Subspace Clustering et typicalité

Rapport de stage

Celina HANOUTI

Maitre de stage :
Marie-Jeanne Lesot
@ LFI, LIP6

27 Mai 2019 - 18 Juillet 2019

Table des matières

1 Introduction

Ce rapport présente les travaux réalisés dans le cadre d'un stage d'une durée de 8 semaines effectué au sein de l'équipe LFI du LIP6 sur une thématique d'apprentissage automatique non supervisé. Le Clustering est une méthode de Machine Learning qui consiste à regrouper des données en des groupes spécifiques de telle sorte à ce qu'ils soient similaires entre eux et dissimilaires aux données affectées à d'autres groupes. Autrement dit, les points de données appartenant au même groupe devraient avoir des propriétés et des caractéristiques similaires, tandis que les points de données de différents groupes devraient avoir des propriétés et des caractéristiques très différentes.

Ce stage a pour but d'effectuer d'abord une étude comparative des différentes méthodes de Clustering et plus particulièrement celles dérivées de fonctions de coût, nous explorerons ensuite l'idée de concevoir un algorithme de Subspace Clustering basé sur la notion de degré de typicalité.

2 Etat de l'art

Dans cette partie, on présente l'état de l'art en Subspace Clustering et en typicalité : la section 2.2 présente la tâche de Clustering, la section 2.3, la tâche de Subspace Clustering et enfin, la section 2.3 introduit les notions de prototypes et de typicalité notamment l'algorithme TBC [3].

2.1 Notations

Dans tout le rapport, nous utilisons les conventions d'écriture suivantes : On considère une matrice de données $X = (x_{ip}) \in \mathrm{R}^{nd}$ représentant n points (X_i) d'un espace à d dimensions. Les clusters sont notés (C_r) avec $r \in [\![1; c]\!]$, où c désigne le nombre de clusters cherchés.

2.2 Le clustering

Le clustering est une tâche qui vise à identifier des groupes denses au sein des données appelés clusters, qui présentent une forte similarité interne et une forte dissimilarité externe. Nous présentons ici en détail les deux algorithmes de clustering itératif les plus connus, à savoir, l'algorithme des k-moyennes et sa variante, l'algorithme des c-moyennes floues qui s'appuie notamment sur le paradigme flou. Il est à noter que pour obtenir un bon partitionnement des données il faut à la fois faire en sorte d'obtenir des groupes les plus homogènes et les plus différents possible.

2.2.1 Algorithme des k-means

L'algorithme des k-means proposé par (MacQueen et al. 1967 ; Lloyd 1982) est l'un des plus anciens algorithmes de clustering. Il est à la base d'une riche littérature théorique ainsi que de nombreux algorithmes. Basé sur une fonction de coût et une stratégie d'optimisation simple, il est sans doute l'algorithme de clustering le plus utilisé en fouille de données. Il a été conçu pour regrouper des données numériques dans lesquelles chaque groupe a un centre appelé *centroid*. l'algorithme des k-means fixe le nombre de Clusters en le prenant comme argument noté k et il est de plus, basé sur une fonction de coût. Pour un nombre initial k de clusters, il affecte chaque point au cluster dont le centroid est le plus proche, il met à jour les affectations jusqu'à ce que la valeur donnée par la fonction de coût ne change plus significativement. On considère la matrice de coefficient $U \in \{0,1\}^{kn}$: $u_{ri} = 1$ si et seulement X_i est affecté au cluster C_r qui designe également son centre.
La fonction de cout à minimiser est :

$$F(C,U) = \sum_{i=1}^{n} u_{ri} \|X_i - C_r\|^2$$

sous les contraintes suivantes :

$$\forall i \in [\![1,n]\!], \sum_{r=1}^{k} u_{ri} = 1 \tag{1}$$

$$\forall r \in [\![1,k]\!], \sum_{r=1}^{k} u_{ri} > 0 \tag{2}$$

$$\forall r, \forall i, u_{ri} \in \{0,1\} \tag{3}$$

La fonction de coût peut être interprétée comme étant l'inertie globale, c'est à dire la somme des inerties intra-cluster. Comme précisé précédemment, le nombre de cluster est un paramètre de l'algorithme, ce qui peut être parfois un inconvénient. Il existe cependant plusieurs mesures de qualité d'une partition d'un ensemble de données, on peut citer notamment l'index de Dunn [5] défini comme étant le rapport entre la distance maximum qui sépare deux points affectés au même cluster et la distance minimum qui sépare deux points classés séparément. Les étapes de l'algorithme des k-means sont explicitées ci-dessous [1].

Algorithm 1 k-means Clustering

1. Initialiser aléatoirement k centres, c_r , $r \in [\![1, k]\!]$
2. Itérer jusqu'à convergence de la fonction de cout
   - Affecter chaque point au cluster dont le centroid est le plus proche, en definissant la matrice d'affectation
     $u_{ri} = \begin{cases} 1 si \text{ r} = argmin \ \|\text{X}_i \text{ - } c_r\|^2 \\ 0 \qquad\qquad\qquad\qquad \text{sinon.} \end{cases}$
   - Mettre à jour les centroids : $r \in [\![1, k]\!]$
     $c_r = \frac{\sum_{i=1}^{n} X_i u_{ri}}{\sum_{i=1}^{n} u_{ri}}$

2.2.2 Algorithme des fuzzy c-means

La méthode des c-moyennes floues fait partie des méthodes dites de "*soft* clustering". Il est parfois possible que les Clusters soient mal délimités ou qu'ils partagent des points, cette méthode propose donc de ne plus prendre en compte la contrainte (3) définie dans l'algorithme des k-means qui impose que chaque point soit affecté à un unique cluster et de la remplacer par un degré d'appartenance $u_{ri} \in [0, 1]$ qui doit tout de même sommer à 1 pour forcer les solutions à représenter équitablement tous les points (X_i).
La fonction de coût à minimiser devient donc :

$$F(C, U) = \sum_{i=1}^{n} \sum_{r=1}^{c} u_{ri}^m \|X_i - C_r\|^2$$

sous les contraintes suivantes :

$$(C1) \ \forall i \in [\![1, n]\!], \sum_{r=1}^{k} u_{ri} = 1$$

$$(C2) \ \forall r \in [\![1, k]\!], \sum_{r=1}^{k} u_{ri} > 0$$

Comme la fonction F est differentiable en C et en U, on peut donc dériver les équations de mise à jour en respectant les contraintes et cela en utilisant la méthode des multiplicateurs de Lagrange. Par simplification, posons $d_{ir} = \|\text{X}_i \text{ - } C_r\|$ Soit la fonction de Lagrange L et $\forall i \in \{1, .., c\}$ λ_i des multiplicateurs de Lagrange.

$$L(X, U, X, \lambda) = \sum_{i=1}^{n} \sum_{r=1}^{c} u_{ri}^m d_{ir}^2 + \sum_{i=1}^{n} \lambda_i (1 - \sum_{r=1}^{c} u_{ri})$$

En annulant les derivés partiels de la fonction de Lagrange, on obtient donc pour chaque $k \in \{1, .., c\}$ et pour chaque $l \in \{1, .., n\}$:

$$\frac{\partial L(X, U, C, \lambda)}{\partial u_{kl}} = m u_{kl}^{m-1} d_{kl}^2 - \lambda_l = 0$$

$$u_{kl} = (\frac{\lambda_r}{m d_{kl}^2})^{\frac{1}{m-1}} \quad (1)$$

$$\frac{\partial L}{\partial \lambda} = 0$$

et donc

$$\sum_{r=1}^{c} u_{ri} = \sum_{r=1}^{c} \left(\frac{\lambda_i}{m d_{ir}^2} \right)^{\frac{1}{m-1}}$$

$$\sum_{r=1}^{c} \left(m d_{ir}^2\right)^{\frac{1}{1-m}} = \lambda_i^{\frac{1}{1-m}}$$

$$\lambda_i = \left(\sum_{r=1}^{c} (m d_{ir}^2)^{\frac{1}{1-m}}\right)^{1-m}$$

En remplacant dans l'equation (1) on obtient :

$$u_{ri} = \frac{d_{ir}^{\frac{2}{1-m}}}{\sum_{s=1}^{c} (d_{is})^{\frac{2}{1-m}}} = \frac{1}{\sum_i \left(\frac{d_i r}{d_i s}\right)^{\frac{2}{m-1}}}$$

En procédant de la même manière on obtient :

$$C_r = \frac{\sum_{i=1}^{n} u_{ri}^m X_i}{\sum_{i=1}^{n} u_{ri}^m}$$

2.2.3 Comparaison et expérimentations

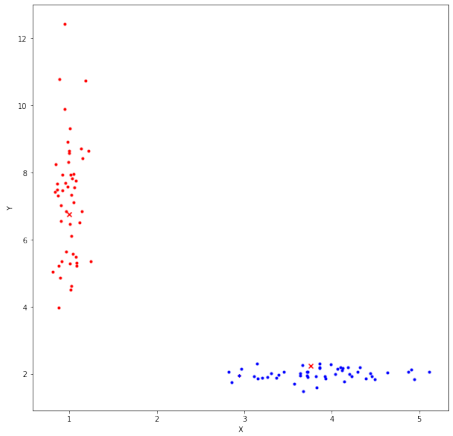
Afin de comparer les deux méthodes présentées précédemment, j'ai effectué différents tests sur des jeux de données artificiels, en testant notamment la robustesse et la sensibilité à l'initialisation aléatoire en comparant à chaque fois les partitions données par chacune des méthodes. Remarquons d'abord que la méthode de k-means affecte une donnée qu'à un seul et unique cluster contrairement à la méthode des fuzzy c-means qui affecte une donnée à plusieurs clusters avec différents degrés d'appartenance. Que ce soit pour des données avec des groupes bien séparés ou pour des données où les clusters ne sont pas bien définis visuellement, les deux algorithmes donnent toujours la même partition, autrement dit, une donnée est toujours affectée au même cluster dans les deux partitions résultantes des deux méthodes. Ces deux algorithmes ont cependant des inconvénients en commun : sensibilité aux outliers, complexité mémoire et le fait qu'on ne puisse savoir le nombre d'itérations requis pour obtenir les clusters. La sensibilité aux outliers est due au calcul de la moyenne pour obtenir le centroid du cluster, en effet, si on suppose que les données contiennent 2 clusters bien définis et un outlier situé assez loin de ces deux derniers, la valeur de la fonction de coût serait élevée et comme l'algorithme fera en sorte de minimiser cette dernière, l'un des deux centroides sera attiré par l'outlier. Les expérimentations détaillées sont présentées dans un notebook Python omis dans ce rapport, nous indiquons ici uniquement les principaux résultats obtenus.

2.3 Subspace Clustering

Le Subspace Clustering est une extension du clustering et une généralisation de réduction de dimensionnalité qui, en plus des clusters, identifie également les sous-espaces dans lesquels ils existent. Souvent, lorsqu'on travaille sur des données avec beaucoup de dimensions, certaines de ces dimensions ne sont pas pertinentes, l'identification des sous-espaces consiste donc à éliminer ces dimensions non-pertinentes. De plus, cela dépend de l'identification des custers et réciproquement. Il existe de nombreux paradigmes de Subspace Clustering, ici on s'intéresse aux algorithmes qui s'appuient sur une fonction de coût, selon les données X, son minimum ou ses minima correspondent aux partitions optimales de Subspace Clustering.

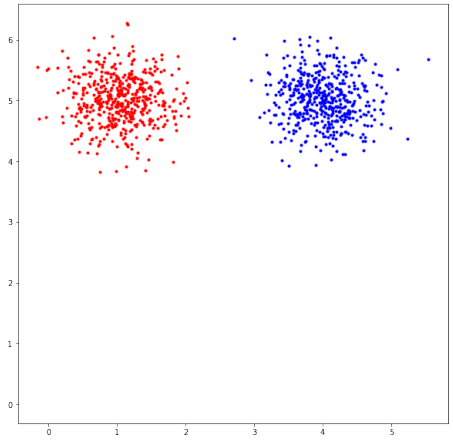
2.3.1 Fuzzy weighted c-means

Comme pour les algorithmes présentés précédemment, les clusters sont décrits par un centre $C_r \in \mathbb{R}^d$, chaque point est affecté à un cluster au moyen d'une matrice de degré d'appartenance $U = u_{ri}$ à valeurs continues pour les algorithmes flous et à valeurs discrètes pour les autres. L'algorithme $AWFCM$ introduit en plus une matrice $W = w_{rp} \in [0, 1]$ qui définit l'importance de l'attribut p par rapport au cluster C_r et cela avec une distance pondérée : plus w_{rp} est faible, moins l'attribut p est significatif pour le cluster C_r.



$$W = \begin{pmatrix} 0.99551384 & 0.00448616 \\ 0.86800793 & 0.131992069 \end{pmatrix}$$

FIGURE 1 – Exemples d'execution d'$AWFCM$ sur la deuxième partition



$$W = \begin{pmatrix} 0.54042859 & 0.45957141 \\ 0.40386222 & 0.59613778 \end{pmatrix}$$

FIGURE 2 – Exemples d'execution d'$AWFCM$ sur la deuxième partition

2.4 Prototypes et degré de typicalité

Les prototypes sont des éléments issus des sciences cognitives, traduites ensuite informatiquement [3], qui représentent des catégories ou des sous-ensembles en mettant en évidence à la fois leurs caractéristiques communes les plus importantes et leur spécificité par rapport aux autres catégories ou aux autres sous-ensembles. Les prototypes sont souvent liés à la notion de typicité aussi appelée typicalité, qui modélise à quel point une donnée est représentative du sous-ensemble auquel elle appartient. Le degré de typicité dépend de deux autres notions : la ressemblance interne et la dissimilarité externe. La ressemblance interne mesure la ressemblance d'une donnée par rapport aux autres données qui appartiennent au même sous-ensemble, la dissimilarité externe, quant à elle, mesure à quel point une donnée est différente des données appartenant aux autres sous-ensembles. Ici on considère la distance euclidienne normalisée comme mesure de dissimilarité externe et le complémentaire de cette mesure pour la ressemblance interne. Formellement, on note d et r la mesure de dissimilarité et la mesure de ressemblance respectivement et on note $D(x, C)$ la dissimilarité externe du point $x \in C$ tel que :

$$D(x, C) = avg(d(x, y), y \notin C)$$

et $R(x, C)$ la ressemblance interne du point x :

$$R(x, C) = avg(r(x, y), y \in C)$$

Le degré de typicalité T est obtenu en agrégeant la ressemblance interne et la dissimilarité externe.

$$T(x, C) = \varphi(R(x, C), D(x, C))$$

avec φ l'opérateur d'agrégation qui peut être défini de plusieurs manières, on considérera principalement les opérateurs d'agrégation suivants : $T = \min(R, D)$, $T = \max(R, D)$, $T = 0.6R + 0.4D$ et $T = \text{MICA}(R, D)$. [2]

2.4.1 Typicality based clustering algorithm

Comme il a été mentionné précédemment, la tache de Clustering a pour but de décomposer des données en sous-groupes qui sont à la fois homogènes et distincts, cette homogénéité est entre autre obtenue en faisant en sorte que les données affectées au même Cluster se ressemblent. Le fait que les Clusters soient distincts implique que les points affectés à des Clusters différents soient dissimilaires entre eux. Ces propriétés peuvent être associées aux notions de ressemblance interne et dissimilarité externe présentés dans la partie précédente. Ces motivations sont présentées plus en détail dans le papier [3]. l'algorithme TBC consiste en deux étapes :

- En partant d'une partition, calculer le degré de typicalité de chaque donnée en prenant en compte le cluster auquel chacune d'entre elle est affecté.
- Modifier la partition de sorte à ce que chaque point soit affecté au cluster dont il est le plus typique.

3 Un algorithme de Subspace Clustering basé sur la notion de typicalité

Dans le chapitre précédent, nous avons présenté la tache de Subspace Clustering et la notion de degré de typicalité en décrivant notamment les algorithmes $AWFCM$ qui résout le problème de Subspace Clustering et l'algorithme TBC qui résout la tache de Clustering en utilisant le degré de typicalité. Durant ce stage, nous nous sommes intéressés à la possibilité d'utiliser la notion de typicalité afin de résoudre le problème des poids dans le Subspace Clustering qui décrit la pertinence des attributs pour chaque cluster, autrement dit, on cherche à savoir pour quels attributs, chaque cluster est le plus typique. On introduira donc une nouvelle définition de la typicalité, cette dernière sera calculé attribut par attribut.
Plusieurs études expérimentales sont nécessaires afin de prouver la pertinence de cette approche et savoir notamment quels informations sur les données nous apporte le degré de typicalité. Les expérimentations ont été effectué sur des données artificielles générées à partir de distributions Gaussiennes. Les figures 1 et 2 nous montrent les résultats obtenus sur les deux partitions, il est à noter que le Cluster $C1$ correspond au Cluster en rouge et le Cluster $C2$ correspond au Cluster en bleu. les poids résultants d'$AWFCM$ nous montrent, que pour la partition 1, le Cluster $C1$ est plus spécifique sur l'attribut X que l'attribut Y, contrairement au Cluster $C2$, pour qui l'attribut Y est plus significatif que l'attribut X. Dans le but de trouver une sémantique identique à celle d'$AWFCM$, nous avons tracé la ressemblance interne, la dissimilarité externe et le degré de typicalité attribut par attribut, les figures 4q et 5 montrent les courbes obtenues et sont commentées en détail dans un notebook jupyter omis dans ce rapport. Les valeurs maximales et les valeurs moyennes sont assez proches pour les deux attributs, autrement dit, agréger les résultats de ces deux manières-là nous apporte aucune information sur les espaces dans lesquels les Clusters sont définis. Une autre façon d'agréger les résultats serait de calculer l'aire sous la courbe, les résultats obtenus sont présentés dans les tableaux 1 et 2 ci-dessous. On remarque également qu'aucun de ces opérateurs d'agrégation nous permettent de conclure quand à la pertinence de cette approche. Ces résultats sont en effet peu satisfaisants ce qui nous laisse croire qu'il y'a bien une sémantique différente entre les poids résultants d'$AWFCM$ et les degrés de typicalité.

Clusters		Aire sous la courbe	Moyenne	Maximum
Cluster C1	Ressemblance interne	Attribut X :1.52 , Attribut Y :8.87	X :0.97 , Y :0.82	X :0.98 , Y :0.87
	Dissimilarité externe	X :0.74 ,Y : 3.07	X :0.53 , Y :0.12	X :0.58 , Y :0.52
	$60\% \cdot ress + 40\% \cdot diss$	X :1.21 ,Y :6.55	X :0.80 ,Y : 0.54	X :0.81 , Y :0.55
Cluster C2	Ressemblance interne	X :2.98 , Y :1.03	X :0.88 , Y :0.98	X :0.91 , Y :0.98
	Dissimilarité externe	X :2.27 , Y :0.13	X :0.53 , Y :0.12	X :0.94 ,Y :0.13
	$60\% \cdot ress + 40\% \cdot diss$	X :2.70 , Y :0.67	X :0.74 , Y :0.63	X :0.78 , Y :0.64

TABLE 1 – Résultats sur la Partition $P1$

Clusters		Aire sous la courbe	Moyenne	Maximum
Cluster C1	Ressemblance interne	Attribut X :0.53 , Attribut Y :2.41	X :0.97 , Y :0.33	X :0.98, Y :0.45
	Dissimilarité externe	X :2.21 , Y :9.81	X :0.57 , Y :0.54	X :0.52 , Y :1.0
	$60\% \cdot ress + 40\% \cdot diss$	X :1.64 , Y :1.33	X :0.75 ,Y :0.56	X :0.78, Y :0.57
Cluster C2	Ressemblance interne	X :3.39 , Y :0.93	X :0.88 , Y :0.90	X :0.91 , Y :0.93
	Dissimilarité externe	X :2.36 , Y :0.58	X :0.52 , Y :0.54	X :0.93 , Y :0.52
	$60\% \cdot ress + 40\% \cdot diss$	X :2.98 , Y :1.17	X :0.74 , Y :0.56	X :0.77 , Y :0.64

TABLE 2 – Résultats sur la Partition $P2$

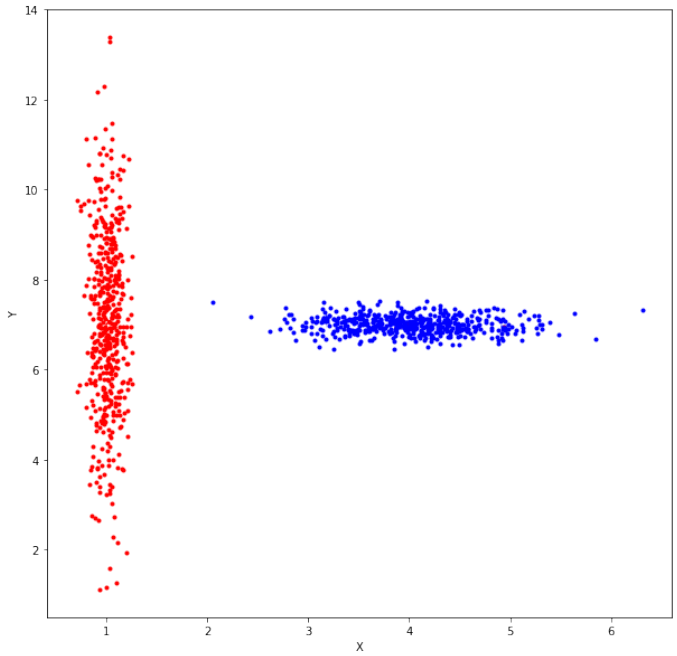
4 Conclusion du stage et perspectives

Mon stage a porté sur la tâche de Subspace Clustering et sur les notions de prototypes et de degré de typicalité. Après avoir fait l'état de l'art sur les méthodes de Clustering et de Subspace Clustering proposées dans la littérature en effectuant notamment une étude comparative et en me référant principalement à la thèse d'Arthur Guillon [1], je me suis placée dans le contexte des méthodes basées sur des fonctions de coût et je me suis intéressée sur la possibilité de résoudre le problème de Subspace Clustering en utilisant le degré de typicalité défini attribut par attribut.

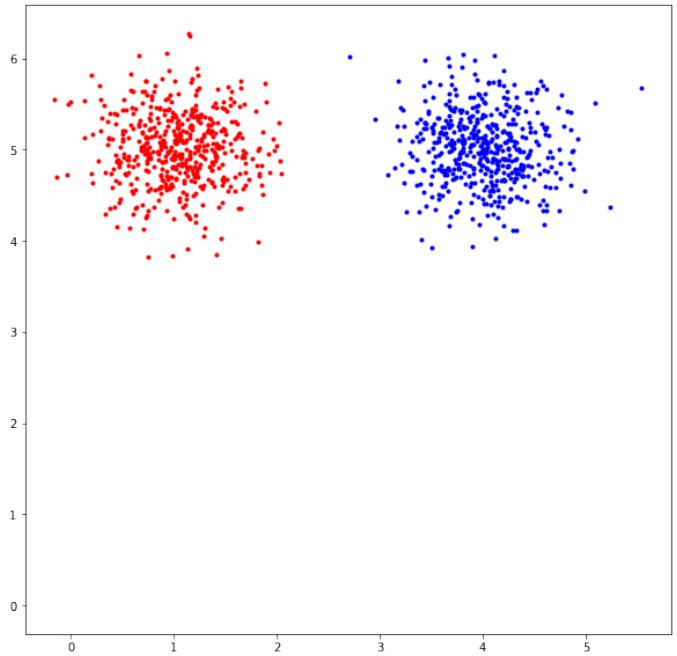
J'ai effectué une étude théorique de deux algorithmes, à savoir l'algorithme $AWFCM$ et l'algorithme TBC afin d'extraire certaines propriétés et me familiariser avec les taches de typicalité et de prototypes.

Des expérimentations sur des bases artificielles ont été menées afin de vérifier la pertinence de cette approche. Les résultats observés sur ces données ne permettent pas de conclure sur l'optimalité de cette méthode et de souligner l'apport du degré de typicalité pour la tâche de Subspace Clustering en général. Une perspective possible serait de travailler sur ce problème en creusant un peu plus sur les opérateurs d'agrégation utilisés.

Sur un plan plus personnel, ce stage m'a initiée à la recherche en laboratoire et m'a confortée quant à mon choix de poursuivre mes études dans ce domaine. Je tiens à remercier mon encadrante, Marie-Jeanne Lesot, ainsi qu'Adrien Revault d'Allonnes, pour m'avoir aidée et accompagnée durant ce stage.

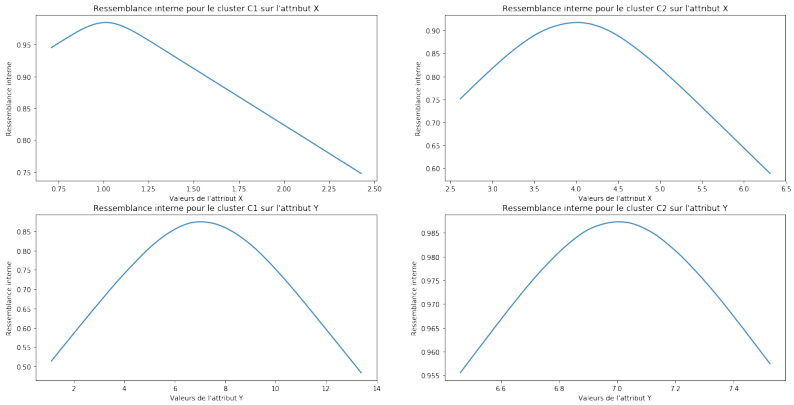


((a)) Partition $P1$

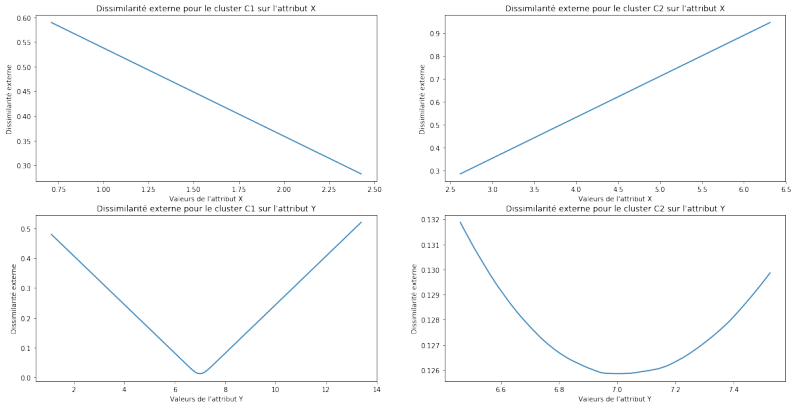


((b)) Partition $P2$

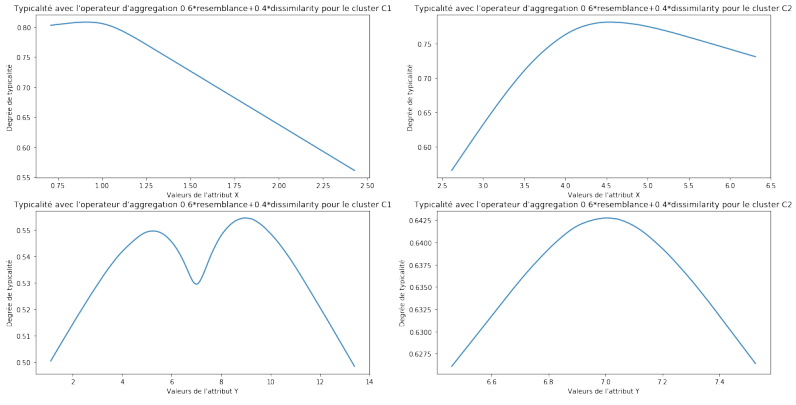
FIGURE 3 – Deux partitions de données artificielles générées à partir de deux distributions gaussiennes distinctes



((a)) Ressemblance interne calculée attribut par attribut

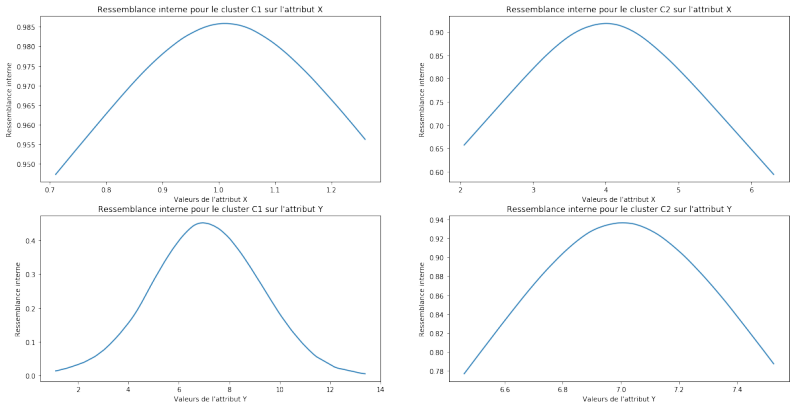


((b)) Dissimilarité externe calculée attribut par attribut

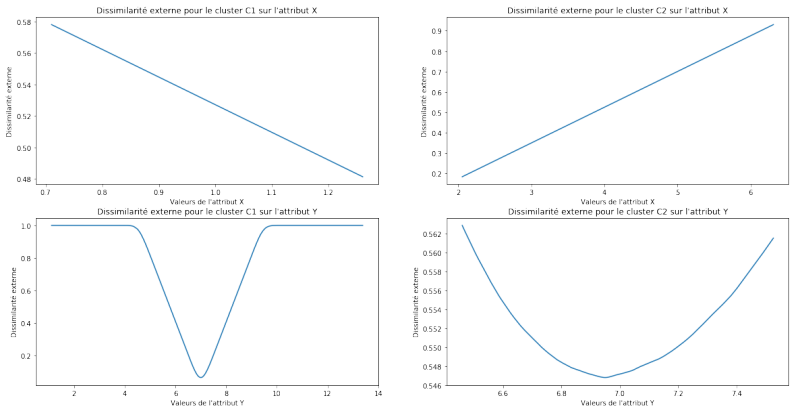


((c)) Degré de typicalité calculée attribut par attribut

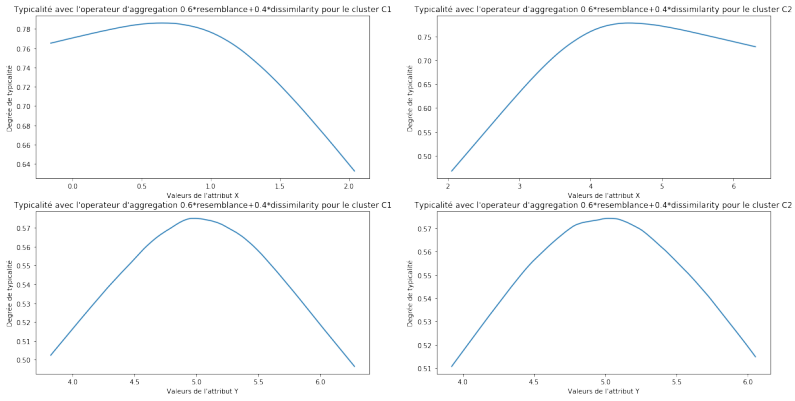
FIGURE 4 – Résultats obtenus pour la partition $P1$



((a)) Ressemblance interne calculée attribut par attribut



((b)) Dissimilarité externe calculée attribut par attribut



((c)) Degré de typicalité calculée attribut par attribut

FIGURE 5 – Résultats obtenus pour la partition $P2$

Références

[1] A. Guillon. "Opérateurs de régularisation pour le subspace clustering flou". Thèse de doct. Sorbonne Université, 2019.

[2] A. Kelman et R. Yager. "On the application of a class of mica operators". In : *Journ. of Uncertainty* 27.10 (1995), p. 113–126.

[3] M. Rifqi, M.-J. Lesot et B. Bouchon-Meunier. "Fuzzy prototypes : From a cognitive view to a machine learning principle". In : *Fuzzy Sets and Their Extensions : Representation, Aggregation and Models* 21 (2007), p. 431–452.