PROJET TUTORÉ

Analyse et implémentation de l'algorithme de l'article intitulé :

On Spectral Clustering : Analysis and an algorithm

Authors : Andrew Y. Ng, Michael I. Jordan andYair Weiss

rédigé.e par : **Sarah CHIKHAOUI** et **Aliou BOUBA**
Supervision : **Prof. Joan Alexis GLAUNES**

UNIVERSITE PARIS CITE

2022-2023

1er janvier 2023

Plan de travail

Introduction

1 Méthodologie

1.1 Implémentation

1.2 Analyse de l'algorithme

1.2.1 Cas idéal

1.2.1 Cas général

1.2.2.1 Interprétation du point de vue de la théorie des graphes

1.3 Méthode possible du choix de sigma

2 Résultats

2.1 Simulation

2.2 Comparaison avec K-means et EM-algorithm

3 Discussion

Remerciements

Références

Introduction

L'apprentissage automatique (machine learning en anglais) est un champ d'étude de l'intelligence artificielle qui vise à donner aux machines la capacité d'apprendre à partir de données, via des modèles mathématiques et statistiques [2, 4, 7]. Ces modèles peuvent être classés en deux catégories : supervisé et non supervisé. Dans le cadre de l'apprentissage non supervisé, les deux principales méthodes utilisées sont ici le regroupement (clustering) et la réduction de la dimensionnalité. Le clustering est l'une des techniques les plus utilisées pour l'analyse exploratoire des données. Son objectif est de séparer des données en groupes homogènes ayant des caractéristiques communes [6]. Ainsi elle divise les points des données en plusieurs groupes de sorte que les points d'un même groupe soient similaires tandis que ceux de différents groupes soient différents les uns des autres. Parmi les algorithmes classiques de clustering, nous retrouvons expectation-maximization algorithm (EM-algorithm), et celui des k-means (k-moyennes). La méthode des k-moyennes est rapide et donne de bons résultats pour des variables quantitatives ou ordinales. Celle de l'EM est en fait l'un des algorithmes les plus rapides pour apprendre les modèles de mélange. Ces deux algorithmes sont simples et capables de traiter de gros volumes des données. Cependant ils pâtissent de plusieurs inconvénients. D'une part, ils imposent des hypothèses simplificatrices sévères comme l'idée selon laquelle chaque cluster suit une loi normale. D'autre part, la log-vraisemblance des données complètes peut avoir plusieurs extremas locaux, ce qui implique de devoir répéter plusieurs fois la méthode itérative à la recherche d'une bonne solution, prohibitif pour des données de grande taille [5]. Ces contraintes ont ainsi motivé le recours à différentes alternatives comme les méthodes spectrales pour le regroupement appelé spectral clustering [1]. Ainsi dans le cadre de notre projet tutoré, nous nous sommes intéressés aux travaux de [5] sur l'algorithme du spectral clustering. Il est question pour nous dans le cadre de ce travail d'implémenter, analyser et de simuler cet algorithme sur différents (au moins deux) ensembles des zones de partitionnement non convexes (le jeux de données MNIST et MLBENCH). Il s'agira d'interpréter les résultats en les comparant à ceux d'autres algorithmes comme l'EM et de discuter de la méthode sous-jacente au spectral clustering. Il convient de rappeler que l'article soumis à notre attention pour ce projet tutoré n'a pas été étudié totalement, donc c'est une étude partielle qui a été fait ici.

1 Méthodologie

L'algorithme du spectral clustering est un algorithme de partitionnement des données reposant sur l'algèbre linéaire et la théorie spectrale des graphes. En effet, l'algorithme utilise le graphe de similarité pour traiter le problème du regroupement. Soit un ensemble de N observations, il est possible d'en obtenir une représentation détaillée sous la forme d'un graphe pondéré où chaque observations correspond à un nœud relié aux autres par des arcs pondérés. La matrice de poids des arcs correspond à la matrice dite de similarité des observations ou nœuds à partir de laquelle sera calculée son spectre afin de partitionner l'ensemble des observations dans un espace de plus faible dimension. Ainsi le problème se restreint à un souci de partitionnement de graphe pour lequel les nœuds d'un même groupe sont similaires et les nœuds appartenant à des groupes différents ne le sont pas. Il s'agit alors de détecter des communautés de nœuds, soit, des nœuds très connectés entre eux que l'on appelle aussi composantes connexes.

Le partitionnement spectral utilise le plus souvent les vecteurs propres d'une matrice de similarités. Pour des algorithmes classiques comme celui des k-means, à l'inverse, le partitionnement de données spectral a pour objectif de créer différents groupes ayant un maximum de points communs.

1.1 Implémentation

La méthodologie proposé par [5] ici, repose pour l'essentiel sur les travaux de [8] et [3]. Ils proposent une technique particulière d'utilisation simultanée des k vecteurs propres en formulant des conditions pour lesquelles l'algorithme marchera correctement. L'algorithme proposé par [5] est composé de six étapes dont l'étape fondamentale est la construction de la matrice de similarité.

La matrice de similarité est construite pour mesurer la distance entre deux objets. Toutefois le choix de cette fonction de similarité dépend de la provenance et mais aussi et surtout du type des données. Cependant, les formulations les plus utilisées sont : le *Noyau Gaussien* et la *Distance cosinus*.
Dans le cadre de ce travail, nous utiliserons la construction de la matrice de similarité avec la technique du *Noyau Gaussien*.

Nous gardons les notations utilisées dans les travaux de [5] pour plus de conformité et de clarté.

— On considère un ensemble des points $S = \{s_1, \dots s_n\}$ dans $\mathbb{R}^l$ quon veut partitionner en plusieurs k groupes. Pour le faire, nous construirons une matrice d'affinité (ou de similarité) A dans $\mathbb{R}^{n \times n}$ définie par

$$A_{ij} = \begin{cases} exp\left(- \frac{d^2}{2\sigma^2} \right) & \text{si } i \neq j, \\ 0 & \text{sinon .} \end{cases} \tag{1}$$

où avec $d = ||s_i - s_j||$ est la distance euclidienne et σ, un paramètre d'échelle dont la valeur est fixé par l'utilisateur (à priori mais un détails sur le calcul de ce dernier sera abordé plus tard voir (section 1.3)). Donc, ici on obtient une matrice symétrique par rapport à la diagonale avec des zéros sur cette dernière.
Ainsi, nous obtenons une matrice de cette forme

$$A = \begin{bmatrix} 0 & A_{12} & \dots & A_{1n} \\ A_{21} & 0 & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ A_{n_1} & \dots & \dots & 0 \end{bmatrix}$$

— Dans cette étape nous allons construire une matrice des degrés diagonale D qui a uniquement des éléments non nuls sur la diagonale et zéros partout ailleurs. Chaque élément de la diagonale (i, i) est la somme de la i^{eme} ligne de la matrice de similarité A construite précédemment.
Ainsi la matrice diagonale D s'écrit sous la forme

$$\begin{aligned} D_{ii} =& diag\Big(D_{11}, \dots, D_{nn}\Big) \\ =& diag\Big(\sum_{j=1}^{n} A_{ij}\Big) \\ =& \begin{pmatrix} \sum_{j=1}^{n} A_{1j} & 0 & \dots & 0 \\ 0 & \sum_{j=1}^{n} A_{2j} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sum_{j=1}^{n} A_{nj} \end{pmatrix}. \end{aligned} \tag{2}$$

Comme on vient de définir notre matrice diagonale D, on peut construire la matrice laplacienne L en utilisant une des nombreuses normalisations (normalisation par division, normalisation par division symétrique, normalisation additive,...) présentes dans la littérature. Dans le cadre de ce travail, on utilisera la normalisation par division symétrique. Cette technique consiste à calculer la racine carré de l'inverse de la matrice diagonale D et après faire le produit avec la matrice de similarité A.

$$L = D^{-1/2} A D^{-1/2},$$

où A est la matrice d'affinité construite plus haut et D la matrice diagonale.

— Dans l'étape ci, nous calculerons et extrairons les k premiers vecteurs propres $x_1, x_2, \dots, x_k$, associés aux plus petites valeurs propres de la matrice laplacienne L, et nous formerons la matrice X de taille $n \times k$ dont les colonnes sont nos vecteurs propres, $X = [x_1, x_2, \dots, x_k] \in \mathbb{R}^{n \times k}$.

— Ici, on forme la matrice Y à partir de la matrice des vecteurs propres obtenues à l'étape précédente, en normalisant chaque ligne de X pour avoir une norme euclidienne 1.

$$Y_{ij} = X_{ij} / \big(\sum_{j} X_{ij}^2\big)^{1/2}$$

.

— Dans l'étape précédente, nous avons normalisé chaque ligne de X pour avoir une matrice Y. L'étape ci consiste à considérer chaque ligne de Y comme représentant un point. Alors ici nous traiterons chaque ligne de

la matrice Y comme un point dans $\mathbb{R}^k$, en les partitionnant en k groupes en utilisant l'algorithme de classification non-supervisée $K-means$, par ailleurs choisie dans le cadre de ce travail.

— Dans cette dernière étape, on affecte le point original s_i au groupe j si et seulement si la ligne i de la matrice Y a été affectée au groupe j.

1.2 Analyse de l'algorithme

1.2.1 Cas idéal

Afin de mieux comprendre notre algorithme, il est important à plus d'un titre d'étudier son comportement dans le cas idéal. Le cas idéal suppose ici que tous les points situés dans les différents groupes (classes) sont infiniment éloignés les uns des autres. Dans ce cas idéal nous supposons qu'on dispose de trois groupes $k = 3$ où n_1, n_2 et n_3 sont leurs tailles respectives. En effet, en éloignant infiniment ces groupes, la distance entre ces derniers devient trop grande (au voisinage de l'infini). Donc, dans notre matrice d'affinité, les blocs correspondant aux clusters différents deviennent nuls car

$$A_{ij} = exp\Big(-\frac{d^2}{2\sigma^2}\Big) = exp\Big(-\frac{\infty^2}{2\sigma^2}\Big) = exp\Big(-\infty\Big) = 0$$

où $d = ||s_i - s_j||$. Puisque les groupes sont infiniment éloignés les uns des autres, ceci implique par exemple que les points s_i et s_j ne sont pas dans le même groupe.

Notons $\hat{A}_{ij}$ comme notre nouvelle matrice d'affinité dans le cas idéal, donc

$$\hat{A}_{ij} = \begin{cases} A_{ij} & \text{si } x_i \text{ et } x_j \text{ sont dans le même groupe,} \\ 0 & \text{sinon .} \end{cases} \tag{3}$$

Ainsi nous pouvons définir la matrice du groupe 1, groupe 2 et groupe 3 respectivement par

$$A^{(11)} = \begin{bmatrix} 0 & A_{1,2} & \dots & A_{1,n_1} \\ A_{2,1} & 0 & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ A_{n_1,1} & \dots & \dots & 0 \end{bmatrix},$$

$$A^{(22)} = \begin{bmatrix} 0 & A_{n_1+1,n_1+2} & \dots & A_{n_1+1,n_1+n_2} \\ A_{n_1+2,n_1+1} & 0 & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ A_{n_1+n_2,n_1+1} & \dots & \dots & 0 \end{bmatrix}$$

et

$$A^{(33)} = \begin{bmatrix} 0 & A_{n_1+n_2+1,n_1+n_2+2} & \dots & A_{n_1+n_2+1,n} \\ A_{n_1+n_2+2,n_1+n_2+1} & 0 & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,n_1+n_2+1} & \dots & \dots & 0 \end{bmatrix}$$

.
Les matrices $A^{(11)}, A^{(22)}$ et $A^{(33)}$ constituent ce qu'on appelle les matrices intra-groupes (ou intra-clusters).

Nous pouvons donc construire maintenant notre nouvelle matrice d'affinité qui est constituée des matrices intra-groupes définies ci-haut.

$$\hat{A} = \begin{bmatrix} A^{(11)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A^{(22)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & A^{(33)} \end{bmatrix}$$

où $\mathbf{0}$ désigne les matrices inter-groupes (donc nulle car les points se trouvant dans des groupes différents).

La construction des autres à savoir $\hat{D}, \hat{L}, \hat{X}$ et $\hat{Y}$ suivra les mêmes astuces et étapes présentées dans la section de l'implémentation de l'algorithme mais en utilisant dans le cas ci notre nouvelle matrice d'affinité intra-groupe $\hat{A}$.

Ainsi pour la matrice des degrés, $\hat{D}$, on a :

$$\hat{D}^{(ii)} = diag\left(\hat{D}^{(11)}, \hat{D}^{(22)}, \hat{D}^{(33)}\right) \tag{4}$$

Connaissant déjà les matrices $\hat{D}$ et $\hat{A}$, on déduit la matrice laplacienne $\hat{L}$ en faisant le produit des matrices précédentes avec quelques transformations sur la matrice $\hat{D}$. On obtient donc,

$$\begin{aligned} \hat{L}^{(ii)} &= (\hat{D}^{(ii)})^{-1/2} A^{(ii)} (\hat{D}^{(ii)})^{-1/2} \\ \hat{L} &= \begin{bmatrix} \hat{L}^{(11)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \hat{L}^{(22)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \hat{L}^{(33)} \end{bmatrix} \end{aligned} \tag{5}$$

où $\mathbf{0}$ désigne les matrices inter-groupes.

En effet, pour construire la matrice $\hat{X}$, on a besoin de chercher les vecteurs propres associés aux valeurs propres de la matrice $\hat{L} \in \mathbb{R}^{n \times n}$. Pour ce faire, on calcul les valeurs propres de chaque sous-blocs $\hat{L}^{(11)}$, $\hat{L}^{(22)}$ et $\hat{L}^{(33)}$ ayant pour dimensions respectives $n_1 \times n_1, n_2 \times n_2$ et $n_3 \times n_3$. Donc chaque sous-bloc possède respectivement n_1, n_2 et n_3 valeurs propres.
Ainsi dans chaque sous-bloc, on prend la plus grande valeur propre et on l'associe à un vecteur propre. D'où pour le sous-bloc $\hat{L}^{(11)}$, la plus grande valeur est $\lambda_{max}^{(11)} = 1$ associé au vecteur propre $x_1^{(1)}$. De même les sous-blocs $\hat{L}^{(22)}$ et $\hat{L}^{(33)}$ ont pour plus grande propre valeur $\lambda_{max}^{(22)} = 1$ et $\lambda_{max}^{(33)} = 1$ associé aux vecteurs propres $x_1^{(2)}$ et $x_1^{(3)}$ respectivement.

Puisque la matrice $\hat{L}$ est constituée des sous-blocs $\hat{L}^{(11)}$, $\hat{L}^{(22)}$ et $\hat{L}^{(33)}$ (et deux matrices supérieures et inférieures nulles) dont chaque sous-bloc a une valeur propre maximale qui est 1 et associée à un vecteur propre $x_1^{(i)}$, alors la valeur propre maximale associée à la matrice $\hat{L}$ est donc le maximum des valeurs propres maximales de $\hat{L}^{(11)}$, $\hat{L}^{(22)}$ et $\hat{L}^{(33)}$ et elle a pour multiplicité 3.
D'où

$$\lambda_{max} = \max\left(\lambda_{max}^{(11)}, \lambda_{max}^{(22)}, \lambda_{max}^{(33)}\right) = 1.$$

Ainsi les vecteurs propres associés à la plus grande valeur propre de $\hat{L}$ sont

$$\tilde{x}_1^{(1)}, \quad \tilde{x}_1^{(2)} \quad \text{et} \quad \tilde{x}_1^{(3)}$$

avec $\tilde{x}_1^{(1)} = \left(x_1^{(1)}, \mathbf{0}, \mathbf{0}\right)^T$, $\tilde{x}_1^{(2)} = \left(\mathbf{0}, x_1^{(2)}, \mathbf{0}\right)^T$ et $\tilde{x}_1^{(3)} = \left(\mathbf{0}, \mathbf{0}, x_1^{(3)}\right)^T$ où les indices représentent la valeur propre et les exposants le groupe auquel appartient le vecteur propre. De plus comme nous sommes dans le cas idéal, on considérera seulement les trois premiers vecteurs propres.

Pour notre matrice, vu que nous sommes toujours dans le cas idéal, nous considérons les trois premiers vecteurs propres de la matrice $k = 3$, $\hat{L}$, comme nous avons déjà toutes les informations nécessaires sur ces vecteurs propres, il suffit donc de faire comme dans la *section* 1.1, on stocke tout simplement ces vecteurs propres dans la matrice colonne de $\hat{X}$.

D'où

$$\hat{X} = \begin{bmatrix} x_1^{(1)} & \vec{\mathbf{0}} & \vec{\mathbf{0}} \\ \vec{\mathbf{0}} & x_1^{(2)} & \vec{\mathbf{0}} \\ \vec{\mathbf{0}} & \vec{\mathbf{0}} & x_1^{(3)} \end{bmatrix} \in \mathbb{R}^{n \times 3}.$$

car les vecteurs propres $x_1^{(i)} \in \mathbb{R}^{n_i}$

Dans cette dernière étape de l'analyse du cas idéal, nous utiliserons la formule de la normalisation vu à la *section* 1.1 en utilisant cette fois ci la matrice $\hat{X}$. On obtient donc la matrice normalisée suivante

$$\hat{Y}^{(1)} = \begin{bmatrix} \vec{1} & \vec{\mathbf{0}} & \vec{\mathbf{0}} \end{bmatrix} \in \mathbb{R}^{n_1 \times 3},$$
$$\hat{Y}^{(2)} = \begin{bmatrix} \vec{\mathbf{0}} & \vec{1} & \vec{\mathbf{0}} \end{bmatrix} \in \mathbb{R}^{n_2 \times 3}$$

et

$$\hat{Y}^{(3)} = \begin{bmatrix} \vec{\mathbf{0}} & \vec{\mathbf{0}} & \vec{1} \end{bmatrix} \in \mathbb{R}^{n_3 \times 3}.$$

D'où,

$$\hat{Y} = \begin{bmatrix} \hat{Y}^{(1)} \\ \hat{Y}^{(2)} \\ \hat{Y}^{(3)} \end{bmatrix} = \begin{bmatrix} \vec{1} & \vec{\mathbf{0}} & \vec{\mathbf{0}} \\ \vec{\mathbf{0}} & \vec{1} & \vec{\mathbf{0}} \\ \vec{\mathbf{0}} & \vec{\mathbf{0}} & \vec{1} \end{bmatrix}$$

avec $\hat{Y}^{(1)}$, $\hat{Y}^{(2)}$ et $\hat{Y}^{(3)}$ représentant respectivement les matrices normalisées associées à chaque groupe. Ainsi Tous les points de nos données deviennent les $\hat{Y}^{(i)} \in \mathbb{R}^{n_i \times k}$

où $1/3$ se trouvant dans $(1, 0, 0)$, $1/3$ se trouvant dans $(0, 1, 0)$ et $1/3$ se trouvant dans $(0, 0, 1)$. Autrement dit, il y a 3 points mutuellement orthogonaux sur la surface de la k-sphère unitaire autour de laquelle les lignes de $\hat{Y}^{(1)}$ se regrouperont.

1.2.2 Cas général

Dans cette section, nous nous intéresserons à l'analyse de l'algorithme dans le cas où $A_{ij} \neq 0$ pour $i \neq j$ (les blocs hors diagonaux de A sont non nuls). En effet, dans le cas où ait lieu une perturbation $E = A - \hat{A}$, du cas idéal $\hat{A}$, (donnant lieu à $A = \hat{A} + A$) on cherche à connaître les conditions permettant un regroupement des lignes de Y, similaire à celui établi par celles de $\hat{Y}$. Plus formellement, on cherche ce qui permet aux vecteurs propres de L de se rapprocher de ceux de $\hat{L}$, où $L = \hat{L} + E$ (L est considérée à présent comme une version

perturbée de $\hat{L}$). D'un point de vue du partitionnement de graphe, cela revient à chercher les groupes de nœuds qui obtiennent des connexions intra-groupes importantes. Nous reparlerons de la relation entre le clustering spectral et le partitionnement de graphe dans la section interprétation du point de vue de la théorie des graphes.

Pour répondre à notre interrogation, considérons d'abord la notion d'écart spectral d'une matrice qui n'est rien d'autre que la différence entre ses deux plus grandes valeurs propres. Or la théorie des perturbations matricielles indique que la stabilité des vecteurs propres d'une matrice est déterminée par l'écart spectral. Plus formellement, cela signifie que les trois plus grands vecteurs propres de $\hat{L}$ seront d'autant plus stable que l'écart spectral $\delta = |\lambda_3 - \lambda_4|$ sera grand, lorsque $\hat{L}$ subit de petites modifications. Or les valeurs propres de $\hat{L}$ sont l'union des valeurs propres de $L^{(11)}$, $L^{(22)}$ et $L^{(33)}$ où la j^{eme} plus grande valeur propre de $L^{(ii)}$ vaut $\lambda_j = \max_{i\in[1,k]} \lambda_{j-2}(i) = \max_{i\in[1,k]} \left\{\lambda_1^{(1)}, \ldots, \lambda_1^{(k)}\right\}$ et où $\lambda_3 = 1$. Ainsi un écart spectral δ de $\hat{L}$ grand équivaut à une valeur de λ_4 très éloignée de 1. On a alors l'hypothèse $A.1$ voir ([5]) suivante : $\exists\ \delta > 0$ tel que, pour tout $i = 1, \ldots, k$, $\lambda_2^i < 1 - \delta$ où λ_2^i ne dépend que de $L^{(ii)}$, qui à son tour ne dépend que de $A^{(ii)} = \hat{A}^{(ii)}$, soit la matrice des similarités intra-cluster pour chaque cluster S_i. Dans le contexte du clustering l'hypothèse sur λ_2^i s'interprète naturellement : elle capture l'idée selon laquelle si nous voulons qu'un algorithme trouve la partition en 3 clusters S_1, S_2 et S_3, alors nous exigeons une connexité forte de chacun de ces sous ensembles. Prenons par exemple le cas où le cluster $S_1 = S_{1.1} \cup S_{1.2}$, se décline en l'union de deux sous clusters bien séparés, alors $S = S_{1.1} \cup S_{1.2} \cup S_2 \cup S_3$ ressemble à l'union d'au moins quatre amas. Or pour une telle partition de l'ensemble des données, il serait déraisonnable d'attendre de l'algorithme qu'il la devine correctement, soit celle en trois en 3 partitions S_1, S_2 et S_3. Ce lien entre l'espace propre et la cohésion (au sens connexité) au sein de chaque cluster peut être formalisé aussi bien du point de vue de la théorie des graphes expanseurs que de celui des marches aléatoires.

1.2.2.1 Interprétation du point de vue de la théorie des graphes

Afin d'étayer notre interprétation, nous commencerons par introduire quelques définitions issus de la théorie des graphes spectraux.

- **Connexité** : Un graphe est dit connexe si on peut toujours relier deux de n'importe quel de ses sommets par un chemin, éventuellement réduit à une arrête. Ainsi il possède des sous-graphe induit connexe maximal que l'on appelle composante connexe.
- **Frontière d'un sous-ensemble de sommets d'un graphe** : Soit
  a. $G = (V,\ E)$ un graphe où les éléments de V sont appelés sommets, et ceux de E sont nommés arrêtes.
  b. $A \subset V$ un sous-ensemble de sommets. On définit alors la frontière de A, notée ∂A, comme étant l'ensemble des arêtes reliant un sommet de A à un sommet du complémentaire de A. Notons que cet ensemble ∂A peut aussi être défini comme la coupe entre A et A^c, que nous noterons l'ensemble B.
- **Graphe biparti** : Un graphe est dit biparti s'il existe une partition de son ensemble de sommets en deux sous-ensembles L et R telle que chaque

arête ait une extrémité dans L et l'autre dans R.

— **Constante de** *Cheeger* : Dans un graphe $G(V,\ E)$, on définit la constante de Cheeger par :

$$h(G) = \inf_{F \subset E,\ |F| \leq 1/2|E|} \frac{|\partial A|}{|F|}$$

où $|\partial A|$ est le cardinal de la frontière de A. Cette constante prend la valeur du plus petit des quotients associés à tous les partages possibles du graphe en deux morceaux, soit, à toutes les versions possibles du graphe biparti. Ainsi, elle permet de mesurer la connexité d'un graphe et de manière intuitive, on comprend qu'elle est non nulle si et seulement si le graphe est connexe.

— **Densité d'un graphe** : La densité d'un graphe est le rapport entre le nombre d'arêtes qu'il possède et le nombre d'arêtes maximales qu'il peut avoir. Ce paramètre mesure donc un taux d'arêtes que l'on appelle densité.

— **Graphe ε-expanseur** Soit un $G(V,\ E)$ graphe non orienté à n sommets. Ce graphe est expanseur de facteur $\epsilon > 0$, si $\exists A \subset V$, un sous-ensemble de sommets A tel que $|A| = Card(A) \leq n/2 : \partial A \geq \varepsilon \times |A|$. Ainsi on comprend que tous graphe connexe, à n sommets, est $2/n$-expanseur car il existe toujours au moins une arête de la coupe pour s'échapper d'une partie du graphe (sinon il ne serait pas connexe).

— **Degré d'un sommet/graphe** : Le degré d'un sommet $v \in V$ est son nombre de voisins : $deg\ v = |w \in V|(v, w) \in E|$, tandis que celui d'un graphe est défini par $deg\ G = \max\{deg\ v | v \in V\}$.

— **Marche aléatoire sur un graphe** G : La marche aléatoire sur G est la chaîne de Markov d'espace d'états V, et de matrice de transition :

$$P(v,\ w) = \begin{cases} 1/deg\ v & \text{si } v \text{ et } w \text{ sont voisins,} \\ 0 & \text{sinon .} \end{cases} \tag{6}$$

Ainsi, une transition de cette chaîne de Markov à partir d'un sommet $v \in V$ est réalisée en choisissant de manière uniforme un sommet parmi les voisins de v, et en sautant vers ce voisin.

La définition d'un graphe expanseur permet de décrire un graphe peu dense dont les nœuds sont bien connectés et c'est ce dont le spectral clustering cherche à réaliser. En effet, du point de vue de la théorie des graphes spectraux, l'idée est de trouver la meilleur valeur d'ε pour laquelle la constante de Cheeger sera faible afin d'obtenir des sous-graphes à forte connexité (ou composantes connexes) mais de tel que le graphe reste le moins dense possible. Ces composantes forment les clusters de l'ensemble des données.

De manière similaire, d'un point de vue des processus aléatoires sur le graphe, le partitionnement issus du spectral clustering peut être vu comme une marche aléatoire. En effet, imaginons une puce se déplaçant à travers le graphe, d'un sommet vers l'un de ses plus proches voisins en un temps t et avec une équi-probabilité de transition p-cluster. Imaginons alors qu'à l'issue de son parcours, la puce ait visité plusieurs fois certains sommets de sortes qu'ils forment des communautés de sommets souvent visités par la puce. Nous pourrions alors soupçonner chez la puce une certaine inertie à passer d'une communauté à une

autre. Or cela sera vrai si la probabilité de transition d'un sommet à un sommet issu d'un autre cluster est faible.

De manière plus formelle, nous pouvons admettre qu'une marche aléatoire modélisée l'exploration dans un graphe d'une particule se déplaçant de manière aléatoire. Dans le cas du spectral clustering, le partitionnement du graphe en plusieurs composantes connexes correspondant au même nombre à des marches aléatoires. Ces dernières ont chacune une probabilité de transition (d'un point à un autre du cluster concerné) proportionnelle aux coefficients A_{ij} de notre matrice de départ. Ainsi, pour une telle marche définie sur les points de l'un des clusters, la matrice de transition correspondante a une valeur propre secondaire d'au plus égale à 1. De plus, le temps de mélange d'une telle marche correspond à l'écart spectral de cette matrice. Par conséquent, ce temps de mélange est entièrement régi par la seconde plus grande valeur propre.

1.3 Méthode possible du choix de sigma (σ)

Dans cette section nous proposons une méthode ne figurant pas dans les travaux de [5] pour le choix du paramétre σ. Soit une partition $\mathcal{P} = \{c_1, \dots c_k\}$ de I en k-classes et des points $X_i = \left(X_i^1, \dots, X_i^p\right)$, on a :

- Inertie totale est
$$I_{tot} = \frac{1}{n} \sum_{i=1}^{n} ||X_i - g_I||^2$$
- Inertie de la classe C_k est
$$I_{tot}(C_k) = \frac{1}{n_k} \sum_{i=1, x_i \in C_k}^{n_k} ||X_i - g_{C_k}||^2$$
- Inertie intra-classe est
$$I_{intra} = \frac{1}{n} \sum_{k=1}^{k} n_K I_{tot}(C_k)$$
.
- Ainsi l'inertie inter classe s'écrit
$$I_{inter} = I_{tot} - I_{intra}$$

Pour choisir le σ optimal pour notre algorithme, nous procédons de la manière suivante.

Nous proposons plusieurs σ choisi arbitrairement. Pour chacun de ces σ, nous exécutons l'algorithme en calculant les *inerties intra-classes*, les *inerties inter-classes* et les *inerties totales* sur les données de Y. Une fois qu'on a les différentes inerties, on fait le ratio pour chaque cas entre *inerties intra-classes* et *inerties totales*. Enfin, parmi ces ratio, on choisit le ratio minimal et c'est ratio minimal est notre σ recherché.

2 Résultats

Dans le cadre de la simulation de notre algorithme de *spectral clustering*, nous utiliserons deux types de jeu de données que sont : les données MNIST et les données spirales (MLBENCH). MNIST est un acronyme défini comme Mixed National Institute of Standards and Technology. C'est une base de données de chiffres écrits à la main. Cette base de données est constituée de 60000 images d'apprentissage et 10000 images de test pour un total de 784 variables explicatives. MLBENCH est un terme anglais signifiant Machine Learning Benchmark qui veut dire Référence d'apprentissage automatique. Quant à cette base de données, elle est correspond à la sortie de la fonction *mlbench.spirals* du package *mlbench* ayant la paramétrisation suivante : elle est constituée de $n = 300$ observations que sont le nombre des motifs à créer avec $cycles = 1.5$, le nombre de cycles que chaque spirale fait. Si l'écart-type sd est strictement positif, alors le bruit gaussien est ajouté à chaque point de données. Ainsi la sortie correspond à une dispersion de points formant deux spirales intriquées.

Nous testons ici la capacité des algorithmes à correctement classer les données, à savoir les spirales issus de la fonction mlbench ainsi que les données MNIST. Pour cette simulation nous utiliserons la distance euclidienne pour calculer la distance entre les objets dans un groupe, le paramétre σ optimal est quant à lui choisi automatiquement avec la méthode proposé ci haut. Pour des raisons d'exécution rapide de notre algorithme, nous prendrons les 1000 premières images seulement dans la base de données MNIST.

2.1 Figures

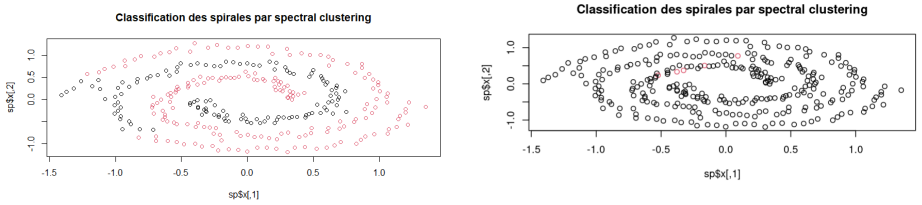
Le test des algorithmes spectral clustering, $k-means$ et EM sur nos jeux de données, nous forunissent notamment le partitionnement :
- des points de spirales en 2, pour les données du même nom
- des chiffres en 10 pour les 1000 premières images du jeux de données MNIST.

	cls_mnist									
labels_train	0	2	3	4	5	6	7	8	9	10
0	0	3	0	0	0	5	0	0	0	89
1	0	0	0	7	0	0	2	0	0	107
2	0	0	0	0	0	0	0	1	0	98
3	0	0	0	0	0	0	0	8	0	85
4	0	0	0	0	0	0	0	0	0	105
5	0	0	0	0	0	0	0	7	0	85
6	0	0	0	0	0	0	0	0	0	94
7	4	0	1	0	21	0	0	0	1	90
8	0	0	0	0	0	0	0	2	0	85
9	0	0	0	0	0	0	0	0	0	100

FIGURE 1 – Table de contingence du spectral clustering pour les données MNIST

- **Classification des données MNIST par spectral clustering**
  On rappel que notre fonction $spec-clus$ met plus de 15 minnutes à générer une classification à partir de 3000 images. Ainsi par soucis de temps, nous avons fait le choix d'une restriction aux 1000 premières données MNIST.

Ainsi le test sur ces données nous a permis de générer les tables de contingences pour chaque algorithme.
Notons qu'ici, il n'y a pas de bijection entre l'ensemble des chiffres arabes et nos 10 catégories des chiffres.
Sur la table de contingence du partitionnement généré par l'algorithme du spectral clustering, on observe que la majorité des images ont été classées dans la 10^{eme} classe ce qui n'est évidemment pas un bon résultat.
Néanmoins au sein d'une classe, nous retrouvons des images dont la fréquence est encouragente :
- la classe 5 contient 21 images du chiffre 7 ;
- les classes 4 et 8 contiennent 7 images respectivement du chiffre 1 et du chiffre 5 ;
- la classe 8 contient aussi 8 images du chiffre 3.
Ainsi malgré des nombresues images mal classées en la 10^{eme} catégorie, d'autres correspondantes au même chiffre manuscrit, sont regroupées dans la même classe.
On s'aperçoit que parmi les images classées en catégorie du chiffre 8, les plus fréquentes avec un score de 8/18 et 7/18 correspondent respectivement à celle du chiffre 3 et 5.
On s'aperçoit que parmi les images classées en catégorie du chiffre 9, la plus fréquente avec un score de 30/35 correspond au chiffre 0 tandis que les 3 autres représentent le 3, le 5 et le 6 qui sont des chiffres dont la manuscrit ressemble plus à un 0 qu'à 1 par exemple.
Ainsi sur au moins un exemple, l'algorithme EM semble meilleur que l'algorithme du spectral clustering pour reconnaitre et classer les images correspondantes à un même chiffre.



- **Classification des données spirales (MLBENCH) par spectral clustering**

Dans la partie concernant la méthode pour choisir la valeur optimale de σ, nous avions trouvé, pour les données spirales, une valeur de $\sigma = 0.7$. Cépendant nous avons constaté que pour ce type de données, cette valeur ne permet pas, à la fois d'optimiser les inerties intra-et inter-classes et la classification dont est capable le spectral clustering voir la figure ci-dessus (à droite). Nous l'obtenons néanmoins avec une valeur $sigma - opt = 0.05$, trouvée par test successif de différentes incluses dans le voisinage du centre des spirales. Avec une telle valeur de σ, nous obtenons une classification visualisable sur le graphe voir la figure ci-dessus(à gauche). Dans le cadre de cette classification, ici nous observons que les points correspondants aux têtes initiales des deux spirales sont regroupés en deux classes distinctes. Néanmoins, à partir de la position de coordonnées $(-0.25, -0.75)$, les points de chaque partition commencent à intervertir leur groupe d'appartenance à l'une des spirales.

Ainsi la méthode de partitionnement du spectral clustering semble assez performante pour ces données et cette valeur de $\sigma = 0.05$.

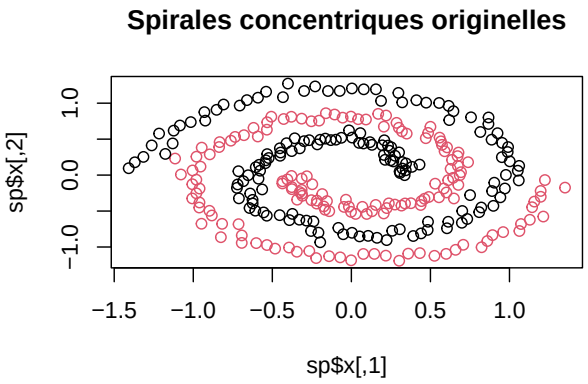


FIGURE 2 – Graphe des données initiales sans classificateur

2.2 Comparaison avec EM-algorithm et k-means

labels_train	cl_EM_mnist 1	2	3	4	5	6	7	8	9	10
0	0	0	0	54	4	0	2	5	31	0
1	0	0	0	0	0	0	2	0	0	1
2	0	1	10	2	2	2	5	53	0	1
3	0	0	72	0	0	0	12	1	1	5
4	0	0	0	0	3	31	27	0	0	36
5	0	0	30	13	2	3	8	1	1	0
6	0	0	0	0	78	3	2	1	2	0
7	0	0	0	0	0	6	39	1	0	58
8	0	0	24	0	2	0	12	1	0	15
9	0	0	2	1	1	11	43	0	0	33

FIGURE 3 – Table de contingence de EM pour les données MNIST

- **Classification des données MNIST par EM-algorithm** Sur la table de contingence du partitionnement généré par l'algorithme EM, on observe une classification des images plus éparse que celle faite par spectral clustering. En effet, une grille de lecture verticale indique que pour chaque classe, les images d'un chiffre donnésont d'un effectif non négligeable. Par exemple, si l'on regarde la composition de la classe 3, on observe qu'elle compte 72 images du chiffre 3, 30 du chiffre 5, et 24 du chiffre 8 et 10 du chiffre 2.
  Ainsi contrairement au spectral clustering, cette table semble indiquer qu'au sein de chaque classe se dégage deux voir trois groupes majoritaires correspondant à deux ou trois chiffres.

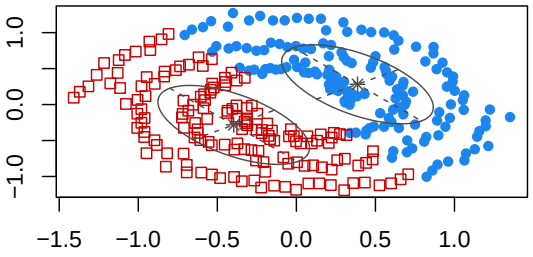


FIGURE 4 – Graphe avec EM-algorithm classifier

- **Classification des données spirales par EM-algorithm** La méthode EM, semble ne pas réussir à classer correctement les points. En effet, l'EM ne montre plus de spirales mais une dichotomie dont la frontière semble aussi être une droite. Il est possible que ces données formant des spirales soient encore trop difficile pour l'algorithme EM. Néanmoins, nous pourrions le tester sur d'autres type de données. Par exemple, des données avec une grande variance seront partionnées par l'EM de façon à ce que les frontières de chaque cluster aient une forme ellipsoïdale voir parabollique .

	cl_km_mnist									
labels_train	0	2	3	4	5	6	7	8	9	10
0	1	0	0	1	6	1	4	7	74	3
1	0	104	0	1	0	0	0	11	0	0
2	0	20	5	1	55	4	5	3	0	6
3	3	1	1	11	1	56	0	19	1	0
4	0	2	5	0	0	0	2	21	0	75
5	1	10	0	8	1	31	2	16	2	21
6	0	6	0	0	0	0	75	8	1	4
7	0	12	77	0	1	0	0	24	0	3
8	23	8	0	33	0	3	1	17	0	2
9	0	4	43	1	0	1	1	41	1	8

FIGURE 5 – Table de contingence de $k-means$ pour les données MNIST

- **Classification des données MNIST par k-means** Tout comme observé sur la table de contingence issus de l'algorithme EM, nous observons, au sein de chaque classe, des groupes d'images (associé à chiffre), majoritaires. Seul les classes 7 et 9 semblent contenir et réfleter le regroupement des chiffre 6 et 0 avec un effectif respectif de 75 et 74 images.
On observe que parmi les images classées en catégorie 1, la plus fréquente avec un score de 23/28 correspond au chiffre 8 tandis que les 5 autres représentent le 3 d'effectif 3, le 0 d'effectif 1 et le 5 d'effectif 1. On comprend que le chiffre 5 soit classé avec le 8 dont les écritures manuscrites sont relativement semblables.

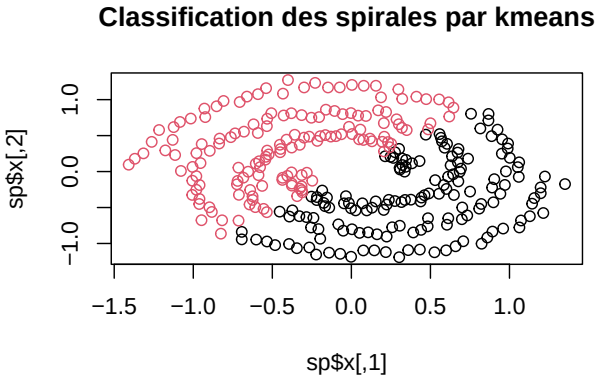


FIGURE 6 – Graphe avec $k-means$ classifier

- **Classification des données spirales par k-means**
  Avec la méthode des $k-means$, le partionnement se fait certes en deux classes, mais celui-ci ne respecte pas la répartition initiale des groupes. En effet, nous observons un regroupement en deux parties qui dans plan peuvent être séparées par une droite.
  Ainsi, comme attendu, $k-means$ échoue à l'exercice du partitionnement. En effet, contrairement au spectral clustering, qui transforme les données en données formant des régions séparables par des régions convexe, $k-means$ ne les modifient pas.

3 Discussion

La classification des données a toujours été une question fondamentale dans plusieurs domaines ou secteurs d'activités de notre quotidien. Avec l'avènement des nouveaux domaines tels que le machine learning et le deep learning, ce secteur a pris son envol et est devenu incontournable non seulement dans le domaine de l'analyse exploratoire des données mais aussi et surtout dans la prise des décisions. Les algorithmes de classification tels que $k-means$ et $EM-algorithm$ classent bien une certaine catégorie des données (par exemple les données MNIST étudiée dans ce projet). Mais toutefois pour des données non clairement séparables par des régions convexes, comme les données spirales, (comme nous l'ont montré les résultats de nos simulations) ces algorithmes ne parviennent pas à bien classer. Contrairement aux précédents algorithmes, l'algorithme de spectral clustering se trompe moins souvent à l'exercice du partitionnement de nos différentes données bien qu'il nécessite souvent plus temps (cas des données MNIST avec plus de 3000 observations). Il convient de dire que l'algorithme de spectral clustering a de beaux jours devant lui si une meilleure façon de choisir le paramètre σ est trouvé afin d'éviter des choix arbitraires à priori.

Remerciements

Nous tenons à remercier chaleureusement notre tuteur de projet Prof. **Joan Alexis Glaunès** pour avoir accepter de nous encadrer. Nous lui exprimons toute notre gratitude pour son encadrement, dévouement et son regard avisé quant à notre travail. Nous remercions également les enseignants du laboratoire MAP5 de l'Université Paris Cité en particulier ceux et celles du Master IMB.

Références

[1] Hongjie Jia, Shifei Ding, Xinzheng Xu, and Ru Nie. The latest research progress on spectral clustering. *Neural Computing and Applications*, 24(7) :1477–1486, 2014.

[2] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9 :381–386, 2020.

[3] Marina Meila and Jianbo Shi. Learning segmentation by random walks. *Advances in neural information processing systems*, 13, 2000.

[4] Tom M Mitchell and Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[5] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering : Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

[6] Mahamed GH Omran, Andries P Engelbrecht, and Ayed Salman. An overview of clustering methods. *Intelligent Data Analysis*, 11(6) :583–605, 2007.

[7] Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. Machine learning definition and basics. In *An Introduction to Machine Learning*, pages 1–17. Springer, 2019.

[8] Yair Weiss. Segmentation using eigenvectors : a unifying view. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 975–982. IEEE, 1999.