\mathcal{E}(n)\}_{n=1..N}$

Pour l'initialisation, on propose trois méthodes différentes, qui sont données dans la librairie ressource. La fonction `pickInitialization` choisit aléatoirement K vecteurs de la base de données, tandis que la fonction `statInitialization` effectue un tirage aléatoire suivant une loi uniforme ayant les mêmes moments statistiques que la vase de données. La fonction `pointInitialization` crée une première partition aléatoire et calcule ses barycentres.

Chacune de ces fonctions a un argument facultatif, `setSeed`, qui permet de fixer la racine du générateur de variables aléatoires, de façon à rendre le processus déterministe. Ceci peut être intéressant lors des phases de debug, et lors de l'étude de la méthode.

À faire : Implémenter la fonction appliquant l'algorithme des K-moyennes à une famille de vecteurs. Dans cette section, l'algorithme sera testé sur des vecteurs de $\mathbb{R}^2$, mais, en vue de la suite du TP, il est important que l'algorithme accepte des vecteurs de taille quelconque. L'ensemble de vecteurs sera donné sous la forme d'une matrice dont chaque colonne sera un vecteur.

Le nombre de partitions et le nombre d'itérations sont des paramètres de la méthode, qui seront des arguments de la fonction à écrire.

On va à présent tester cet algorithme avec une distribution synthétique de points dans $\mathbb{R}^2$.

À faire : Générer une série de vecteurs dans le plan à l'aide de la fonction `generateSamples2D`, et lui appliquer l'algorithme des K-moyennes précédemment développé. On utilisera ensuite la fonction `plot2DClusters` pour visualiser les partitions.

La fonction `generateSamples2D` a un argument facultatif, `setSeed`, qui permet de la rendre déterministe. Attention : quand la racine du générateur de variables aléatoires a été fixée une fois dans le code, toutes les fonctions deviennent déterministes.

À faire : Tester l'algorithme sur différentes populations de vecteurs (en changeant les arguments de `generateSamples2D`) et avec différents jeux de paramètres K et $niter$. En particulier, on observera la convergence de l'algorithme.

Bonus : Implémenter un critère d'arrêt prématuré pertinent pour la fonction de partitionnement.

2 Application de l'algorithme aux données brutes

Dans cette section, on applique l'algorithme des K-moyennes aux données de microstructures non-renumérotées. Afin d'évaluer l'algorithme de classification, les microstructures ont été rangées dans trois dossiers distincts. Une méthode de partitionnement « parfaite » serait capable de trouver trois partitions dans lesquelles se retrouveraient les microstructures issues de chacun des trois dossiers. Une méthode de partitionnement un peu inférieure parviendrait à séparer la base de données en deux partitions. Dans l'une se retrouveraient les microstructures issues de l'un des dossiers, et dans l'autre seraient mélangées les autres microstructures.

À faire : Utiliser la fonction `readMicrostructures` pour lire les données brutes de microstructure (sans re-numérotation des fibres). Appliquer l'algorithme des K-moyennes à la famille de vecteurs résultant de cette lecture de données. On fera varier les deux paramètres de la méthode.

Attention : dans cette section, on ne peut plus afficher les partitions à l'aide de la fonction `plot2DClusters`, puisqu'on n'est plus en 2D.

Le programme de lecture des données est construit de façon à ce que des vecteurs issus des trois dossiers se suivent toujours dans le même ordre. On peut donc facilement connaître le partitionnement souhaité à-partir du reste de la division entière de l'indice d'un vecteur par 3.

À faire : Construire une matrice (appelée matrice de confusion) de taille $3 \times k$ donnant pour chaque dossier le nombre de vecteurs de ce dossier ayant été placés dans chaque partition. Par exemple :

$$\begin{pmatrix} 1 & 5 \\ 3 & 3 \\ 4 & 2 \end{pmatrix}$$

signifie que 5 vecteurs du premier dossier ont été placés dans la partition 2 un seul dans la partition 1, 3 vecteurs du second dossier ont été placés dans la partition 1 et autant dans la partition 2. 4 vecteurs du premier dossier ont été placés dans la partition 1 et 2 dans la partition 2.

À l'aide de cette matrice, évaluer la méthode de partitionnement. Les résultats sont-ils pertinents ? Donner une explication.

Remarque. *À toutes fins utiles, la fonction* `plotMicrostruct` *permet d'afficher graphiquement une microstructure donnée à-partir du vecteur correspondant.*

3 Application après re-numérotation

On explique dans le poly de cours qu'il est nécessaire de re-numéroter les fibres afin que le partitionnement soit pertinent. Ceci a été fait et la fonction `readMicrostructures` peut être utilisée (second argument) pour récupérer les microstructures renumérotées.

À faire : Appliquer l'algorithme des K-moyennes aux microstructures renumérotées. Analyser et commenter les résultats.

Bonus : Implémenter la méthode de renumérotation, présentée dans le polycopié de cours (p5), qui a été utilisée pour générer les microstructures re-numérotées.

4 Tests d'un critère en déformation

Comme vu en TD, on va maintenant tenter d'utiliser la norme du gradient de la déformation de la grille de référence comme critère permettant de partitionner la population de microstructures.

À faire : Pour chaque microstructure, calculer le tenseur gradient $\mathbb{H}$ tel que vu en TD en chaque point de la grille de référence. On utilisera la méthode des différences finies centrées, et on n'oubliera pas que la microstructure est périodique (pour le calcul des gradients sur les bords).

On va déterminer pour chaque microstructure la norme du gradient de la déformation de la grille de référence comme suit :

$$\|\mathbb{H}\|^2 = \sum_{i=1}^{n} \Big(H_{11}(X_n)^2 + H_{12}(X_n)^2 + H_{21}(X_n)^2 + H_{22}(X_n)^2 \Big) \tag{1}$$

À faire : Tester l'algorithme de partitionnement en utilisant la norme du tenseur gradient comme « vecteur » (à un seul ddl) donnée. Analyser les résultats.

Bonus : Tester d'autres variantes de la méthode des différences finies. Tester aussi la nécessité de faire l'hypothèse de périodicité.

5 Critère à base de distance au plus proche voisin

Dans le domaine des microstructures fibreuses, on sait de longue date que la distance d'une fibre à son plus proche voisin est un paramètre important de la microstructure. On donne la fonction `computeDistances`, qui calcule ces distances pour toutes les fibres de la microstructures. On se propose d'utiliser cette information pour partitionner la population.

À faire : Tester l'algorithme de partitionnement en utilisant la moyenne des distances comme critère.

Bonus : Implémenter la méthode de partitionnement la plus fiable possible en utilisant au mieux les différents ingrédients introduits dans ce TP.

Conclusion

À faire : Proposer une conclusion synthétique, claire et pertinente à ce travail. On s'interrogera notamment (mais pas nécessairement exclusivement) sur les questions suivantes :

- Les avantages et inconvénients des méthodes non-déterministes
- L'importance de la préparation des données, et notamment la place de l'expertise métier dans cette étape
- Le compromis entre quantité d'information et qualité de celle-ci