Thèse de doctorat de l'Université Sorbonne Université

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Arthur Guillon

Pour obtenir le grade de
Docteur de l'Université Sorbonne Université

Sujet de la thèse :

Opérateurs de régularisation pour le subspace clustering flou

soutenue —

devant le jury composé de :

M. Nicolas Labroche	Université de Tours	Rapporteur
M. Julien Velcin	Université Lyon 2	Rapporteur
M. Antoine Cornuéjols	AgroParisTech	Examinateur
M. Carl Frélicot	Université de la Rochelle	Examinateur
M. Patrick Gallinari	Sorbonne Université	Examinateur
Mme Marie-Jeanne Lesot	Sorbonne Université	Directrice de thèse
M. Christophe Marsala	Sorbonne Université	Directeur de thèse

RÉSUMÉ

Cette thèse considère une tâche d'apprentissage automatique appelée *subspace clustering* flou, une généralisation du clustering qui consiste à découvrir les attributs caractéristiques des clusters en même temps que les clusters eux-mêmes, en autorisant les points à être affectés partiellement à plusieurs clusters. Nous nous plaçons dans le cadre de l'optimisation mathématique, qui consiste à formuler une fonction de coût dont on dérive ensuite un algorithme qui minimise cette fonction pour produire un partitionnement des données en clusters et sous-espaces.

Nous formulons trois propriétés souhaitables en fouille de données, dont nous montrons qu'elles peuvent être traduites dans le vocabulaire de l'optimisation sous forme de fonctions de pénalité qui sont intégrées aux fonctions de coût. La première de ces propriétés est la parcimonie des descriptions des sous-espaces, qui consiste à limiter la longueur de ces descriptions pour les restreindre aux informations essentielles. La deuxième est la prise en compte de la représentativité des données au sein des clusters auxquels elles sont affectées, de façon à produire une description plus fidèle de ces clusters et à rendre les algorithmes plus résistants au bruit. La troisième propriété consiste à prendre en compte la géométrie des données dans le calcul de leur affectation aux clusters, permettant de préserver des informations locales qui contribuent à leur découverte.

Certaines de ces pénalités sont non différentiables, ce qui nous pousse à proposer l'utilisation du cadre théorique de l'optimisation proximale pour minimiser les fonctions de coût que nous proposons. Nous présentons un algorithme baptisé PSFCM qui généralise le schéma d'optimisation standard du clustering flou, qui combine à la fois optimisation alternée et descente de gradient proximal ; nous montrons ainsi que le cadre proximal peut être adapté aux spécificités du subspace clustering flou en dérivant plusieurs opérateurs proximaux adaptés aux différentes pénalités que nous introduisons.

Nous proposons ainsi plusieurs algorithmes de subspace clustering flou, parmi lesquels Prosecco, un algorithme parcimonieux ; Possecco, une variante possibiliste de Prosecco ; et WLFC, un algorithme de subspace clustering flou qui intègre une mesure de similarité basée sur la distance euclidienne non pondérée pour tenir compte de la géométrie des données. Ces algorithmes sont implémentés dans une bibliothèque Python et comparés aux principaux algorithmes de l'état de l'art sur des jeux de données qui permettent de mettre en avant leurs spécificités.

Mots-clefs : apprentissage non supervisé, subspace clustering, clustering flou, optimisation proximale, optimisation convexe, parcimonie, clustering possibiliste, régularisation laplacienne

TABLE DES MATIÈRES

PRINCIPAUX ALGORITHMES RENCONTRÉS DANS LA THÈSE

PARADIGME	ACRONYME	RÉFÉRENCE	PAGE
clustering	k-moyennes	Lloyd (1982)	p. 11
	FCM	Bezdek (1981)	p. 13
identification explicite	CLIQUE	Agrawal et al. (1998)	p. 16
	PROCLUS	Aggarwal et al. (1999)	p. 18
distances pondérées	AWFCM	Keller et Klawonn (2000)	p. 20
	GK	Gustafson et Kessel (1978)	p. 24
parcimonie	Borgelt	Borgelt (2010)	p. 27
	EWKM	Jing et al. (2007)	p. 27
	EWKM flou	—	p. 29
SSC	SSC	Elhamifar et Vidal (2009)	p. 32
contributions	PSFCM	Guillon et al. (2016b)	p. 46
	Prosecco		p. 56
	PFSCM	Guillon et al. (2016b)	p. 75
	WPPCM		p. 78
	Posseco		p. 79
	WLFC	Guillon et al. (2017)	p. 95

1

INTRODUCTION

Un grand nombre de disciplines scientifiques, économiques et industrielles reposent sur l'obtention, le traitement et l'extrapolation de données, dont sont extraites des informations ou des prédictions compréhensibles par les experts d'un domaine. Des exemples de données non traitées, dites « brutes », peuvent être les paramètres et résultats d'un essai clinique, les transmissions de capteurs placés le long d'une chaîne de production, les images de vidéo-surveillance d'une ville ou encore les avis des utilisateurs d'un site web marchand.

L'étude de ces données brutes et l'extraction d'informations de plus haut niveau ont justifié le développement de domaines entiers des mathématiques tels que les statistiques et précèdent de longtemps l'informatique. Cependant, la démocratisation des ordinateurs et les capacités modernes d'acquisition de données ont entraîné l'essor d'approches qui ne font pas nécessairement usage des statistiques, généralement regroupées sous le terme de « fouille de données » (*data mining* en anglais) ou plus récemment « science des données » (*data science*). Les approches modernes de fouille de données s'appuient souvent sur l'apprentissage automatique (*machine learning*), notamment non supervisé, qui consiste à extraire des données des informations majeures, prédominantes, appelées aussi « motifs » ou « tendances » (*patterns*). Ces motifs peuvent être des associations fréquentes au sein des données, des profils types de goûts d'utilisateurs et des recommandations associées ou encore au contraire des comportements atypiques, distincts des autres.

La tâche de clustering

La tâche considérée dans cette thèse et les motifs dont il est question sont illustrés en figure 1.1. Les données sont ici considérées comme des vecteurs en deux dimensions et représentées graphiquement par des points du plan. La figure de gauche représente ces données brutes, sans étiquette, faisant apparaître des groupes plus ou moins bien délimités ; la figure de droite montre ces groupes plus clairement, en attribuant des symboles et des couleurs différents aux points qui les constituent. De tels groupes sont appelés *clusters*.

De façon générale, la tâche de *clustering* a pour objectif d'attribuer de telles classes aux données de départ, produisant ainsi un résumé de ces données. Les algorithmes de clustering sont utilisés dans une phase exploratoire de la fouille de données et contribuent à faire apparaître une structure au sein des données.

Selon les données étudiées, il est possible que plusieurs regroupements soient pertinents ; la proposition d'un tel partitionnement peut donc reposer sur des choix arbitraires. Cette ambiguïté peut être exacerbée par les données elles-mêmes, comme l'illustre la figure 1.1 : la figure de droite représente par exemple deux points voisins, de coordonnées respectives (5.5, 2.4) et (5.2, 2.0), à mi-chemin entre les groupes rouge et bleu, qui, bien que très proches, sont affectés à des groupes différents. Ce choix a quelque chose d'arbitraire : d'autres possibilités auraient été de les affecter tous les

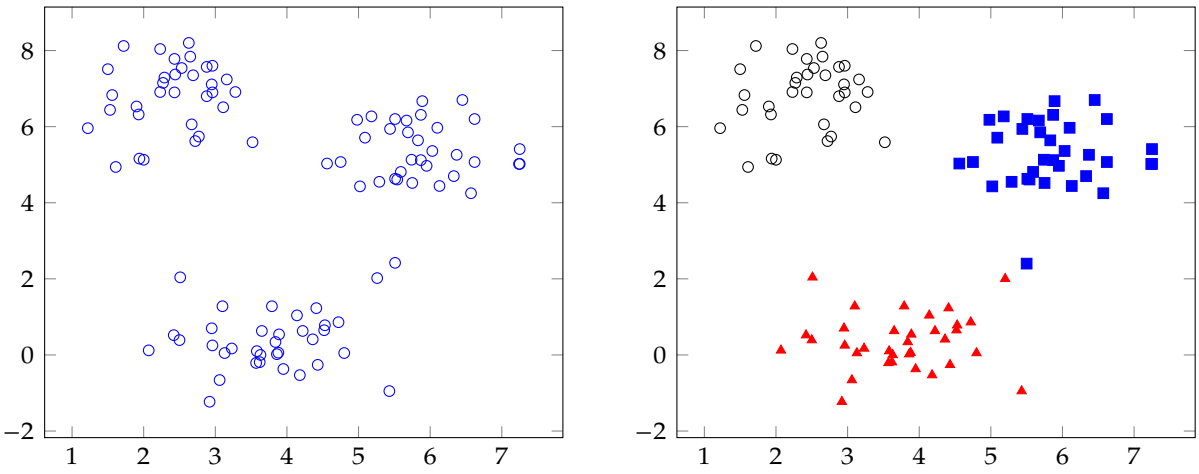


FIG. *1.1 :* Des données bidimensionnelles non étiquetées (à gauche) et un résultat de clustering possible (à droite)

deux à l'une des deux classes, de les affecter partiellement aux deux classes, de renoncer à les affecter, ou encore de les ranger dans une quatrième classe.

Il n'existe pas de définition formelle et générale de la tâche de clustering qui engloberait toutes les applications considérées en fouille de données. La définition exacte d'un cluster dépend de la nature des données et du but de la fouille. Cependant, les clusters correspondent toujours à une notion de groupes denses au sein des données, présentant deux propriétés essentielles :

- ils sont homogènes : les données regroupées au sein d'un cluster sont similaires les unes aux autres, selon un critère de similarité lié au paradigme et à l'algorithme de clustering retenus. Par exemple, l'algorithme identifie un objet auquel ces données ressemblent ou il s'appuie sur une relation de similarité qui les identifie les unes aux autres ;

- ils sont distincts : les données d'un cluster sont différentes des données d'un autre cluster. Il existe alors un moyen de les distinguer : l'algorithme les associe à des objets différents ou bien elles sont explicitement dissimilaires.

Par exemple, la figure 1.1 propose des clusters convaincants, les groupes formés contenant bien des données similaires. Premièrement, ces groupes sont construits autour de trois « centres » fictifs, des points qui ne font pas partie des données initiales mais qui pourraient être proposés par un algorithme pour décrire les classes formées. D'autre part, ces groupes sont denses, contenant des points qui sont tous proches les uns des autres. Ils sont également bien distincts, dans la mesure où ils sont clairement séparés et ne se chevauchent pas.

Tout algorithme de clustering dépend d'une notion de similarité, utilisée pour définir les clusters, qui doit être adaptée aux données et aux clusters recherchés. De nombreuses approches sont possibles, ce qui a pour conséquence la coexistence de nombreux paradigmes de clustering, dont les principaux sont présentés dans le chapitre 2. La similarité utilisée est *globale*, comparant tous les couples de points selon le même principe, sans tenir compte des spécificités locales des données. Pour certaines

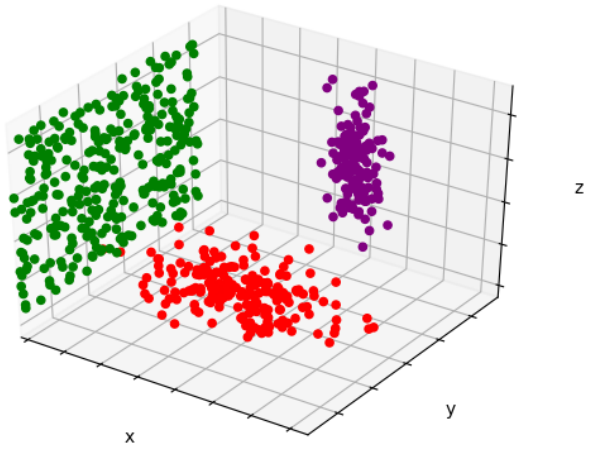


FIG. *1.2 :* Des données étiquetées en trois dimensions.

applications cependant, des données affectées à des clusters différents peuvent être similaires pour des raisons différentes.

Le subspace clustering

Le *subspace clustering* (Agrawal et al. 1998) est une généralisation de la tâche de clustering qui relâche la globalité de la mesure de similarité et consiste à chercher à la fois les clusters et des relations de similarité *locales* qui définissent ces clusters, fournissant des informations supplémentaires sur les groupes identifiés.

Les clusters et les similarités locales sont alors définis mutuellement et recherchés simultanément. De fait, de nouveaux groupes denses peuvent être découverts localement, permettant au *subspace clustering* de découvrir plus de groupes denses que le clustering — mais denses au sens de relations locales.

La figure 1.2 représente des groupes de points, cette fois-ci dans un espace en trois dimensions, dont certains se trouvent aplatis dans des sous-espaces de l'espace initial. Les amas sont moins clairement identifiés que dans la figure précédente, et un algorithme de clustering pourrait donc ne pas identifier certains d'entre eux. Ainsi, le groupe violet n'est dense que dans deux des trois dimensions originales, c'est-à-dire dans un sous-espace. De même, les points du groupe vert vivent dans un plan d'équation $x = k$, c'est-à-dire dans un sous-espace différent du précédent. C'est cette intuition géométrique qui est à l'origine du subspace clustering.

En outre, certains nuages de points de la figure s'entrecoupent. Comme pour la figure 1.1, les données présentent une ambiguïté qui peut être résolue de différentes façons. Dans cette thèse, nous adhérons au paradigme du subspace clustering flou, qui modélise de telles ambiguïtés en autorisant une affectation partielle des données, qui peuvent appartenir à plusieurs clusters simultanément. Le chapitre 2 donne une présentation formelle du subspace clustering flou.

Représentation des données : attributs et points

Cette thèse se place dans un cadre géométrique classique, les données étant considérées comme des points d'un espace euclidien de dimension potentiellement grande, dont les coordonnées sont les attributs des données. De plus, les clusters sont identifiés par des centres, qui sont des points de l'espace résumant les données affectées aux clusters. Enfin, toutes les données possèdent les mêmes attributs, dont la valeur exacte est connue avec certitude.

Ce cadre fournit la possibilité de se faire une intuition géométrique du fonctionnement des algorithmes de subspace clustering flou et des solutions qu'ils tendent à produire, au moins en basse dimension. Des exemples graphiques, produits à partir de données générées artificiellement par des mécanismes simples, sont utilisés tout au long de la thèse pour distinguer les caractéristiques majeures des algorithmes étudiés.

Malédiction de la dimensionnalité

Des données extraites d'un espace en haute dimension, en revanche, sont plus difficiles à étudier. Dans ces espaces, des sous-ensembles de données similaires peuvent différer d'après quelques attributs, qui peuvent correspondre à du bruit (par exemple des valeurs obtenues de capteurs défectueux) ou plus généralement à des informations qu'il est nécessaire d'ignorer pour découvrir davantage de clusters.

Ce phénomène classique, amplifié en haute dimension, est lié à la « malédiction de la dimensionnalité » (Bellman 1961). Celle-ci désigne le principe selon lequel, dans des espaces de grande dimension, la distance entre deux points choisis aléatoirement tend à s'uniformiser. Il devient alors impossible de partitionner les données en clusters convaincants. Des objets statistiques classiques telles que les distributions gaussiennes exhibent des comportements contre-intuitifs (Domingos 2012 ; Blum et al. 2018). D'autres outils classiques de fouille des données tels que la réduction de la dimensionnalité sont souvent utilisés, par exemple l'analyse en composantes principales, mais ceux-ci peuvent détruire à leur tour la structure des clusters (Chang 1983 ; Arabie et Hubert 1996 ; Agrawal et al. 1998).

À la recherche de solutions simples

Cependant, en haute dimension, les données exhibent également parfois des structures simples (Murtagh 2009 ; Gorban et al. 2016), par exemple en rendant les données facilement séparables dès lors que l'on dispose de davantage d'informations — ou que l'on en oublie, comme dans l'exemple de la figure 1.2. Le subspace clustering tel qu'introduit par Agrawal et al. (1998) consiste à chercher les clusters après projection des données sur différents sous-espaces de l'espace original. Il ignore ainsi certains attributs originaux pour former les clusters et adapte la relation de similarité localement à chaque cluster, contrairement aux techniques de réduction de dimensionnalité précédentes qui s'appliquent à l'ensemble des données. Il se distingue également des techniques de sélection d'attributs, qui produisent une nouvelle relation de similarité globale, construite avant la recherche des clusters.

Dans le but de faire apparaître cette structure, il est donc nécessaire de sélectionner les attributs adaptés à la description de chaque cluster, et seulement eux. Cette mini-

malité, appelée parcimonie (Jing et al. 2007 ; Oktar et Turkan 2018), permet également de fournir un résumé synthétique à l'utilisateur, ainsi que d'estimer la dimensionnalité intrinsèque des clusters, fournissant davantage d'informations sur l'organisation des données. Le chapitre 2 détaille cette notion de parcimonie et discute de sa formalisation, avant de présenter un certain nombre d'approches parcimonieuses.

DE L'IMPERFECTION DES SOLUTIONS À LA MINIMISATION

Dans la pratique, les données affectées à un même cluster sont rarement tout à fait similaires, même après projection sur un sous-espace. Les centres trouvés pour identifier des clusters sont généralement une synthèse approximative, et les attributs mis en avant pour décrire un cluster peuvent être plus ou moins pertinents.

De nombreuses approches de subspace clustering s'appuient sur la définition d'une *fonction de coût*, qui exprime la qualité d'une solution. Une telle fonction permet de comparer les solutions entre elles et de préférer celle dont le coût est le plus faible. Résoudre la tâche de subspace clustering est alors ramené à un problème de minimisation (Delfour 2012).

Cette approche est courante en apprentissage automatique, où des fonctions de coût, définies sur l'espace des solutions possibles, expriment par exemple l'écart entre un modèle et des données d'apprentissage ou encore entre des données et l'approximation qui en est faite. La surface de cette fonction de coût est elle-même un objet de très haute dimension, mais des techniques d'optimisation peuvent être appliquées pour guider la recherche du minimum.

Le fait de se ramener à un problème de minimisation présente plusieurs avantages. Premièrement, les fonctions de coût spécifient certaines propriétés des solutions, et permettent de raisonner sur celles-ci. D'autre part, de nombreux algorithmes, parmi lesquels les algorithmes de subspace clustering flou que nous présentons dans le chapitre 2, produisent des solutions qui partagent certaines caractéristiques, lesquelles sont reflétées au niveau de la fonction de coût, ce qui permet de les comparer à un niveau d'abstraction plus élevé.

Enfin, le cadre de l'optimisation est adapté à la notion de *régularisation*, qui consiste à exprimer des contraintes sur les solutions, qui sont ajoutées au problème initial et favorisent l'apparition de certaines propriétés au sein des solutions. Ces contraintes sont implémentées dans la fonction de coût au moyen de termes de pénalité ; les nouvelles fonctions ainsi obtenues possèdent des minima différents, orientant les algorithmes qui les minimisent vers de nouvelles solutions.

CONTRIBUTIONS

Cette thèse s'inscrit dans le paradigme du subspace clustering flou, dans lequel la tâche du subspace clustering est résolue à travers la formulation et la minimisation d'une fonction de coût. L'expression de cette fonction fait intervenir les clusters, leurs centres et les poids des dimensions, représentés par des vecteurs, ainsi qu'une distance euclidienne pondérée. Minimiser cette fonction de coût revient à apprendre les clusters et les poids des dimensions décrivant la meilleure solution de subspace clustering possible. Ces minima sont étudiés pour justifier trois améliorations.

La première de ces améliorations est la parcimonie des vecteurs de poids des attributs, correspondant à une minimalité de l'estimation des dimensions des sous-espaces. Cette minimalité est obtenue en pénalisant d'un coût additionnel, proportionnel à la dimensionnalité des sous-espaces identifiés. Nous proposons ainsi un algorithme de subspace clustering flou parcimonieux, appelé Prosecco. Nous prouvons la correction de son schéma d'optimisation, dont nous montrons qu'il produit des solutions aussi parcimonieuses que le choisit l'utilisateur de l'algorithme.

Ce travail est ensuite généralisé et la fonction de pénalité correspondante est utilisée pour autoriser les algorithmes à produire des degrés d'appartenance possibilistes. Ces degrés permettent de refléter l'importance des points dans les solutions, en témoignant notamment de leur centralité au sein des données, ce qui a pour effet de rendre les algorithmes de clustering plus résistants au bruit et aux points périphériques. Le nouvel opérateur obtenu est utilisé sur plusieurs algorithmes de subspace clustering flou, notamment Prosecco, produisant une variante possibiliste baptisée Possecco.

Enfin, nous montrons que l'affectation des points aux clusters ne tient pas compte d'informations locales telles que le voisinage des points, ce qui peut conduire à des partitionnements contre-intuitifs. Nous illustrons ce phénomène par un exemple graphique et montrons qu'il touche tous les algorithmes basés sur la fonction de coût précédente. Nous proposons une troisième fonction de pénalité, issue des algorithmes de clustering spectral, qui contraint les solutions du subspace clustering à respecter la géométrie globale des données. Un nouvel algorithme est dérivé de cette fonction de coût, baptisé WLFC pour *Weighted Laplacian Fuzzy Clustering*.

La minimisation de ces pénalités présente de nouvelles difficultés, que nous surmontons en utilisant des outils de la théorie de l'optimisation convexe, tels que l'optimisation proximale. Un cadre de travail homogène et générique est proposé, basé sur l'utilisation d'opérateurs proximaux et adapté aux spécificités du subspace clustering flou. Plusieurs algorithmes de subspace clustering originaux sont introduits.

Afin d'évaluer ces algorithmes sur des données artificielles et réelles et de comparer leurs performances, il est nécessaire de définir des critères de qualité. La nature non supervisée du clustering ainsi que les spécificités du subspace clustering, qui autorise les algorithmes à choisir des sous-espaces différents pour y retrouver des clusters, compliquent cette approche basée sur l'évaluation. Nous sommes donc conduits à proposer des méthodes d'évaluation particulières pour comparer nos différents algorithmes et faire apparaître leurs spécificités.

PLAN DE LA THÈSE

Ce manuscrit est structuré comme suit : le chapitre 2 présente l'état de l'art en subspace clustering, ainsi que le contexte technique de cette thèse.

Le schéma général des fonctions de coût et algorithmes utilisés par la suite est décrit dans le chapitre 3, qui énumère également les propriétés des fonctions de pénalité requises pour dériver ces algorithmes, telles que la proximabilité des pénalités non différentiables.

Le chapitre 4 applique ce schéma à la recherche de solutions de subspace clustering parcimonieuses, qui estiment la dimensionnalité des sous-espaces identifiés. L'opérateur proximal correspondant est dérivé dans un contexte géométrique, qui est aussi

employé pour prouver sa correction. Cet opérateur est utilisé pour proposer l'algorithme Prosecco, un nouvel algorithme de subspace clustering flou produisant une description parcimonieuse des sous-espaces.

Cet opérateur est généralisé dans le chapitre 5 pour obtenir un terme introduisant un comportement de clustering possibiliste ; l'opérateur possibiliste correspondant est générique et nous l'appliquons aux algorithmes parcimonieux étudiés dans le chapitre précédent pour proposer de nouveaux algorithmes de subspace clustering possibiliste, parmi lesquelles l'algorithme Possecco. Nous montrons expérimentalement que les algorithmes obtenus sont résistants aux données bruitées.

Enfin, le chapitre 6 propose un terme de régularisation laplacienne pour contraindre l'affectation des points aux clusters à être cohérente avec celle de leurs voisins, ainsi que l'algorithme correspondant, WLFC.

Nous concluons ce manuscrit en résumant les contributions de la thèse et en présentant les perspectives qu'elle ouvre.

2

ÉTAT DE L'ART

Comme indiqué dans le chapitre précédent, la tâche de subspace clustering vise à identifier simultanément des sous-groupes de données homogènes et distincts, ainsi que les espaces dans lesquels ils sont définis. De même que pour la tâche de clustering, de nombreuses approches de subspace clustering coexistent, qui correspondent à des modélisations des sous-espaces et à des paradigmes de clustering différents.

Ce chapitre présente l'état de l'art en subspace clustering : la section 2.1 présente la tâche de façon générale, ainsi que les notations utilisées dans ce manuscrit. La section 2.2 introduit les travaux fondamentaux en subspace clustering, qui trouvent leur origine dans le contexte du *big data*. Dans la section 2.3, les algorithmes de subspace clustering descendant de l'algorithme des k-moyennes, dérivés d'une fonction de coût, sont présentés. La section 2.4 est consacrée au problème de la parcimonie du point de vue du subspace clustering. Elle présente également des algorithmes qui induisent une description parcimonieuse des sous-espaces. Cette recherche de solutions et de représentations parcimonieuses nous conduit enfin à présenter des techniques standard d'optimisation convexe en section 2.5, qui jouent un rôle central dans le reste de la thèse.

2.1 PRÉSENTATION DU PROBLÈME DE SUBSPACE CLUSTERING

Comme évoqué en introduction, le clustering a de nombreux contextes d'utilisation. Il n'existe pas de formalisme général pour décrire cette tâche d'apprentissage non supervisé, ce qui complique la présentation d'un état de l'art général ainsi que la comparaison des multiples approches existantes. Cette difficulté est accrue dans le cas du subspace clustering, qui généralise la tâche de clustering en ajoutant la variabilité induite par la définition de sous-espaces adaptés à leurs besoins propres.

Dans cette section, nous présentons les notations utilisées dans ce manuscrit. Nous rappelons ensuite les grandes familles d'approches de clustering, en mettant l'accent sur les approches par partitionnement, qui sont à la base des algorithmes de subspace clustering considérés dans la thèse. Enfin nous présentons brièvement le cadre général du subspace clustering, qui sera repris dans les sections suivantes.

2.1.1 *Notations et conventions*

Dans toute la thèse, nous utilisons les conventions d'écriture suivantes : les grandeurs scalaires sont écrites en minuscules, les vecteurs et les matrices en majuscules. Les familles indicées comme les suites sont notées $V = (v_{ab})$. La norme euclidienne du vecteur V est abrégée $\|V\|$, les autres normes considérées sont toujours explicitées.

Les données considérées sont représentées par une matrice $X = (x_{ip}) \in \mathbb{R}^{n \times d}$, représentants 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. Pour certains algorithmes, c est spécifié par l'utilisateur, sinon il est déterminé implicitement.

Plusieurs des algorithmes que nous présentons représentent les clusters par des centres, représentés par une matrice $C = (c_{rp}) \in \mathbb{R}^{c \times d}$ de c centres (C_r), points d'un espace à d dimensions. La notation C_r est ainsi surchargée, désignant à la fois le cluster auquel sont affectés les points et son centre.

Les relations de similarité locales, ou sous-espaces, sont notées (S_r) pour $r \in [\![1, c]\!]$. Lorsqu'elles correspondent à une pondération des d attributs originaux, on utilise plutôt une matrice $W = (w_{rp}) \in \mathbb{R}^{c \times d}$ contenant les poids de chaque dimension pour chaque cluster.

2.1.2 *La tâche du 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. De nombreux paradigmes coexistent, qui correspondent à différentes définitions des clusters et sont utilisés avec des données de différentes sortes. Cette section commence par survoler les paradigmes de clustering les plus connus, avant de présenter en détail deux algorithmes de partitionnement itératif, l'algorithme des k-moyennes et sa variante, l'algorithme des c-moyennes floues. Nous présentons enfin le principe des algorithmes de clustering possibiliste, qui s'appuient sur le paradigme flou pour produire des descriptions plus riches des données.

Survol des paradigmes de clustering dominants

Bien que l'état de l'art du clustering soit vaste et qu'il soit possible de le partitionner de différentes manières (voir par exemple Kaufman et Rousseeuw (1990) et Xu et Tian (2015)), une approche classique consiste à distinguer les quatre grandes familles d'algorithme de clustering suivantes : hiérarchique, par densité, spectral et par partitionnement.

Le clustering hiérarchique (*hierarchical cluster analysis* ou HCA, voir Kaufman et Rousseeuw (1990)) regroupe les données à différents niveaux de détail, laissant le choix à l'utilisateur de la finesse des informations qu'il attend, en regroupant itérativement les petits clusters dans des structures arborescentes de clusters imbriqués. Plusieurs distances et stratégies d'agglomération sont disponibles, qui ont une influence à la fois sur les clusters identifiés et les propriétés des algorithmes de clustering (Carlsson et Mémoli 2010).

Le clustering par densité (Ester et al. 1996 ; Kriegel et al. 2011), dont l'avatar le plus connu est l'algorithme DBSCAN (Ester et al. 1996), regroupe les points en fonction d'une relation de voisinage transitive, faisant intervenir un critère de densité spécifique. Ainsi, deux points font partie d'un même cluster s'ils sont directement voisins ou si un chemin de points centraux les relie, un point étant dit *central* s'il possède un nombre minimum de voisins dans un rayon donné.

Le clustering spectral (Ng et al. 2002 ; Von Luxburg 2007) se base sur des résultats de théorie matricielle. Il consiste à former une matrice particulière, le *laplacien* de la matrice de similarité des données. Les propriétés algébriques du laplacien sont alors

mises en œuvre : les vecteurs propres du laplacien fournissent des informations permettant notamment de retrouver les clusters, alors que ses valeurs propres aident à trouver un nombre de clusters c adapté.

Enfin, les algorithmes de partitionnement itératif (Evers et al. 1999 ; Döring et al. 2006) décomposent les données en un nombre de clusters fourni par l'utilisateur, proposant ainsi une partition des données. Ces algorithmes servent de fondations à ceux que nous étudions dans cette thèse. Nous présentons ci-dessous deux algorithmes de cette famille, l'algorithme des k-moyennes et sa variante floue.

Notons finalement que les quatre familles précédentes ne forment pas une partition stricte des approches de clustering : beaucoup d'algorithmes sont *hybrides*, dans le sens où ils combinent des idées issues de plusieurs paradigmes. C'est aussi le cas pour certains des algorithmes de subspace clustering que nous considérons.

L'algorithme des k-moyennes

L'algorithme des k-moyennes (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 simples, il reste très utilisé en fouille de données.

L'algorithme des k-moyennes prend en argument le nombre k de clusters à déterminer. Il représente les clusters formés par une matrice de centres $C \in \mathbb{R}^{k \times d}$, définis dans le même espace que les données X. Chaque point X_i est affecté à exactement un cluster à travers une matrice de coefficients $U \in \{0,1\}^{k \times n}$: $u_{ri} = 1$ si et seulement si X_i est affecté au cluster C_r.

L'algorithme est dérivé de la fonction de coût suivante, minimisée sous contraintes :

$$
\begin{aligned}
K(C,U) &= \sum_{i=1}^{n} \sum_{r=1}^{k} u_{ri} \|X_i - C_r\|^2 \\
\text{sous les contraintes} \quad & (C1)\ \forall i \in [\![1,n]\!],\ \sum_{r=1}^{k} u_{ri} = 1, \\
& (C2)\ \forall r \in [\![1,r]\!],\ \sum_{i=1}^{n} u_{ri} > 0, \\
& (C3)\ \forall r, \forall i, \qquad u_{ri} \in \{0,1\}
\end{aligned}
\tag{2.1}
$$

Ce type de problème, qui sont centraux dans cette thèse, exprime une certaine fonction (notée ici K) à valeurs réelles positives, qui associe un coût à une solution (C,U) au problème de clustering. Le but est, pour des données (X_i) fixées, de trouver un couple (C^*,U^*) qui minimise K, tout en respectant les contraintes listées. Les contraintes $(C1)$ et $(C3)$ s'assurent qu'un point est affecté à exactement un cluster. La contrainte $(C2)$ force l'algorithme à trouver k clusters, sans former de cluster trivial vide.

La fonction de coût K fait apparaître le carré de la distance euclidienne entre le centre du cluster C_r et le point X_i, pondéré par le coefficient d'appartenance u_{ri}.

Comme ce dernier vaut 1 uniquement pour le cluster auquel X_i est affecté, la fonction de coût est souvent définie par :

$$K'(C) = \sum_{r=1}^{c} \sum_{i \in C_r} \|X_i - C_r\|^2 \tag{2.2}$$

où la notation C_r est surchargée, désignant à la fois le cluster auquel sont affectés les points et son centre.

Les fonctions K et K' présentent généralement plusieurs minima locaux ; un algorithme de clustering dérivé de ces fonctions tel que l'algorithme des k-moyennes est alors chargé de trouver l'un de ces minima, ce qui en pratique revient à proposer un couple (C, U) qui s'en approche autant que possible.

L'algorithme des k-moyennes est dérivé du problème 2.1 comme un algorithme d'optimisation alternée : pour affecter les données aux clusters et déterminer les centres, il calcule alternativement des matrices C et U qui minimisent la fonction K, en maintenant la valeur de l'autre fixée.

La fonction K, pour U fixée, est continue et différentiable en C. Tout minimum de K vérifie $\nabla K(C) = 0$, ce qui conduit à l'équation de mise à jour suivante :

$$C_r = \frac{\sum_{i=1}^{n} u_{ri} X_i}{\sum_{i=1}^{n} u_{ri}} = \frac{1}{|C_r|} \sum_{i \in C_r} X_i \tag{2.3}$$

c'est-à-dire que le centre C_r est calculé comme la moyenne des points affectés au cluster — donnant ainsi son nom à l'algorithme des k-moyennes.

La fonction K n'est pas différentiable en U. On peut cependant montrer que, pour C fixé, la matrice U minimisant K est donnée par

$$u_{ri} = \begin{cases} 1 & \text{si } r = \operatorname{argmin}_{s \in [\![1,k]\!]} \|X_i - C_s\|^2 \\ 0 & \text{sinon} \end{cases} \tag{2.4}$$

La définition de l'équation 2.4 est mal posée, dans le sens où plusieurs indices r peuvent convenir ; cependant un seul doit être choisi, de façon arbitraire, pour respecter les contraintes du problème 2.1.

Les deux équations de mise à jour 2.3 et 2.4 sont interdépendantes et doivent être itérées en utilisant à chaque fois la valeur la plus récente du paramètre fixé pour calculer l'autre, et ce jusqu'à stabilisation. L'algorithme standard des k-moyennes est initialisé par une solution aléatoire. Des stratégies plus efficaces ont cependant été étudiées, par exemple l'algorithme *k-means++* (Arthur et Vassilvitskii 2007).

À cause de l'ambiguïté de l'équation 2.4 mais aussi plus généralement parce que la fonction de coût K n'est pas convexe, le résultat de l'algorithme dépend fortement de son initialisation. Ceci n'est pas lié à une mauvaise définition des équations, dont on peut montrer qu'elles sont optimales pour la stratégie d'optimisation alternée. Ce phénomène concerne également tous les algorithmes dérivés des k-moyennes que nous étudions dans cette thèse. Des variantes convexes du problème des k-moyennes ont été étudiées pour remédier à ce défaut (Lindsten et al. 2011).

L'algorithme des c-moyennes floues

Dans la pratique, il est fréquent que les clusters soient mal délimités, qu'ils se chevauchent ou qu'ils partagent des points. L'algorithme des c-moyennes floues (Bezdek 1981) propose de relâcher la contrainte $(C3)$ du problème 2.1 imposant l'unicité de l'affectation des X_i aux clusters et de la remplacer par un *degré d'appartenance* $u_{ri} \in [0,1]$, qui doit néanmoins sommer à 1 pour forcer les solutions à représenter équitablement tous les points (X_i). L'algorithme gagne alors en expressivité, car il caractérise différemment les points centraux d'un cluster de ceux qui sont partagés par plusieurs d'entre eux.

La matrice U est désormais cherchée dans l'espace $[0,1]^{c\times n}$. Le nouveau problème de minimisation est le suivant, où $m > 1$ un hyperparamètre :

$$\begin{aligned} F(C,U) &= \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^m \|X_i - C_r\|^2 \\ \text{sous les contraintes} \quad (C1)\ &\forall i \in [\![1,n]\!],\ \sum_{r=1}^{c} u_{ri} = 1, \\ (C2)\ &\forall r \in [\![1,c]\!],\ \sum_{i=1}^{n} u_{ri} > 0 \end{aligned} \tag{2.5}$$

L'exposant m est rajouté au facteur u_{ri}, qui provoque l'apparition de solutions floues, c'est-à-dire qu'un point X_i appartient à plusieurs clusters $r_1, r_2 \ldots$, avec $u_{r_1 i} > 0$, $u_{r_2 i} > 0$, etc. Un choix classique est de prendre $m = 2$ (Pal et Bezdek 1995).

La fonction F est différentiable en C et en U ; un algorithme d'optimisation peut être obtenu par la méthode des multiplicateurs de Lagrange, qui permet de trouver les équations de mise à jour tout en respectant les contraintes du problème. On dérive alors les équations suivantes :

$$C_r = \frac{\sum_{i=1}^{n} u_{ri}^m \cdot X_i}{\sum_{i=1}^{n} u_{ri}^m} \tag{2.6}$$

$$u_{ri} = \frac{\|X_i - C_r\|^{\frac{2}{1-m}}}{\sum_{s=1}^{c} \|X_i - C_s\|^{\frac{2}{1-m}}} \tag{2.7}$$

La deuxième équation n'est définie que si $\forall r, X_i \neq C_r$. Dans le cas contraire, on prend $u_{ri} = 1$ pour le C_r tel que $X_i = C_r$.

L'algorithme des c-moyennes floues peut donc être compris comme une version continue des k-moyennes. L'utilisation de degrés d'appartenance flous est également plus informative : en grande dimension, il est difficile de constater que des points affectés à des clusters différents sont en réalité voisins, donc semblables, comme dans la situation de la figure 1.1 p. 2. L'algorithme des c-moyennes floues explicite ces ambiguïtés et permet donc à l'utilisateur d'obtenir une description plus fidèle des données.

Algorithmes de clustering possibiliste

Introduit par Krishnapuram et Keller (1993), le clustering possibiliste étend l'approche floue précédente, en calculant une affectation des points X_i aux clusters qui reflète leur représentativité ou leur centralité au sein du cluster. Ceci n'est pas le cas de l'algorithme des c-moyennes floues, qui partage chaque X_i entre tous les clusters en fonction de la distance de X_i aux centres des clusters mais sans tenir compte du fait que certains points sont éloignés de tous les clusters.

Dans l'optique de modéliser la représentatitivité à l'aide des degrés d'appartenance, il est nécessaire de relâcher la contrainte de sommation des c-moyennes floues, $(C1)$. Les points *outliers*, qui ne sont par définition représentatifs d'aucun cluster, peuvent ainsi être désignés comme tels par l'algorithme, qui peut leur affecter des degrés d'appartenance faibles à tous les clusters. Les algorithmes de clustering possibiliste sont donc intéressants lorsque les données sont bruitées, c'est-à-dire que des points parasites, indésirés ou peu informatifs sont présents dans les données.

Krishnapuram et Keller (1993) proposent de supprimer la contrainte de sommation et la remplacent par une pénalité particulière, dépendant d'une famille (η_r) d'hyperparamètres strictement positifs :

$$\begin{aligned} J(C,U) &= \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^m \|X_i - C_r\|^2 + \sum_{r=1}^{c}\eta_r \sum_{i=1}^{n}(1-u_{ri})^m \\ \text{sous les contraintes} \quad &(C1)\ \forall r \in [\![1,c]\!],\ \sum_{i=1}^{n} u_{ri} > 0 \end{aligned} \tag{2.8}$$

La contrainte $(C1)$ sert à s'assurer qu'aucun cluster produit n'est vide et que les degrés d'appartenance calculés sont bien compris entre 0 et 1. Elle n'a cependant pas de traduction dans le lagrangien du problème 2.8. Krishnapuram et Keller (1993) dérivent l'équation de mise à jour suivante pour u_{ri} :

$$u_{ri} = \frac{1}{1 + \left(\frac{\|X_i - C_r\|^2}{\eta_r}\right)^{\frac{1}{m-1}}} \tag{2.9}$$

qui satisfait bien les conditions précédentes. De plus, les auteurs remarquent que cette équation ne fait intervenir, pour un point et un cluster donnés, que la distance qui les sépare : contrairement à l'équation de mise à jour des c-moyennes floues et des algorithmes qui en sont dérivés, le degré u_{ri} calculé ne dépend pas des autres clusters, mais seulement de la représentativité de X_i au sein de C_r.

Krishnapuram et Keller (1993) proposent un algorithme d'optimisation alternée, appelé PCM (pour *Possibilistic c-Means*, ou c-moyennes possibilistes), faisant intervenir l'équation précédente et une équation de mise à jour des centres classique. L'équation 2.9 montre que le degré u_{ri} décroît avec la distance au cluster C_r : les c-moyennes possibilistes sont plus résistantes que les c-moyennes floues aux *outliers* et donc aux données bruitées.

En revanche, Barni et al. (1996) montrent que l'algorithme converge fréquemment vers une solution (C,U) dans laquelle plusieurs clusters sont quasiment confondus. Les auteurs expliquent ce comportement par l'absence de « lien » entre les clusters : à la différence de l'équation 2.7, l'équation 2.9 de mise à jour d'un degré u_{ri} ne fait

intervenir que le cluster C_r ; par conséquent, le minimum global de la fonction de coût est atteint en une solution (C^*, U^*) qui uniformise toutes les valeurs des (u_{ri}) et donc des (C_r).

L'algorithme PCM ne fournit donc de résultats intéressants que lorsqu'il converge vers un minimum local et que le minimum global de la fonction n'est *pas* atteint, ce qui est peut satisfaisant d'un point de vue conceptuel.

2.1.3 *La tâche de subspace clustering*

Le subspace clustering est une généralisation du clustering qui vise à identifier simultanément les clusters et les sous-espaces dans lesquels ils existent. Une approche classique de cette formulation interprète l'identification des sous-espaces comme la détermination de relations de similarité locales à chaque cluster, caractéristiques de ces sous-espaces.

Certains clusters peuvent être denses dans l'espace $\mathbb{R}^d$, mais le plus souvent ils ne le sont que dans des sous-espaces qui doivent être identifiés par l'algorithme, comme c'était le cas sur la figure 1.2 p. 3.

La spécificité du subspace clustering réside dans le fait que l'identification des sous-espaces dépend de celle des clusters, et réciproquement. Le subspace clustering se distingue en cela d'une simple sélection d'attributs locale à chaque cluster, qui supposerait que les clusters soient déjà formés, ou d'une sélection d'attributs globale en amont de la tâche de clustering. Il est donc une généralisation à la fois du clustering et de certaines techniques de réduction de la dimensionnalité.

Comme en clustering, il existe de nombreux paradigmes de subspace clustering, qui cherchent des sous-espaces de nature et de forme différentes. Les approches originales (CLIQUE et PROCLUS, voir section 2.2) cherchent des sous-espaces construits d'après les attributs originaux des données. On parle de sous-espaces *parallèles aux axes*. D'autres approches autorisent des rotations des axes originaux (l'algorithme de Gustafson-Kessel, présenté dans la section 2.3.5 p. 24) et se rapprochent ainsi d'une analyse en composantes principales locale à chaque cluster.

De même, les notions de « groupes denses » ou de « similarité interne » sont très variables : si certains algorithmes s'appuient sur une définition explicite de la densité (section 2.2), d'autres partitionnent les données en k clusters, sans garantie de densité, où k est un paramètre choisi par l'utilisateur, généralisant ainsi l'algorithme des k-moyennes.

Enfin, le subspace clustering peut être comparé à d'autres généralisations du clustering qui cherchent à caractériser les clusters et les attributs qui les définissent. L'une de ces approches, le *co-clustering* (ou *biclustering*, voir Mirkin (1996)) cherche ainsi à regrouper non seulement les données en clusters, mais aussi leurs attributs caractéristiques, obtenant aussi une décomposition par blocs de la matrice X. Comme pour le subspace clustering, une relation de similarité locale aux clusters et adaptée à ces blocs d'attributs est, au moins implicitement, construite. Le co-clustering diffère pourtant du subspace clustering en deux points : d'une part en construisant une relation de similarité entre les attributs alors que celle-ci n'existe pas en subspace clustering ; d'autre part parce que les sous-espaces ne sont pas vraiment indépendants, cette opération de regroupement des attributs étant globale.

2.2 DES ALGORITHMES ISSUS DU *BIG DATA*

Le terme de *subspace clustering* a été introduit par Agrawal et al. (1998), qui se placent dans le cadre de la fouille de données dans des espaces de très grande dimension et proposent l'algorithme CLIQUE. Dans ces espaces, il est nécessaire de projeter les points sur des sous-espaces pour former des clusters, qui ne sont pas denses dans l'espace de départ. Différentes projections peuvent alors convenir, regroupant les données dans des clusters différents.

Deux familles d'approches ont émergé : la première, présentée en section 2.2.1 p. 16, consiste à énumérer tous les clusters et sous-espaces possibles, ce qui conduit à un temps de calcul élevé. La deuxième, initialement appelée *projected clustering* et rangée dans une famille distincte du subspace clustering, partitionne au contraire les points en clusters disjoints, eux-mêmes projetés dans un seul sous-espace par cluster. Présentée en section 2.2.2, cette approche, qui est par la suite devenue prédominante dans le domaine du subspace clustering, perd donc en exhaustivité, mais gagne en efficacité.

2.2.1 *L'algorithme CLIQUE*

CLIQUE (Agrawal et al. 1998) est le premier-né de la famille des *algorithmes itératifs bottom-up*. Dans cette famille, le but est d'identifier tous les clusters et donc tous les sous-espaces qui les contiennent, puis d'en produire une description textuelle aussi courte que possible.

CLIQUE cherche les groupes denses dans tous les sous-espaces $\mathbb{R}^p$, $p \leq d$. Pour cela, il découpe l'espace des données $\mathbb{R}^d$ en une grille de cubes unitaires u de côté δ parallèles aux axes, où δ est un hyperparamètre fourni par l'utilisateur. L'algorithme commence par former des cubes denses en basse dimension et utilise un principe de monotonie pour reconstruire les clusters en plus haute dimension, suivant une procédure incrémentale.

Étant donné u un cube unitaire de $\mathbb{R}^p$, on note $\text{card}(u)$ le nombre de points (X_i) contenus dans u et $d(u) = \frac{\text{card}(u)}{\text{card}(X)}$ sa *densité* ; un cube est alors dit *dense* si et seulement si $d(u) \geq \tau$, où τ est un autre hyperparamètre. On utilise également la notation u^p pour désigner un cube unitaire de dimension p.

Clusters et sous-espaces

Pour CLIQUE, les sous-espaces (S_r) sont des ensembles maximaux de cubes unitaires adjacents parallèles aux axes, tous denses. Les clusters sont définis comme les pavés rectangulaires inclus dans ces sous-espaces ; ces pavés sont en outre maximaux, c'est-à-dire qu'ils ne sont inclus dans aucun autre cluster. Un même sous-espace peut donc contenir plusieurs clusters si les cubes qui le constituent ne forment pas un pavé entier.

Présentation de l'algorithme

L'algorithme CLIQUE identifie les sous-espaces par dimension croissante, en partant de sous-espaces de dimension 1 qui sont ensuite assemblés en espaces de dimension 2, puis 3, etc. Il comporte trois phases :

- Recherche des sous-espaces d'intérêt : CLIQUE commence par former les cubes u^1 denses. Ensuite, pour chaque $p \in [\![2, d]\!]$, il identifie les cubes u^p denses à partir des cubes u^{p-1} denses, en utilisant une propriété de *monotonie* inspirée de l'algorithme APRIORI (Agrawal et Srikant 1994) : si un cube u^p contenant des points projetés dans un espace de dimension p n'est pas dense, alors il ne fait partie d'aucun sous-espace dense de dimension supérieure et il est inutile de chercher des sous-espaces contenant u^p.

  Les *cubes unitaires maximaux*, notés u^p_M, sont les cubes unitaires denses de dimension p qui ne sont contenues dans aucun $u^{p'}$ pour $p' > p$. Les sous-espaces S^p_r sont définis comme les unions maximales de cubes unitaires maximaux u^p_M adjacents en dimension p. Ils ne forment pas nécessairement des pavés.

- Identification des clusters : les clusters sont les pavés maximaux contenus dans ces sous-espaces. Ainsi, un même sous-espace composé de cubes unitaires adjacents qui ne forment pas un pavé peut contenir plusieurs clusters.

- Description des clusters : après identification des sous-espaces intéressants, CLIQUE produit une description textuelle des clusters contenus dans les sous-espaces, qui résume chaque cluster par les bornes du pavé correspondant, pour tous les attributs du sous-espace correspondant.

Discussion

Malgré l'utilisation de la propriété de monotonie pour élaguer l'espace de recherche, CLIQUE est un algorithme potentiellement coûteux en grande dimension. De plus, les deux paramètres δ et τ sont difficiles à choisir (Parsons et al. 2004). Une grille trop fine (correspondant à un δ trop faible) entraîne un temps de calcul trop élevé, alors qu'une grille trop large est peu adaptée à l'identification de clusters de tailles variées. Enfin, parce qu'il cherche à produire des sous-espaces maximaux, CLIQUE produit des descriptions longues et peu lisibles.

Plusieurs algorithmes reprennent le principe de CLIQUE pour apporter différentes modifications : l'algorithme ENCLUS (Cheng et al. 1999), basé sur une minimisation de l'entropie des cellules plutôt que sur une maximisation de la densité ; l'entropie permet de chercher des sous-espaces qui sont non seulement denses, mais qui contiennent également des valeurs d'attributs décorrélées. ENCLUS remplace le seuil de densité τ par un hyperparamètre équivalent ω qui définit le seuil d'entropie en dessous duquel un sous-espace est jugé pertinent, ce qui semble encore plus difficile à choisir pour l'utilisateur. MAFIA (Burdick et al. 2001) améliore la grille de largeur fixe de CLIQUE pour proposer une grille adaptative, basée sur la distribution des valeurs selon chaque attribut, gagnant ainsi en flexibilité par rapport à CLIQUE mais sans pour autant changer de paradigme : CLIQUE et ses variantes cherchent à construire tous les clusters qui sont denses dans des sous-espaces différents. Outre le coût calculatoire qu'il induit, ce choix a pour conséquence qu'un même point peut être affecté à plusieurs clusters et que la description de la solution produite par l'algorithme peut être relativement longue.

2.2.2 *Algorithmes de projected clustering*

Le *projected clustering* (Aggarwal et al. 1999) se distingue du paradigme précédent en cherchant une partition stricte des données. De plus, les algorithmes de cette famille identifient les sous-espaces et les clusters par dimension décroissante, c'est-à-dire en commençant par les chercher dans $\mathbb{R}^d$ puis en raffinant les sous-espaces itérativement. Cette approche est par la suite devenue standard en subspace clustering en raison de son efficacité calculatoire.

Le premier exemple d'algorithme de cette famille est l'algorithme PROCLUS (Aggarwal et al. 1999), basé sur l'algorithme des k-médoïdes (une variante des k-moyennes).

Clusters et sous-espaces

Les clusters (C_r) sont identifiés par des médoïdes notés (m_r), c'est-à-dire des points centraux pris parmi les données (Kaufman et Rousseeuw 1987). PROCLUS produit également un ensemble particulier d'*outliers*, points qui ne sont affectés à aucun cluster. Les sous-espaces (S_r) sont parallèles aux axes et sont représentés par un ensemble de dimensions sélectionnées.

PROCLUS définit plusieurs grandeurs géométriques (distances, ensembles de points, etc.) en utilisant la distance de Manhattan, notée $d(x, y)$, alléguant qu'elle présente davantage de sens dans la plupart des applications de l'algorithme.

Présentation de l'algorithme

L'algorithme PROCLUS nécessite deux hyperparamètres k et l qui sont respectivement le nombre de clusters à identifier dans les données et leur dimensionnalité, détaillé ci-dessous. PROCLUS est initialisé par tirage aléatoire de k médoïdes $(m_r)_{r=1..k}$ et de sous-espaces (S_r) tous égaux à $\mathbb{R}^d$, dont les dimensions inutiles vont être progressivement éliminées.

À chaque itération, l'algorithme détermine le sous-espace S_r sur lequel projeter le cluster C_r : il calcule d'abord L_r, l'ensemble des points voisins de m_r, définis comme $L_r = \{x \mid \forall s, d(x, m_r) \leq d(x, m_s)\}$. Sont définis ensuite :

- la distance moyenne D_{rp} de L_r à m_r selon la dimension p, $D_{rp} = moy\left\{|x_{ip} - m_{rp}| \;\middle|\; X_i \in L_r\right\}$;
- la distance moyenne D_r entre les points de L_r et m_r ;
- l'écart-type σ_r des distances D_{rp} à D_r ;
- la grandeur $E_{rp} = \frac{D_{rp} - D_r}{\sigma_r}$, qui témoigne de l'utilité de la dimension p pour décrire le cluster C_r : une valeur négative de E_{rp} indique que les points de C_r sont significativement proches dans la dimension p.

PROCLUS sélectionne ensuite les $k \times l$ couples (r, p) qui minimisent E_{rp} sous la contrainte que tous les S_r soient de dimension 2, et détermine ainsi les dimensions de chaque sous-espace : le calcul de la dimension d'un sous-espace dépend donc non

seulement du paramètre l mais également des autres $E_{r'p}$. Après le calcul des dimensions, les points X_i sont réaffectés au cluster C_r qui minimise la distance $d(\pi_r(X_i), m_r)$ entre m_r et le projeté de X_i sur S_r, noté $\pi_r(X_i)$.

Une phase finale, optionnelle, permet d'éliminer les points (X_i) affectés aux clusters (C_r) mais qui en sont trop éloignés. Ces points, appelés *outliers*, ne ressemblent à ceux d'aucun cluster. Leur identification explicite est une information supplémentaire fournie par l'algorithme quand les données sont en haute dimension et sont donc difficiles à représenter graphiquement. PROCLUS identifie les outliers avec le critère suivant : X_i est un outlier si

$$d(\pi_r(X_i), m_r) \geq \min_{s \neq r} d(\pi_r(m_s), m_r)$$

c'est-à-dire qu'il est plus loin du centre auquel il est affecté que le plus proche des autres centres.

Discussion

En théorie, PROCLUS identifie moins de clusters et de sous-espaces que CLIQUE et produit une partition stricte des données ; il est en contrepartie plus rapide. Comme CLIQUE, l'algorithme identifie des sous-espaces parallèles aux axes. Enfin, PROCLUS repose sur des paramètres particuliers qui peuvent être difficiles à estimer. Parsons et al. (2004) expliquent en outre que la méthode utilisée par l'algorithme pour identifier les sous-espaces est défavorable aux clusters de grande dimension.

L'algorithme ORCLUS (Aggarwal et Yu 2000) généralise PROCLUS pour identifier des sous-espaces non parallèles aux axes au moyen d'une matrice de covariance, comme pour l'algorithme de Gustafson-Kessel (voir section 2.3.5). À chaque itération, les clusters dont les centres sont proches et dont les directions principales sont identiques sont en outre fusionnés.

2.3 ALGORITHMES DÉRIVÉS D'UNE FONCTION DE COÛT

Les approches par partitionnement itératif, tels que les algorithmes des k-moyennes et des c-moyennes floues présentés en section 2.1.2, sont également applicables au subspace clustering. Certains d'entre eux, comme PROCLUS, se basent sur un algorithme ad-hoc pour identifier les centres des clusters puis les dimensions significatives.

Ceux que nous présentons dans cette section reposent au contraire sur une fonction de coût à valeurs positives qui exprime la qualité d'une solution : selon les données X, son minimum ou ses minima correspondent aux solutions de subspace clustering que l'on recherche. Les algorithmes dérivés de ces fonctions de coût sont des algorithmes d'optimisation chargés de trouver l'un de ces minima, fût-il seulement local.

Après avoir rappelé les caractéristiques générales de ces algorithmes en section 2.3.1, nous en détaillons quatre, en insistant à chaque fois sur la fonction de coût proposée et les équations de mise à jour dérivées pour la minimiser. Le choix que nous faisons, volontairement restreint, est un échantillon représentatif de la diversité des fonctions de coût et des équations de mise à jour associées en subspace clustering.

2.3.1 *Caractéristiques générales*

Comme en section 2.1.2, les clusters (C_r) identifiés forment une partition des données X, qui peut être stricte ou floue selon les algorithmes. Les clusters sont décrits par un centre $C_r \in \mathbb{R}^d$. Les points (X_i) sont affectés aux clusters au moyen d'une matrice de degrés d'appartenance $U = u_{ri}$, avec $U \in [0,1]^{c \times n}$ pour les algorithmes flous, et $U \in \{0,1\}^{c \times n}$ pour les autres. Le nombre de clusters c à identifier est fourni par l'utilisateur et aucun cluster vide n'est admis.

Sauf pour l'algorithme de Gustafson-Kessel détaillé en section 2.3.5, les sous-espaces sont parallèles aux axes et ne sont pas explicitement construits mais caractérisés par une matrice de poids $W = (w_{rp})$ définissant une mesure de distance locale aux sous-espaces, sous la forme d'une distance pondérée : plus w_{rp} est faible, moins la dimension p est considérée comme significative pour le cluster C_r. En règle générale, les poids des dimensions sont définis sur $[0,1]$; c'est le cas des algorithmes AWFCM (section 2.3.2) et FSC (section 2.3.3) présentés dans cette section, ainsi que de la plupart des algorithmes de partitionnement présentés dans la section 2.4 ou de nos propres algorithmes, introduits dans des chapitres ultérieurs. Cela n'est cependant pas une règle générale, comme le montrent les algorithmes de Gustafson-Kessel ou l'algorithme FL1AD (section 2.3.4).

Plutôt que des sous-espaces, cette matrice W définit donc une relation de distance locale aux clusters, contenant des degrés analogues aux degrés d'appartenance du clustering flou. Cette analogie entre les matrices U et W est visible à plusieurs niveaux : dans les fonctions de coût proposées, où les mêmes opérateurs leurs sont appliqués, mais également dans les équations de mise à jour obtenues.

Pour cette raison, certains auteurs utilisent parfois l'expression *"fuzzy subspace clustering"* pour désigner des variantes pondérées de l'algorithme des k-moyennes, alors que d'autres préfèrent le nom de *"soft subspace clustering"*. On consultera par exemple Deng et al. (2016), qui énumèrent et classifient ainsi plus d'une trentaine d'algorithmes de soft subspace clustering publiés depuis les années 2000.

2.3.2 *Attribute Weighting Fuzzy c-Means*

Keller et Klawonn (2000) introduisent une fonction de coût dérivée de celle des c-moyennes floues, dans laquelle la distance euclidienne globale est remplacée par des distances pondérées par les w_{rp}. L'algorithme correspondant, appelé *Attribute Weighted Fuzzy C-Means* (AWFCM), calcule la solution (C, U, W) par optimisation alternée.

Fonction de coût

Étant donné une famille d'hyperparamètres strictement positifs (α_r) et un $v > 1$, la fonction de coût proposée est la suivante :

$$
\begin{aligned}
F_K(C, U, W) &= \sum_{i=1}^{n} \sum_{r=1}^{c} u_{ri}^m \sum_{p=1}^{d} w_{rp}^v (x_{ip} - c_{rp})^2 \\
\text{sous les contraintes} \quad (C1)\ & \forall i \in [\![1, n]\!],\ \sum_{r=1}^{c} u_{ri} = 1, \\
(C2)\ & \forall r \in [\![1, c]\!],\ \sum_{i=1}^{n} u_{ri} > 0, \\
(C3)\ & \forall r \in [\![1, c]\!],\ \sum_{p=1}^{d} w_{rp} = \alpha_r
\end{aligned}
\tag{2.10}
$$

La fonction F_K enrichit la fonction de coût initiale des c-moyennes floues de l'équation 2.5 p. 13, remplaçant la distance euclidienne par une distance euclidienne pondérée. Keller et Klawonn ajoutent un facteur w_{rp} spécifiant le poids de la dimension p pour le cluster C_r, ainsi qu'un hyperparamètre v jouant un rôle analogue à m. Dans la pratique, on choisit généralement $m = v = 2$.

Aux deux premières contraintes $(C1)$ et $(C2)$ héritées des c-moyennes floues, ils ajoutent la contrainte $(C3)$, qui dépend d'une famille d'hyperparamètres (α_r). Cette contrainte interdit la solution triviale $W = 0$. De plus, les (α_r) peuvent aussi être utilisés pour moduler l'importance des différents clusters dans la solution du problème 2.10 : un α_r plus grand augmente le coût du cluster C_r associé dans la fonction. À notre connaissance, cet effet n'a cependant jamais été étudié et dans la pratique $\forall r, \alpha_r = 1$.

Minimisation de la fonction de coût

La fonction F_K est positive, convexe en chacune de ses variables (mais pas en le triplet (C, U, W)) et différentiable. Comme pour les c-moyennes floues (voir section 2.1.2 p. 13), Keller et Klawonn (2000) forment le lagrangien du problème 2.10 et en déduisent les équations de mise à jour pour les deux paramètres C et U ci-dessous. Celles-ci sont identiques aux équations 2.6 et 2.7 dérivées pour les c-moyennes floues, au changement de distance près pour U :

$$
c_{rp} = \frac{\sum_{i=1}^{n} u_{ri}^m \cdot x_{ip}}{\sum_{i=1}^{n} u_{ri}^m} \tag{2.11}
$$

$$
u_{ri} = \frac{d_{ri}^{\frac{2}{1-m}}}{\sum_{s=1}^{c} d_{si}^{\frac{2}{1-m}}} \quad \text{où} \quad d_{ri}^2 = \sum_{p=1}^{d} w_{rp}^v (x_{ip} - c_{rp})^2 \tag{2.12}
$$

ainsi qu'une équation originale pour la matrice W :

$$w_{rp} = \frac{s_{rp}^{\frac{2}{1-v}}}{\sum_{q=1}^{d} s_{rq}^{\frac{2}{1-v}}} \quad \text{où} \quad s_{rp}^2 = \sum_{i=1}^{n} u_{ri}^m (x_{ip} - c_{rp})^2 \tag{2.13}$$

dans laquelle s peut être interprété comme la variance pondérée du cluster r selon la dimension p.

L'algorithme AWFCM consiste alors à mettre en œuvre un schéma d'optimisation alternée qui itère ces trois équations de mise à jour jusqu'à convergence. L'initialisation est effectuée le plus souvent par une affectation aléatoire des points aux clusters, définissant un U initial qui permet de calculer C grâce à l'équation 2.11, puis W.

2.3.3 *Fuzzy Subspace Clustering*

Comme l'équation de mise à jour de U des c-moyennes floues (voir équation 2.7), l'équation 2.13 peut ne pas être définie dans des cas très particuliers. Ce problème est résolu par Gan et Wu (2008), qui présentent un algorithme de partitionnement strict, FSC (*Fuzzy Subspace Clustering*) qui pondère également les attributs initiaux localement à chaque cluster, basé sur la fonction de coût suivante, avec deux hyperparamètres $\varepsilon > 0$ et $v > 1$:

$$F_{FSC}(C, U, W) = \sum_{i=1}^{n} \sum_{r=1}^{c} u_{ri} \sum_{p=1}^{d} w_{rp}^v (x_{ip} - c_{rp})^2 + \varepsilon \sum_{r=1}^{c} \sum_{p=1}^{d} w_{rp}^v$$

$$\text{sous les contraintes} \quad \begin{aligned} &(C1)\ \forall i \in [\![1, n]\!], \ \sum_{r=1}^{c} u_{ri} = 1, \\ &(C2)\ \forall i, r, \qquad u_{ri} \in \{0, 1\}, \\ &(C3)\ \forall r \in [\![1, c]\!], \ \sum_{p=1}^{d} w_{rp} = 1 \end{aligned} \tag{2.14}$$

Le rôle de ε est d'empêcher les facteurs (s_{rp}) de l'équation 2.13 de s'annuler. En effet, le premier terme de cette fonction de coût est celui de la fonction F_K, la nouvelle équation de mise à jour est la suivante :

$$w_{rp} = \frac{s_{rp}^{\frac{2}{1-v}}}{\sum_{q=1}^{d} s_{rq}^{\frac{2}{1-v}}} \quad \text{où} \quad s_{rp}^2 = \sum_{i=1}^{n} u_{ri} (x_{ip} - c_{rp})^2 + \varepsilon \tag{2.15}$$

Ce paramètre ε est généralement choisi proche de 0, sa seule fonction étant d'assurer que l'équation 2.15 est bien définie. L'algorithme FSC est un algorithme d'optimisation alternée classique. En outre, Gan et Wu (2008) fournissent une preuve de convergence de leur algorithme, basée sur le théorème de Zangwill (1969).

2.3.4 *Fuzzy L1 Adaptative Distances*

de Carvalho et Pimentel (2012) proposent d'étudier une fonction de coût dans laquelle la distance euclidienne pondérée est remplacée par une distance de Manhattan pondérée. Ils considèrent le problème suivant :

$$\begin{aligned} F_{\ell_1}(C,U,W) &= \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^{m} \sum_{p=1}^{d} w_{rp} \left|x_{ri} - c_{rp}\right| \\ \text{sous les contraintes} \quad (C1)\ & \forall i \in [\![1,n]\!],\ \sum_{r=1}^{c} u_{ri} = 1, \\ (C2)\ & \forall r \in [\![1,c]\!],\ \sum_{i=1}^{n} u_{ri} > 0, \\ (C3)\ & \forall r \in [\![1,c]\!],\ \prod_{p=1}^{d} w_{rp} = 1 \end{aligned} \tag{2.16}$$

Les contraintes $(C1)$ et $(C2)$ sont les mêmes que celles du problème 2.5. En revanche, la contrainte $(C3)$ porte sur le produit et non la somme des poids (w_{rp}) de chaque dimension au sein de chaque cluster : un poids w_{rp} calculé peut être inférieur à 1 si la dimension p est peu importante pour le cluster C_r et supérieur à 1 sinon. de Carvalho et Pimentel (2012) proposent un algorithme d'optimisation alternée, *Fuzzy L_1 Adaptative Distances* (FL1AD) qui diffère des algorithmes précédent en ceci qu'il ne repose pas uniquement sur des équations de mise à jour.

L'utilisation de distance de Manhattan à la place de la distance euclidienne est inspirée de la méthode des moindres déviations (*least absolute deviation*, voir Fuchs (2010) et Bloomfield et Steiger (1984)), une alternative à la méthode des moindres carrés qui permet de gagner en robustesse, mais qui perd un certain nombre d'autres propriétés comme l'unicité de la solution. La non-différentiabilité de la valeur absolue interdit l'utilisation de la technique standard : il est possible de dériver une équation pour U et W, mais pas pour C, ce qui oblige à avoir recours à des algorithmes de minimisation (Jajuga 1991).

Nous détaillons l'algorithme de calcul des centres (C_r) employé par de Carvalho et Pimentel (2012). Comme la somme de la fonction F_{ℓ_1} est une somme de fonctions positives indépendantes, il suffit de minimiser chacune d'elles indépendamment. Pour déterminer (c_{rp}) on doit donc résoudre

$$\operatorname*{argmin}_{c_{rp}\in\mathbb{R}} \sum_{i=1}^{n} u_{ri}^{m} \left|x_{ri} - c_{rp}\right|$$

Ensuite,

- soit $(z_i)_{i=1}^{n}$ la suite des u_{ri}^{m} triés par valeurs de x_{ip} croissantes, c'est-à-dire que $z_1 = u_{ri_1}^{m}$ tel que $x_{i_1 p}$ est le plus petit (x_{ip}) et ainsi de suite ;
- soit $j_0 = \operatorname{argmin}_{j\in[\![1,n]\!]} \sum_{i=1}^{j} z_i - \sum_{i=j+1}^{n} z_i$
- si ce minimum est négatif, on prend $c_{rp} = x_{j_0 p}$. S'il est positif, on prend $c_{rp} = x_{j_0+1 p}$. S'il est égal à 0, on prend la moyenne des deux.

Cette procédure, appliquée à tous les (c_{rp}), construit une nouvelle matrice C.

La dérivation des équations de mise à jour pour U et W est plus standard et peut être faite par la méthode des multiplicateurs de Lagrange. L'équation de mise à jour de U est classique ; celle de W est donnée par

$$w_{rp} = \frac{\left(\prod_{q=1}^{d}\left[\sum_{i=1}^{n} u_{ri}^{m}\left|x_{iq} - c_{rq}\right|\right]\right)^{\frac{1}{d}}}{\sum_{i=1}^{n} u_{ri}^{m}\left|x_{ip} - c_{rp}\right|} \tag{2.17}$$

2.3.5 *Algorithme de Gustafson-Kessel*

Les algorithmes de subspace clustering précédents reposent sur l'utilisation d'une distance pondérée dont les vecteurs de poids sont appris localement, correspondant à des sous-espaces parallèles aux axes. D'autres fonctions de distance peuvent cependant être utilisées, menant à des solutions de subspace clustering différentes.

L'algorithme de Gustafson-Kessel (Gustafson et Kessel 1978), introduit plusieurs dizaines d'années avant les premiers algorithmes de subspace clustering, peut néanmoins être vu comme une variante des algorithmes précédents qui s'appuie sur la distance de Mahalanobis (De Maesschalck et al. 2000). Celle-ci fait intervenir l'inverse de la matrice de covariance $\Sigma_r \in \mathbb{R}^{d\times d}$ des données du cluster. Cela revient à chercher des clusters dans des sous-espaces ellipsoïdaux, non nécessairement parallèles aux axes.

Nous commençons par rappeler le principe de l'algorithme de Gustafson-Kessel, avant de présenter des travaux sur la régularisation de celui-ci, qui identifient des propriétés désirables propres aux solutions de cet algorithme ainsi que des propositions qui permettent de les faire apparaître dans les solutions — donnant ainsi un avant-goût des chapitres à venir.

Fonction de coût et équation de mise à jour

Le nouveau problème de minimisation considéré est, avec (ρ_r) une famille d'hyperparamètres strictement positifs :

$$\begin{aligned} F_{GK}(C, U, \Sigma) &= \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^{m}\left(x_i - c_r\right)^{\top}\Sigma_r^{-1}\left(x_i - c_r\right) \\ \text{sous les contraintes} \quad &(C1)\ \forall i \in [\![1, n]\!],\ \sum_{r=1}^{c} u_{ri} = 1, \\ &(C2)\ \forall r \in [\![1, c]\!],\ \sum_{i=1}^{n} u_{ri} > 0, \\ &(C3)\ \forall r \in [\![1, c]\!],\ \det(\Sigma_r^{-1}) = \rho_r \end{aligned} \tag{2.18}$$

Les $(\rho_r) > 0$ empêchent la solution triviale $\Sigma_r = 0$. Ils correspondent à une notion de « volume » des clusters, dont la contrainte $(C3)$ s'assure qu'il reste constant.

Les (ρ_r) jouent un rôle analogue aux paramètres (α_r) du problème 2.10 de l'algorithme AWFCM (voir section 2.3.2 p. 20).

Gustafson et Kessel (1978) dérivent les équations de mise à jour suivantes pour Σ_r, les autres étant inchangées (à l'utilisation de Σ près) :

$$\Sigma_r = \frac{S_r}{\sqrt[d]{\rho_r \det(S_r)}} \tag{2.19}$$

$$\text{où} \quad S_r = \frac{\sum_{i=1}^{n} u_{ri}^m \, (x_i - c_r) \, (x_i - c_r)^\top}{\sum_{i=1}^{n} u_{ri}^m}$$

Les matrices Σ_r et leurs inverses peuvent de plus être contraintes pour ne contenir que des termes diagonaux, c'est-à-dire que les covariances calculées sont toutes nulles : les sous-espaces identifiés sont alors parallèles aux axes. Cette variante de l'algorithme de Gustafson-Kessel est connue sous le nom d'algorithme de Gath et Geva (Gath et Geva 1989). Sans cette contrainte, l'algorithme de Gustafson-Kessel identifie des clusters ellipsoïdaux mais dont les axes ne sont pas parallèles aux axes de $\mathbb{R}^d$. Borgelt (2010) propose une décomposition des matrices (Σ_r) faisant apparaître un facteur de rotation et de pondération des axes séparés et permettant de dériver une variante de l'algorithme de la section 2.4.1 incluant une rotation des axes locale à chaque cluster.

Cette thèse se concentre sur les sous-espaces parallèles aux axes ; nous récapitulons néamoins deux contributions en matière de régularisation de la fonction de coût de Gustafson-Kessel.

Travaux sur la régularisation de Gustafson-Kessel

Comme les clusters ellipsoïdaux de l'algorithme de Gustafson-Kessel ne sont pas nécessairement parallèles aux axes, le nombre élevé de paramètres devant être calculés par l'algorithme le rend peu robuste en haute dimension. Borgelt et Kruse (2004) proposent de maîtriser cet effet en s'inspirant des méthodes de régularisation standard, telles que la régularisation de Tikhonov. Ils distinguent deux types de régularisation, la forme et la taille des clusters.

La régularisation sur la forme vise à moduler l'allongement des ellipses, c'est-à-dire le rapport entre ses axes. Borgelt et Kruse (2004) proposent de remplacer l'équation 2.19 de mise à jour des matrices de covariance par

$$\Sigma_r = \frac{S_r + h^2 \mathbb{1}}{\sqrt[d]{\rho_r \det(S_r + h^2 \mathbb{1})}}$$

où $\mathbb{1}$ est une matrice carrée de 1 et h un paramètre de régularisation. Ce nouveau terme de mise à jour ajoute une valeur fixée h à toutes les valeurs propres de la matrice Σ_r et renormalise la matrice pour respecter la contrainte $(C3)$ du problème 2.18. Ceci a pour conséquence d'égaliser les valeurs propres de Σ_r, donc la longueur des axes des ellipses.

La régularisation sur la taille vise à équilibrer le volume ρ_r des ellipsoïdes. Étant donné deux paramètres t et b, la nouvelle formule de calcul de ce volume est donnée par :

$$\rho_r = \sqrt[d]{t \frac{\sum_{s=1}^{c} \rho_s^d}{\sum_{s=1}^{c} (\rho_s^d + b)} (\rho_r^d + b)}$$

Cette nouvelle équation a pour effet d'ajouter b à chaque taille et de la renormaliser après, de façon à maintenir constante la somme des tailles des clusters. Cependant, le paramètre s permet également de multiplier la taille par une constante. Cette nouvelle équation de mise à jour pour ρ_r, qui était auparavant un hyperparamètre de l'algorithme, peut être intégrée à l'algorithme de Gustafson-Kessel.

Ces deux propositions illustrent la régularisation pour un problème particulier : contrôler des propriétés des solutions d'un algorithme de subspace clustering au moyen de plusieurs paramètres choisis par l'utilisateur de l'algorithme. Dans cette thèse, nous proposons d'ajouter plusieurs termes de régularisation à une même fonction de coût pour faire apparaître différents effets. Deux de ces effets sont liés à la propriété de parcimonie, que nous détaillons dans la section suivante.

2.4 L'ÉMERGENCE D'APPROCHES PARCIMONIEUSES

Alors que les approches de la section 2.2 reposent sur une construction explicite des sous-espaces, celles de la section 2.3 calculent des vecteurs de poids (W_r) qui décrivent l'importance relative des différentes dimensions mais qui sont tous strictement supérieurs à 0, signifiant que toutes les dimensions sont plus ou moins importantes pour tous les clusters.

Pour retrouver et présenter à l'utilisateur des clusters simples, particulièrement dans des espaces de haute dimension, il est nécessaire « d'oublier » certains attributs originaux et de retrouver des sous-espaces au sens de la section 2.2. Ces sous-espaces peuvent être explicités par l'algorithme, qui identifie alors la dimensionnalité intrinsèque des sous-espaces qu'il découvre. Dans le cas de la distance euclidienne pondérée par exemple, cela revient à attribuer la valeur 0 aux (w_{rp}) des dimensions p qui ne sont pas importantes pour décrire le cluster r.

Cette section présente plusieurs approches différentes qui construisent de telles solutions parcimonieuses. Les trois premières sont des algorithmes de subspace clustering dérivés d'une fonction de coût qui enrichissent celle-ci à l'aide de fonctions *inductrices de parcimonie*, c'est-à-dire que leurs minima tendent à être des vecteurs parcimonieux.

La section 2.4.4 présente une famille d'algorithmes différents, baptisée *Sparse Subspace Clustering* (SSC), qui reconstruit les sous-espaces et les clusters en estimant un nombre *minimum* de voisins de chaque point, illustrant ainsi d'une autre façon le lien entre les approches parcimonieuses et le subspace clustering. Enfin la section 2.4.5 présente une approche parcimonieuse pour la sélection d'attributs en clustering, basée sur une distance globale plutôt que locale.

2.4.1 *L'algorithme de Borgelt (2010)*

Borgelt (2010) étudie des variantes de l'approche de Keller et Klawonn en remplaçant les deux exposants m et v par deux fonctions notées h et g proposées par Klawonn et Höppner (2003). La fonction de coût a la forme suivante :

$$F_B(C,U,W) = \sum_{i=1}^{n}\sum_{r=1}^{c} h(u_{ri}) \sum_{p=1}^{d} g(w_{rp})(x_{ip}-c_{rp})^2 \tag{2.20}$$

Elle est optimisée sous les mêmes contraintes que le problème 2.10. Étant donné un hyper-paramètre β régulant la parcimonie des solutions, la fonction g est définie comme

$$g(x) = \frac{1-\beta}{1+\beta}x^2 + \frac{2\beta}{1+\beta}x \quad \text{avec } \beta \in [0,1[\tag{2.21}$$

Ainsi, si $\beta = 0$ la fonction de coût de l'équation 2.20 est équivalente à la fonction F_K d'AWFCM avec $m = v = 2$.

La fonction F_B obtenue est différentiable, et Borgelt (2010) propose l'équation de mise à jour suivante :

$$w_{rp} = \frac{1}{1-\beta}\left(\frac{1+\beta(m_{r\oplus}-1)}{\sum\limits_{\substack{q=1\\ w_{rq}>0}}^{d} s_{rq}^{-2}} s_{rp}^{-2} - \beta\right) \tag{2.22}$$

$$\text{où} \quad m_{r\oplus} = \max\left\{k \;\middle|\; s_{r\varsigma(p)}^{-2} > \frac{\beta}{1+\beta(k-1)} \sum_{p=1}^{k} s_{r\varsigma(p)}^{-2}\right\}$$

$$\text{et} \quad s_{rp}^2 = \sum_{i=1}^{n} h(u_{ri})(x_{ip}-c_{rp})^2 \tag{2.23}$$

où l'équation 2.23 généralise le terme s_{rp}^2 de l'équation 2.13 et $\varsigma(\cdot)$ est une permutation des indices qui trie, pour r fixé, les (s_{rp}^{-2}) en ordre décroissant.

L'équation 2.22 est relativement cryptique, mais est à rapprocher d'un algorithme de projection sur le simplexe unitaire (Wang et Carreira-Perpinán 2013) : β étant fixé, $m_{r\oplus}$ correspond à l'indice au-delà duquel les poids w_{rp} solutions sont nuls. Les autres poids, non nuls ($w_{rq} > 0$), sont alors re-normalisés.

L'algorithme Prosecco que nous présentons dans le chapitre 4 p. 49 utilise un principe similaire d'exploration du simplexe unitaire Δ pour chercher les solutions W parcimonieuses.

2.4.2 *Entropy Weighting k-Means*

Jing et al. (2007) se placent dans le contexte du clustering de groupes de documents textuels, représentés par des vecteurs dont les coordonnées correspondent au nombre d'occurrences des mots d'un vocabulaire donné dans le document. Comme la plupart des mots de ce vocabulaire sont absents des documents, les données X qu'ils considèrent sont naturellement parcimonieuses. Les points ont alors peu d'attributs non

nuls en communs, ce qui complique la formation de sous-espaces et de clusters intéressants.

Jing et al. (2007) proposent de maîtriser cette parcimonie à l'aide d'un terme de régularisation entropique portant sur la matrice des poids W, afin de permettre l'identification de sous-espaces portant sur un sous-ensemble suffisamment grand de mots. Cependant, ce terme de régularisation, dont l'influence sur les minima de la fonction de coût proposée est modulée par un hyperparamètre $\gamma > 0$, promeut également des vecteurs de poids parcimonieux quand la valeur de γ tend vers 0.

L'algorithme proposé, *Entropy Weighting k-Means* (EWKM), est un algorithme de subspace clustering proposant une pondération des attributs dont le niveau de parcimonie peut être choisi par l'utilisateur via γ.

Fonction de coût et équation de mise à jour

Jing et al. (2007) proposent d'étudier la fonction de coût suivante :

$$F_E(C,U,W) = \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri} \sum_{p=1}^{d} w_{rp}(x_{ip}-c_{rp})^2 + \gamma \sum_{p=1}^{d} w_{rp} \log w_{rp}$$

$$\text{sous les contraintes} \quad \begin{aligned} &(C1)\ \forall i \in [\![1,n]\!], && \sum_{r=1}^{c} u_{ri} = 1, \\ &(C2)\ \forall i \in [\![1,n]\!], \forall r \in [\![1,c]\!], && u_{ri} \in \{0,1\}, \\ &(C3)\ \forall r \in [\![1,c]\!], && \sum_{p=1}^{d} w_{rp} = 1 \end{aligned} \tag{2.24}$$

Le premier terme de cette fonction de coût correspond à une version pondérée de la fonction de coût de l'algorithme des k-moyennes. Un terme de pénalité consistant en une somme d'entropie (Gray 2011) est ajouté à la fonction de coût pour contrôler le degré de parcimonie des solutions W : le deuxième terme équilibre les niveaux de flou et de parcimonie au sein des solutions, en fonction d'un hyperparamètre γ.

En utilisant les mêmes techniques que précédemment, Jing et al. (2007) dérivent l'équation de mise à jour suivante :

$$w_{rp} = \frac{\exp\left(\frac{-s_{rp}^2}{\gamma}\right)}{\sum_{q=1}^{d} \exp\left(\frac{-s_{rq}^2}{\gamma}\right)} \quad \text{où} \quad s_{rp}^2 = \sum_{i=1}^{n} u_{ri}(x_{ip}-c_{rp})^2 \tag{2.25}$$

En théorie, de par leur définition, l'équation 2.25 ne peut pas s'annuler, interdisant les vecteurs W_r parcimonieux. Dans la pratique cependant, c'est la précision limitée des nombres flottants qui fait que, en dessous d'une certaine valeur, certains w_{rp} très faibles s'annulent.

L'influence de γ apparaît plus clairement dans l'équation 2.25 : un γ élevé équilibre la valeur des différentes $\exp\left(\frac{-s_{rp}^2}{\gamma}\right)$, et donc des w_{rp}, égalisant l'importance des différentes dimensions et diminuant le degré de parcimonie des solutions.

Terme de régularisation entropique

Dans la suite de cette thèse, afin de comparer l'effet du terme de régularisation entropique aux autres termes étudiés, nous considérons une variante floue de l'algorithme EWKM, qui minimise le problème suivant :

$$
\begin{aligned}
F'_E(C,U,W) &= \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^m \sum_{p=1}^{d} w_{rp}(x_{ip}-c_{rp})^2 + \gamma \sum_{p=1}^{d} w_{rp}\log w_{rp} \\
\text{sous les contraintes} \quad & (C1)\ \forall i \in [\![1,n]\!],\ \sum_{r=1}^{c} u_{ri} = 1, \\
& (C2)\ \forall r \in [\![1,c]\!],\ \sum_{i=1}^{n} u_{ri} > 0, \\
& (C3)\ \forall r \in [\![1,c]\!],\ \sum_{p=1}^{d} w_{rp} = 1
\end{aligned}
\tag{2.26}
$$

où le facteur u_{ri} a été remplacé par u_{ri}^m et la contrainte $(C2)$ a été modifiée pour autoriser des solutions floues.

Notons enfin que l'utilisation de l'entropie comme d'un terme de régularisation promoteur de solutions floues a fait l'objet de différents travaux, en subspace clustering comme en clustering flou. Sadaaki et Masao (1997) l'introduisent comme alternative au facteur u_{ri}^m des c-moyennes floues. Hanmandlu et al. (2013) considèrent deux termes de régularisation entropiques, l'un pour U, l'autre pour W. Rodríguez et de Carvalho (2017) généralisent l'approche de Sadaaki et Masao (1997) en un algorithme de subspace clustering.

Tous ces algorithmes faisant cependant intervenir le même terme de pénalité pour régulariser les poids (w_{rp}), nous nous limitons dans la suite de la thèse à l'algorithme EWKM pour les comparaisons expérimentales.

2.4.3 *Enhanced Soft Subspace Clustering*

Les algorithmes de subspace clustering flou reposant sur l'optimisation d'une fonction de coût minimisent la dispersion interne de chaque cluster, donnée par la matrice suivante (Wu et al. 2005) :

$$
S = \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^m (x_i - c_r)(x_i - c_r)^\top
$$

En effet, cette matrice joue par exemple un rôle dans la fonction de coût des c-moyennes floues, qui peut s'écrire selon S :

$$
\begin{aligned}
F(C,U) &= \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^m \left\|x_i - c_r\right\|^2 \\
&= \operatorname{trace}(S)
\end{aligned}
$$

c'est-à-dire que la fonction de coût ne dépend en réalité que de la dissimilarité interne de chaque cluster, et qu'un algorithme qui minimise cette fonction pourrait ne

pas maximiser la dissimarité externe (inter-clusters). Wu et al. (2005) montrent que la quasi-totalité des algorithmes de clustering flou suivent le même principe, avant de proposer une amélioration des c-moyennes floues.

L'algorithme ESSC (*Enhanced Soft Subspace Clustering*), proposé par Deng et al. (2010), le met en œuvre et minimise la fonction de coût suivante, constituée de la variante floue de la fonction de coût de l'algorithme EWKM à laquelle un troisième terme est ajouté, modulé par un hyperparamètre $\eta > 0$:

$$\begin{aligned} F_{ES}(C,U,W) = &\sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^{m}\sum_{p=1}^{d} w_{rp}(x_{ip}-c_{rp})^2 + \gamma\sum_{r=1}^{c}\sum_{p=1}^{d} w_{rp}\log w_{rp} \\ &- \eta\sum_{r=1}^{c}\left(\sum_{i=1}^{n} u_{ri}^{m}\right)\sum_{p=1}^{d} w_{rp}(c_{rp}-\overline{c_p})^2 \end{aligned}$$

$$\begin{aligned} \text{sous les contraintes} \quad & (C1)\ \forall i \in [\![1,n]\!],\ \sum_{r=1}^{c} u_{ri} = 1, \\ & (C2)\ \forall r \in [\![1,c]\!],\ \sum_{i=1}^{n} u_{ri} > 0, \\ & (C3)\ \forall r \in [\![1,c]\!],\ \sum_{p=1}^{d} w_{rp} = 1 \end{aligned} \tag{2.27}$$

Le dernier terme correspond à une version pondérée de la matrice de dispersion externe des clusters, où $\overline{c}$ est le centre des données X. Deux paramètres de régularisation $\gamma > 0$ et $\eta \in [0,1[$ sont désormais utilisés. Deng et al. (2010) proposent les équations de mise à jour suivantes :

$$u_{ri} = \frac{d_{ri}^{\frac{2}{1-m}}}{\sum_{s=1}^{c} d_{si}^{\frac{2}{1-m}}} \quad \text{où} \quad d_{ri}^2 = \sum_{p=1}^{d} w_{rp}(x_{ip}-c_{rp})^2 - \eta\sum_{p=1}^{d} w_{rp}(c_{rp}-\overline{c_p})^2 \tag{2.28}$$

$$c_{rp} = \frac{\sum_{i=1}^{n} u_{ri}^{m}(x_{ip} - \eta\overline{c_p})}{\sum_{i=1}^{n} u_{ri}^{m}(1-\eta)} \tag{2.29}$$

$$w_{rp} = \frac{\exp\left(\frac{-s_{rp}^2}{\gamma}\right)}{\sum_{q=1}^{d}\exp\left(\frac{-s_{rq}^2}{\gamma}\right)} \quad \text{où} \quad s_{rp}^2 = \sum_{i=1}^{n} u_{ri}^{m}(x_{ip}-c_{rp})^2 - \eta\sum_{i=1}^{n}(c_{rp}-\overline{c_p})^2 \tag{2.30}$$

Dans l'équation 2.28, selon la valeur de η, la constante d_{ri}^2 peut prendre des valeurs négatives, et u_{ri} aussi. Par conséquent, Deng et al. (2010) recommandent de mettre à jour η avant d'appliquer l'équation 2.28, en utilisant l'équation suivante :

$$\eta = \min\left(\min_{r}\left\{\frac{\sum_{p=1}^{d} w_{rp}(x_{ip}-c_{rp})^2}{\sum_{p=1}^{d} w_{rp}(c_{rp}-\overline{c_p})^2}\right\}, \eta\right) \tag{2.31}$$

Pour l'équation 2.30, aucune précaution particulière n'est indiquée dans l'article. L'algorithme ESSC consiste donc en l'itération des 4 équations 2.31, 2.28, 2.29 et 2.30.

2.4.4 *Segmentation en sous-espaces : Sparse Subspace Clustering*

Elhamifar et Vidal (2009) introduisent un problème similaire au subspace clustering appelé segmentation en sous-espaces. Il consiste à décomposer la matrice des données X comme un ensemble de clusters provenant de sous-espaces vectoriels (S_r) indépendants en identifiant les dimensions et les bases de ces sous-espaces. Une fois ces sous-espaces connus, il est possible de retrouver les clusters qu'ils contiennent. Décrits dans l'espace original $\mathbb{R}^d$, les vecteurs (X_i) ne font pas apparaître la décomposition en sous-espaces souhaitée. Le but de la segmentation en sous-espaces est donc de trouver une nouvelle expression des données originales, adaptée à cette décomposition.

Elhamifar et Vidal (2009) proposent un algorithme original baptisé *Sparse Subspace Clustering* (SSC) pour résoudre le problème de la segmentation en sous-espaces, qui s'appuie sur des outils modernes d'optimisation et de traitement du signal, que nous détaillons dans la suite.

Après avoir présenté les différences entre la segmentation en sous-espaces le subspace clustering au sens de la section 2.1.3, nous présentons les hypothèses sur les clusters et les sous-espaces formulées par Elhamifar et Vidal (2009), puis le principe général de l'algorithme. Enfin, nous donnons quelques références d'algorithmes de cette même famille.

Différences avec le subspace clustering

La segmentation en sous-espaces est une tâche légèrement différente du subspace clustering présenté dans le reste du chapitre. Formulée dans un contexte algébrique, elle s'appuie sur une décomposition des données en une somme de sous-espaces vectoriels indépendants. Bien que ses résultats sur des données bruitées soient bons en pratique, une telle décomposition n'est pas toujours envisageable avec les données considérées en subspace clustering.

Les autres algorithmes de subspace clustering considérés dans ce chapitre découvrent des sous-espaces différents, sans hypothèse d'indépendance. En particulier, ils sont capables de découvrir des clusters dans l'espace original, sans réduction de dimension, quand ces clusters sont déjà suffisamment denses dans $\mathbb{R}^d$. Une telle propriété est cohérente avec le fait que le subspace clustering généralise le clustering standard, ce qui n'est pas le cadre de travail de la segmentation en sous-espaces, qui ne fait pas la différence entre les clusters et les sous-espaces qui les contiennent. Enfin, à notre connaissance il n'existe pas de comparaison entre les algorithmes des sections précédentes et les approches basées sur SSC, les deux domaines évoluant indépendamment.

La segmentation en sous-espaces apporte toutefois plusieurs idées au domaine du subspace clustering, telles que l'utilisation d'outils issus de l'analyse convexe pour contraindre les solutions ou celle de représentations parcimonieuses des données, plus adaptées à la décomposition en sous-espaces et en clusters. Des approches mixtes, combinant le clustering standard et la segmentation en sous-espaces, commencent de plus à apparaître (voir par exemple Gu et al. (2018)).

Hypothèses sur les clusters et les sous-espaces

Dans sa forme la plus simple, SSC travaille sous les hypothèses suivantes : les données $X \in \mathbb{R}^{n \times d}$ forment une matrice dont les données sont tirées d'une union de sous-espaces vectoriels indépendants (Y_r) inconnus. Soient n_r et d_r respectivement le nombre de points échantillonnés de Y_r et sa dimension. On suppose en outre que $n_r \geq d_r$ et que qu'il n'existe pas d_r points de Y_r vivant dans un espace de dimension strictement inférieure à d_r.

L'algorithme SSC introduit par Elhamifar et Vidal (2009) cherche une représentation des données en fonction d'elles-mêmes, en se basant sur l'idée suivante : exprimer chaque vecteur X_i comme une combinaison de ses voisins, et seulement en fonction d'eux, pour retrouver la décomposition en sous-espaces.

Principe de l'algorithme

Afin d'obtenir cette nouvelle représentation des données, l'algorithme SSC (Elhamifar et Vidal 2009) résout d'abord un problème d'optimisation pour chaque point X_i, qui fournit une représentation de X_i en fonction des points $(X_j)_{j \neq i}$ voisins. Cette représentation est adaptée à la décomposition en sous-espaces et SSC lui applique ensuite un algorithme de clustering spectral pour retrouver les clusters.

Elhamifar et Vidal (2009) construisent une matrice $Z \in \mathbb{R}^{n \times n}$ de coefficients à l'aide du problème suivant :

$$\underset{Z \in \mathbb{R}^{n \times n}}{\operatorname{argmin}} \|ZX - X\|_2 + \gamma \|Z\|_1 \quad \text{tel que} \quad \operatorname{diag}(Z) = 0 \tag{2.32}$$

Cette équation comporte deux termes : un problème des moindres carrés, cherchant à reconstruire X à partir d'elle-même, ainsi qu'un terme de pénalité favorisant la parcimonie de Z. Ici encore, un hyperparamètre γ est nécessaire pour équilibrer l'importance des deux termes : un γ faible diminue l'erreur de reconstruction, alors qu'un γ élevé augmente la parcimonie de Z. De plus, la contrainte interdit la solution triviale où chaque X_i est exprimé en fonction de lui-même ($Z = I$, la matrice identité).

Les (Z_i) solutions forment une nouvelle représentation parcimonieuse des données qui fait apparaître la structure de sous-espaces. En effet, Elhamifar et Vidal (2009) prouvent que, sous les hypothèses précédentes, la matrice Z vérifie :

$$\text{si } X_i \in Y_r, \text{ alors } \forall j, X_j \notin Y_r \Rightarrow z_{ij} = 0$$

c'est-à-dire que chaque point X_i n'est exprimé qu'en fonction des points qui sont situés dans le même sous-espace que lui.

D'après cette propriété, la matrice Z est diagonale par blocs à une permutation près des (X_i) ; chaque sous-espace Y_r correspond à un ou plusieurs blocs de la matrice Z. La dernière étape de SSC est d'appliquer un algorithme de clustering spectral (Shi et Malik 2000) qui identifie les clusters à partir de Z.

Une famille nombreuse

Le problème de la segmentation en sous-espaces et l'algorithme SSC de Elhamifar et Vidal (2009) s'inscrivent dans la famille des approches algébriques en vision par

ordinateur. Vidal (2010) en regroupe quelques-unes et compare SSC aux approches de type *low-rank representation* (LRR).

Ces approches cherchent une approximation de faible rang de la matrice Z précédente, c'est-à-dire qu'elles pénalisent la grandeur $\mathrm{rg}(Z)$ plutôt que $\|Z\|_1$. Par exemple, Liu et al. (2010) utilisent la norme nucléaire $\|Z\|_* = \sum \sigma_p(Z)$ comme approximation convexe, c'est-à-dire la somme des valeurs singulières de Z. Cette approche suit le même principe que SSC : exprimer les données en fonctions d'elles-mêmes, mais en s'appuyant sur une approximation de faible rang (Udell et al. 2016). Liu et al. (2010) proposent un algorithme d'optimisation alternée pour calculer la matrice Z

Lu et al. (2012) considèrent une variante du problème 2.32 qui s'appuie sur la norme de Frobenius :

$$\min_Z \|ZX - X\| + \gamma \|Z\|_F \quad \text{tel que} \quad \mathrm{diag}(Z) = 0 \tag{2.33}$$

Lu et al. (2012) remarquent que la contrainte sur la diagonale de la matrice Z peut même être ignorée, le but étant de construire une matrice de similarité diagonale par blocs. On retrouve alors un problème classique de régression d'arête (*ridge regression*, Hoerl et Kennard (1970)). La solution au problème 2.33 est classique :

$$Z^* = (X^\top X + \gamma I)^{-1} X^\top X$$

Lu et al. (2012) proposent un algorithme similaire à SSC mais basé sur le problème 2.33, *Least Squares Regression* (LSR) ; ils prouvent un résultat d'optimalité concernant leur approche, qui est exprimée en fonction de la corrélation entre les couples de points (x_i, x_j), et arguent que cette propriété de regroupement est absente de certaines méthodes basées sur la parcimonie, telles que SSC. La solution Z^* calculée est diagonale par blocs, n'est pas nécessairement de faible rang ($\mathrm{rg}(Z^*) = \mathrm{rg}(X)$), mais produit des résultats de clustering comparables, en s'appuyant sur des méthodes conceptuellement plus simples.

De nombreuses autres approches existent, basées sur la reconstruction d'une relation de similarité Z adaptée à la décomposition en sous-espaces indépendants. Chen et al. (2017) enrichissent SSC, LRR et LSR pour calculer en même temps une projection P de $\mathbb{R}^d$ vers un espace de plus faible dimension, celle-ci étant fournie par l'utilisateur en paramètre. Ils proposent un algorithme d'optimisation alternée pour construire Z et P avant d'appliquer un algorithme de clustering spectral comme les approches précédentes.

2.4.5 *Une approche de sélection d'attributs*

Nous terminons cette section par la présentation d'un algorithme de clustering avec sélection d'attributs qui utilise une pénalité non différentiable pour induire des vecteurs de poids parcimonieux. Bien que cette sélection d'attributs soit globale, les techniques employées sont appliquables au subspace clustering.

Witten et Tibshirani (2010) proposent une reformulation du problème original des k-moyennes pour faire apparaître un vecteur $W \in \mathbb{R}^d$ pondérant le rôle des dimensions

dans le coût global de la solution. Ils reformulent la fonction de coût originale des k-moyennes comme le problème de maximisation suivant :

$$\begin{aligned} & \sum_{p=1}^{d} w_p \left(\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ijp}^2 - \sum_{r=1}^{c} \frac{1}{n_r} \sum_{i,j \in C_r} d_{ijp}^2 \right) \\ \text{sous les contraintes} \quad & \|w\|^2 \leq 1, \\ & \|w\|_1 \leq s, \\ & \forall j, \qquad w_j \geq 0 \end{aligned} \tag{2.34}$$

où d_{ijp} est la distance entre X_i et X_j dans la dimension p et s est un hyper-paramètre contraignant la parcimonie des solutions, au moyen d'une contrainte inspirée de la méthode LASSO (Tibshirani 1996).

Le premier terme de la différence est constant, il s'agit de la distance entre tous les points de X en fonction de la seule dimension p. Cependant, le second terme étant négatif, le premier terme permet d'éliminer la solution triviale $W = 0$. Maximiser cette expression revient donc à minimiser le second terme, qui est une version pondérée de l'expression des k-moyennes donné par l'équation 2.2, ainsi qu'à sélectionner des poids positifs.

Qiu et al. (2015) généralise ce travail aux c-moyennes floues, sans apport du point de vue de la parcimonie.

2.5 RAPPELS D'OPTIMISATION PROXIMALE

Dans les sections précédentes, plusieurs catégories d'approches ont été distinguées : certaines se basent sur des algorithmes ad-hoc, comme CLIQUE et PROCLUS ; d'autres sont basées sur l'optimisation d'une fonction de coût continue et différentiable, minimisée sous contraintes grâce à des techniques usuelles telles que la descente de gradient ou la méthode du lagrangien. En choisissant des fonctions appropriées, certaines de ces approches parviennent à pénaliser les solutions en fonction de leur parcimonie. C'est le cas par exemple de l'algorithme de Borgelt ou de EWKM, que nous présentons dans la première moitié de la section 2.4.

Une troisième catégorie, présentée dans la section 2.4.4, fait intervenir des pénalités non différentiables. Cette nouvelle catégorie s'inspires d'approches modernes telles que l'acquisition comprimée (*compressed sensing*, Chen et al. (2001)) ou la technique LASSO (Tibshirani 1996) dans le but de proposer des estimations parcimonieuses des solutions de différents problèmes d'optimisation.

À notre connaissance, ces approches n'ont pas été étudiées par la communauté du subspace clustering flou. Il semble pourtant intéressant de chercher à étendre l'expressivité des fonctions de coût considérées en clustering flou grâce à de telles pénalités. De nouvelles propriétés deviennent envisageables, qui peuvent être introduites dans les solutions des algorithmes de clustering au moyen de ces pénalités. Cependant, cela complique la tâche d'optimisation : les approches usuelles comme celle du lagrangien ne suffisent plus et de nouvelles techniques doivent être mises en œuvre pour minimiser les fonctions de coût proposées.

Dans cette thèse, nous proposons d'explorer l'utilisation de ces nouvelles techniques en subspace clustering flou, notamment pour induire de la parcimonie dans les solu-

tions (chapitre 4) mais également pour des problématiques originales, comme la production de solutions possibilistes (chapitre 5). Dans cet optique, nous adoptons des techniques d'optimisation modernes telles que l'optimisation proximale. Dans cette section, indépendante de la tâche de subspace clustering, nous proposons un rappel des notions fondamentales en optimisation proximale, que nous mettons en œuvre pour optimiser les fonctions non différentiables que nous proposons dans les chapitres suivants.

L'une des principales applications de l'optimisation proximale est l'utilisation de normes inductrices de parcimonie (Bach et al. 2012) ; à ce titre, nous nous inspirons de cet exemple dans la suite de la thèse tout en prenant en compte les spécificités du subspace clustering flou, présentées dans le chapitre 3. Dans ce contexte, il est fréquent de partir d'un problème d'optimisation classique et d'ajouter une pénalité non différentiable pour contraindre les solutions à la parcimonie. L'algorithme d'optimisation doit alors être adapté en conséquence. Les techniques utilisées dans la section 2.3 ne s'appliquant plus, cette thèse en présente une généralisation dans le chapitre 3. Les pénalités non différentiables étudiées sont alors optimisées grâce à la technique de l'optimisation proximale (Moreau 1962 ; Rockafellar 1976), que nous rappelons ici.

2.5.1 *Cadre théorique*

La technique de l'optimisation proximale, introduite par Moreau (1962) et Rockafellar (1976), s'applique à la minimisation d'une fonction de coût qui s'écrit elle-même comme la somme de plusieurs fonctions convexes. Elle consiste à proposer un algorithme itératif qui sépare chaque étape de minimisation en plusieurs sous-étapes correspondant aux différentes fonctions. Les fonctions différentiables sont minimisées par des méthodes classiques ; celles qui ne sont pas différentiables sont minimisées grâce à leur opérateur proximal. Cette section rappelle la notion d'opérateur proximal ainsi qu'un algorithme de descente simple. On pourra consulter Parikh et Boyd (2013) pour une introduction plus détaillée.

On considère le problème suivant :

$$J(V) = F(V) + \gamma G(V) \tag{2.35}$$

dans lequel les deux fonctions F et G sont à valeur dans $\mathbb{R} \cup \{+\infty\}$. Elles sont supposées convexes, mais seule la fonction F est différentiable. La constante γ équilibre l'influence des deux fonctions dans les solutions du problème.

Descente itérative

La descente de gradient est une méthode d'optimisation classique adaptée aux fonctions de coût différentiables. Selon les caractéristiques du problème et les propriétés de la fonction de coût, de nombreuses variantes sont utilisables, présentant des performances différentes (voir par exemple Ruder (2016)).

La descente de gradient consiste à minimiser F en partant d'un point V^0 et en itérant le schéma de mise à jour suivant :

$$V^{t+1} = V^t - \frac{1}{L} \cdot \nabla F(V^t) \tag{2.36}$$

où $\frac{1}{L}$ est un pas de descente. Pour un L suffisamment grand, la suite $(F(V^t))$ décroît et, si F est minorée, cette suite est également convergente.

La fonction G n'étant pas différentiable, la descente de gradient ne peut pas être utilisée directement sur J. La descente proximale enrichit ce schéma en séparant l'optimisation de G de celle de F au moyen de l'*opérateur proximal* de G, défini par l'équation suivante, avec $\gamma > 0$:

$$\text{prox}_{\gamma G}(V) = \underset{V' \in \mathbb{R}^d}{\arg\min} \left\{ \frac{1}{2} \|V - V'\|^2 + \gamma G(V') \right\} \tag{2.37}$$

Ce problème est bien défini du fait de la convexité de la fonction G. À partir d'un point V, il consiste à chercher un nouveau point V' qui minimise G tout en restant à faible distance de V. La constante γ pondère à nouveau l'influence de G : plus γ est élevé et plus la solution de ce problème dépend de G, autorisant à chercher des minima de G plus éloignés du point courant.

L'optimisation proximale décompose une étape d'optimisation $V^t \to V^{t+1}$ en deux sous-étapes : un pas de minimisation de F grâce à la descente du gradient, puis un pas de minimisation grâce à l'opérateur proximal de G défini par le problème 2.37. Le nouveau schéma est :

$$V^{t+1} = \text{prox}_{\frac{\gamma}{L} G} \left(V^t - \frac{1}{L} \cdot \nabla F(V^t) \right) \tag{2.38}$$

La recherche du minimum de G se fait autour du point temporaire minimisant F.

La constante L influence à la fois le pas de la descente de gradient et la recherche du minimum de G. Le choix du pas $\frac{1}{L}$ est généralement crucial pour les algorithmes itératifs de descente. Un pas trop petit ralentit la convergence, alors qu'un pas trop grand peut empêcher la descente vers un minimum. Dans la pratique, il est courant d'ajouter à l'algorithme d'optimisation une étape de sélection du pas, par exemple par recherche linéaire (Beck et Teboulle 2010) et de faire varier L pendant l'exécution.

À l'instar de la descente de gradient classique, de nombreuses améliorations de ce schéma ont été proposées. Beck et Teboulle (2009) introduisent par exemple l'algorithme FISTA pour résoudre des problèmes inverses avec une vitesse de convergence quadratique. Combettes et Pesquet (2011) passent en revue quelques-uns de ces schémas et proposent de nombreux résultats permettant de calculer des opérateurs proximaux. Dans la thèse nous considérons uniquement des variantes du schéma de l'équation 2.38, qui suffit en pratique à nos besoins.

2.5.2 *Deux exemples d'opérateur proximaux*

La descente proximale fait apparaître un problème de minimisation local 2.37, qui doit être résolu à chaque itération, pour résoudre le problème de minimisation initial 2.35. L'intérêt de cette méthode réside dans l'existence de solutions efficaces, données sous forme analytique ou algorithmique, à ce problème local, pour de nombreuses fonctions de pénalité usuelles. Cependant, le calcul de l'opérateur proximal d'une fonction G donnée obéit à des règles parfois contre-intuitives. Cette section donne deux exemples d'opérateurs proximaux qui interviennent dans la suite de la thèse.

Généralisation de la projection euclidienne

La notion d'opérateur proximal peut être vue comme une généralisation de celle de projection sur un ensemble convexe. Soit $\mathcal{C}$ un convexe fermé. Sa fonction caractéristique (au sens de l'analyse convexe) $\iota : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ est donnée par :

$$\iota(V) = \begin{cases} 0 & \text{si } V \in \mathcal{C} \\ +\infty & \text{sinon} \end{cases} \tag{2.39}$$

En utilisant la définition dans le problème de minimisation 2.37, on obtient :

$$\begin{aligned} \operatorname{prox}_{\iota}(V) &= \operatorname*{argmin}_{V' \in \mathbb{R}^d} \left\{ \tfrac{1}{2}\|V - V'\|^2 + \iota(V') \right\} \\ &= \operatorname*{argmin}_{V' \in \mathcal{C}} \left\{ \tfrac{1}{2}\|V - V'\|^2 \right\} = \pi_{\mathcal{C}}(V) \end{aligned} \tag{2.40}$$

c'est-à-dire la projection euclidienne $\pi_{\mathcal{C}}(V)$ de V sur $\mathcal{C}$. Cet opérateur est généralisé dans le chapitre 4 pour produire des solutions parcimonieuses.

Opérateur proximal de la norme ℓ_1

Par définition (voir l'équation 2.37), l'opérateur proximal correspondant à la norme ℓ_1 est donné par :

$$\operatorname{prox}_{\gamma\|\cdot\|_1}(V) = \operatorname*{argmin}_{V' \in \mathbb{R}^d} \left\{ \tfrac{1}{2}\|V - V'\|^2 + \gamma\|V'\|_1 \right\} \tag{2.41}$$

Pour le calculer, on s'appuie sur la proposition suivante (Parikh et Boyd 2013, p. 129)

Proposition 2.5.1 *Soit G une fonction convexe. Si G s'écrit comme une somme séparable, c'est-à-dire $G(V) = \sum_{p=1}^{d} G_p(v_p)$, alors*

$$\operatorname{prox}_G(V) = (\operatorname{prox}_{G_1}(v_1), \operatorname{prox}_{G_2}(v_2), \ldots)$$

En l'appliquant à la fonction convexe $\|\cdot\|_1 = \sum_{p=1}^{d} |\cdot|$, le problème est ramené à la recherche d'un opérateur proximal pour la valeur absolue, c'est-à-dire à une solution au problème

$$\operatorname{prox}_{\gamma|\cdot|}(v) = \operatorname*{argmin}_{v' \in \mathbb{R}} \left\{ \tfrac{1}{2}(v - v')^2 + \gamma|v'| \right\} \tag{2.42}$$

La solution, appelée « opérateur de seuillage doux », est donnée par la fonction suivante, pour un $\gamma > 0$ fixé :

$$\operatorname{prox}_{\gamma|\cdot|}(v) = \operatorname{sign}(v) \cdot \max(|v| - \gamma, 0) \tag{2.43}$$

$$= \begin{cases} v - \gamma & \text{si } v \geq \gamma \\ 0 & \text{si } -\gamma < v < \gamma \\ v + \gamma & \text{si } v \leq -\gamma \end{cases} \tag{2.44}$$

La courbe de cette fonction est donnée en figure 2.1. L'opérateur proximal de la norme ℓ_1 est alors donné par

$$\operatorname{prox}_{\gamma\|\cdot\|_1}(V) = (\operatorname{sign}(v) \cdot \max(|v| - \gamma, 0))_p$$

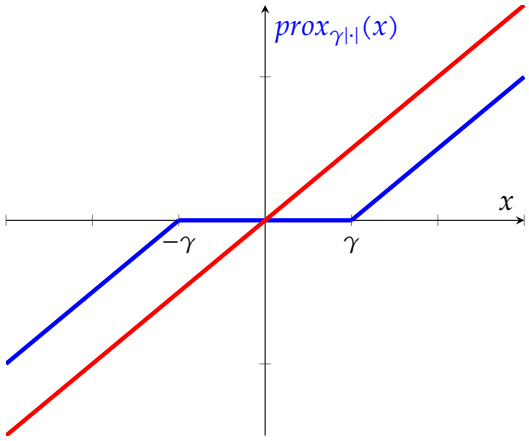


FIG. *2.1 :* Représentation de la fonction de l'équation 2.42 (en bleu)

2.6 CONCLUSION

Sous le même nom « subspace clustering » sont regroupées plusieurs familles d'approches très différentes. Leur objectif — partitionner les données en clusters homogènes tout en adaptant la relation de similarité ou distance utilisée localement à chaque cluster — est commun, mais les méthodes qu'elles emploient et les résultats qu'elles produisent sont différents.

Nous avons particulièrement détaillé les approches basées sur des fonctions de coût, dont les minima sont des solutions au problème de subspace clustering. Certaines de ces approches mettent l'accent sur des solutions vérifiant en outre des propriétés supplémentaires, comme la parcimonie, qui permet d'obtenir des solutions minimales. Dans le cadre du subspace clustering, cette minimalité apporte une autre information, à savoir une estimation de la dimensionnalité des sous-espaces.

Plusieurs de ces approches parcimonieuses reposent sur l'utilisation de pénalités non différentiables, qui font apparaître de nouvelles propriétés dans les solutions. Ces approches sont moins explorées en subspace clustering flou, notamment à cause des difficultés supplémentaires présentées par les approches usuelles de clustering flou, tels que l'optimisation sous contraintes. Par conséquent, de nouveaux schémas doivent être proposés pour s'adapter à ces contraintes.

Dans le chapitre 3 suivant, nous présentons un nouveau schéma d'optimisation basé sur la descente proximale, ainsi qu'un algorithme de subspace clustering flou qui l'intègre. Cet algorithme est générique, dans le sens où il peut être utilisé pour optimiser des fonctions de coût intégrant différentes pénalités du moment que celles-ci satisfont certaines hypothèses, notamment disposer d'un opérateur proximal.

Les trois chapitres qui suivent, chapitres 4, 5 et 6, introduisent trois propriétés différentes sous forme de fonctions de pénalité. Nous dérivons alors des opérateurs proximaux et des équations de mise à jour adaptés à ces pénalités qui permettent d'instancier l'algorithme du chapitre 3, obtenant ainsi de nouveaux algorithmes de subspace clustering flou qui satisfont les propriétés voulues.

3

UN ALGORITHME GÉNÉRIQUE DE SUBSPACE CLUSTERING

Dans le chapitre précédent, nous avons rappelé plusieurs modèles de subspace clustering qui reposent sur l'optimisation d'une fonction de coût différentiable, dérivée de celles des k-moyennes ou des c-moyennes floues. Différents modèles ont été décrits, certains introduisant des propriétés particulières dans les solutions, telle la parcimonie, en modifiant la fonction de coût originale pour ajouter des termes de pénalisation qui contraignent les solutions.

Dans cette thèse, nous nous intéressons à la possibilité d'utiliser des termes de pénalité non différentiables dans le but d'augmenter l'expressivité des fonctions de coût utilisées en clustering flou. La méthode standard d'optimisation des algorithmes précédents consiste à former le lagrangien des problèmes d'optimisation et à en dériver des équations de mise à jour ; elle ne s'applique donc qu'à des fonctions continues et différentiables. Dans ce chapitre, nous proposons un nouveau schéma d'optimisation adapté à nos besoins, basé sur l'optimisation proximale rappelée en section 2.5.

Plus précisément, ce chapitre présente une forme générale de problèmes d'optimisation, instanciée dans les chapitres suivants pour proposer de nouvelles fonctions de coût et en dériver de nouveaux algorithmes de subspace clustering flou. Ce cadre théorique, présenté en section 3.1, nous permet de considérer des problèmes d'optimisation faisant intervenir de nouveaux termes de pénalité. Ces termes de pénalité doivent cependant respecter un certain nombre de conditions, dont nous faisons la liste.

La section 3.2 présente le schéma d'optimisation générique que nous proposons pour résoudre ces problèmes d'optimisation. Nous l'implémentons dans un algorithme générique d'optimisation, PSFCM (pour *Proximal Subspace Fuzzy c-Means*), présenté en section 3.3, qui est un algorithme d'optimisation alternée faisant intervenir la descente de gradient proximale pour tenir compte des pénalités non différentiables.

L'algorithme PSFCM est issu de travaux préliminaires sur la séparation proximale pour le subspace clustering flou présentés aux conférences IPMU 2016 et LFA 2016 (Guillon et al. 2016b ; Guillon et al. 2016a) ainsi que dans le journal FSS 2018 (Guillon et al. 2018a). À notre connaissance, ces travaux sont les premiers à appliquer le cadre de l'optimisation proximale au clustering flou.

3.1 UN PROBLÈME D'OPTIMISATION GÉNÉRAL

Afin de pouvoir étudier des fonctions de coût faisant intervenir des termes de pénalité non différentiables, nous proposons un problème d'optimisation d'une forme générale, faisant intervenir un ou plusieurs termes de pénalité qui seront instanciés dans les chapitres suivants pour contraindre les solutions de différentes manières.

Ce problème d'optimisation se base sur un modèle simple de subspace clustering flou, inspiré de la fonction de coût de l'algorithme AWFCM (Keller et Klawonn (2000), voir section 2.3.2 p. 20). Trois paramètres doivent être appris : en utilisant les notations introduites dans la section 2.1.1 p. 9, en écrivant c le nombre de clusters cherchés, d la

dimension des données et n le nombre de points, ces paramètres sont les centres des clusters $C \in \mathbb{R}^{c\times d}$, les degrés d'appartenance des points aux clusters $U \in [0,1]^{c\times n}$ et les poids de chaque dimension pour chaque cluster $W \in [0,1]^{c\times d}$. Dans la suite de ce chapitre on note $\mathcal{A} = \{C, U, W\}$ l'ensemble de ces trois paramètres ; de plus, pour éviter toute confusion avec d'éventuels hyperparamètres, on désignera plus volontiers U, C et W par le terme de *variable*.

La section 3.1.1 présente la forme générale des problèmes que nous considérons. La section 3.1.2 revient sur les contraintes d'optimisation portant sur U et W, dont nous devons tenir compte pour formuler nos propres problèmes d'optimisation.

3.1.1 *Forme générale*

Afin de proposer de nouvelles fonctions de coût et d'en dériver des algorithmes de subspace clustering flou, nous nous basons sur le problème d'optimisation introduit par Keller et Klawonn (2000) pour l'algorithme AWFCM. Ce choix s'explique d'abord par la simplicité de la fonction de coût et des équations de mise à jour des paramètres de $\mathcal{A}$.

De plus, contrairement par exemple à l'algorithme de Borgelt (voir section 2.4.1 p. 27) qui produit des solutions parcimonieuses, les propriétés auxquelles nous nous intéressons dans les chapitres suivants sont absentes des solutions de l'algorithme AWFCM. Il sera par conséquent possible d'évaluer l'efficacité des pénalités que nous proposons.

Le problème d'optimisation

La forme générale des problèmes que l'on considère est la suivante. Au terme de subspace clustering de la section 2.3.2 s'ajoute une somme de termes de pénalités portant sur chaque variable $V \in \mathcal{A}$, modulés par des hyperparamètres γ_V :

$$J(C,U,W) = \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^2 \sum_{p=1}^{d} w_{rp}^2 (x_{ip} - c_{rp})^2 + \sum_{V\in\mathcal{A}} \gamma_V \cdot G_V(V)$$

$$\text{sous les contraintes} \quad \begin{aligned} &(C1)\ \forall i \in [\![1,n]\!],\ \sum_{r=1}^{c} u_{ri} = 1, \\ &(C2)\ \forall r \in [\![1,c]\!],\ \sum_{i=1}^{n} u_{ri} > 0, \\ &(C3)\ \forall p \in [\![1,d]\!],\ \sum_{r=1}^{c} w_{rp} = 1 \end{aligned} \tag{3.1}$$

La fonction de coût considérée dans ce problème de minimisation est un cas particulier de celle de l'algorithme AWFCM, dans laquelle les exposants m et v sont fixés à 2 pour simplifier l'analyse mathématique. Les contraintes originales $(C1)$, $(C2$ et $(C3)$ sont reprises à l'identique.

Des fonctions de pénalité G_V, pour l'instant non précisées, sont ajoutées. Dans le problème d'optimisation original, $\forall V \in \mathcal{A}, G_V = 0$: seul le terme principal est expri-

mé. Dans la suite, on s'autorise à ne pas écrire le nom des pénalités nulles. La fonction J a donc la forme

$$J(C, U, W) = F_K(C, U, W) + \sum_{V \in \mathcal{A}} \gamma_V \cdot G_V(V)$$

dans laquelle, en particulier, la fonction F_K est différentiable et convexe indépendamment en chacune de ses trois variables. En revanche, les pénalités G ne sont pas nécessairement convexes. En outre, F n'est pas convexe en le triplet (C, U, W).

Les pénalités G_V, précisées dans les chapitres 4, 5 et 6 selon la façon dont on souhaite contraindre les solutions du problème de minimisation, ne dépendent que de l'une des trois variables U, C, W à chaque fois et pénalisent d'un coût plus élevé les solutions qui ne présentent pas la propriété que l'on souhaite faire apparaître. Par exemple, si G est utilisée pour sélectionner des vecteurs $V \in \mathcal{A}$ parcimonieux, alors elle favorise les V parcimonieux en leur attribuant un coût plus faible.

Pénalités non différentiables et contraintes

Pour les variables sur lesquelles une pénalité non différentiable s'applique, le schéma d'optimisation que nous proposons d'utiliser n'est pas compatible avec les contraintes du problème 3.1. Par exemple si $V = U$ et qu'une pénalité non différentiable $G_U \neq 0$ est étudiée, alors $(C1)$ et $(C2)$ ne sont pas autorisées dans le problème.

Dans les chapitres suivants, nous précisons donc, pour chaque instanciation du problème 3.1, lesquelles des contraintes $(C1)$, $(C2)$ et $(C3)$ restent actives. Par ailleurs, ces contraintes, dont la principale fonction est d'interdire les solutions triviales $W = 0$ et $U = 0$, doivent être remplacées par des pénalités bien choisies.

En effet, le paradigme de subspace clustering que nous considérons a pour particularité que ces solutions triviales minimisent effectivement la fonction de coût, ce qui n'est généralement pas le cas pour les problèmes d'optimisation considérés en apprentissage automatique, comme par exemple LASSO (Tibshirani 1996). Les pénalités habituellement rencontrées en théorie de l'optimisation, dont l'un des exemples prépondérants sont les pénalités inductrices de parcimonie (Bach et al. 2012), se contentent en effet d'ajouter un coût aux solutions qui ne respectent pas certaines propriétés, ce qui n'est pas suffisant dans notre cas.

3.1.2 *De nouvelles pénalités pour le subspace clustering flou*

En instanciant les fonctions G_V du problème 3.1 par des pénalités particulières, de nouveaux problèmes d'optimisation peuvent être considérés, dont les minima diffèrent de ceux de la fonction F_K et présentent de nouvelles propriétés.

Nous faisons un certain nombre d'hypothèses sur les fonctions G considérées. Nous supposons d'abord qu'elles sont positives, à valeur dans $\mathbb{R}^+ \cup \{+\infty\}$. Le fait de pouvoir prendre des valeurs infinies est utile pour remplacer les contraintes $(C1)$, $(C2)$ et $(C3)$ précédentes ou en exprimer de nouvelles : la fonction G peut ainsi donner un coût infini aux solutions qui ne vérifient pas certaines conditions essentielles, en plus de donner un coût fini plus ou moins élevé aux solutions que l'on souhaite autoriser mais décourager.

Résoudre le problème 3.1 conduit donc à un minimum local, qui est une solution potentielle du problème de subspace clustering initial. Cependant ce minimum ne correspond ni nécessairement à un minimum de la fonction F_K initiale, ni à un minimum d'une des fonctions G : en minimisant la fonction J, l'effet de tous ces termes est combiné et le degré de cet effet peut être maîtrisé selon la valeur de la constante γ associée.

Les fonctions utilisées comme pénalités doivent cependant être bien choisies : de mauvaises pénalités pourraient conduire à des minima tout à fait inintéressants, par exemple de très mauvais partitionnements des données initiales. En outre, elles doivent également rester suffisamment simples pour pouvoir être minimisées par un algorithme dans un temps raisonnable.

3.1.3 *Non différentiabilité des pénalités*

Comme détaillé en section 2.4 p. 26, les pénalités utilisées pour introduire de la parcimonie dans les solutions ne sont généralement pas différentiables. Dans le cas général, nous ne pouvons donc pas supposer la différentiabilité de J. Nous nous privons donc de la technique habituelle qui consiste à former le lagrangien $\mathcal{L}$ du problème 3.1 et à résoudre les équations $\frac{\partial \mathcal{L}}{\partial v} = 0$ pour toutes les variables v pour en obtenir des équations de mise à jour utilisables dans un algorithme d'optimisation alternée.

Plus précisément, étant donnée une instance du problème 3.1, on note $\mathcal{D}i \subset \mathcal{A}$ les variables qui n'apparaissent que dans des termes différentiables et $\mathcal{N}i = \mathcal{A} \setminus \mathcal{D}i$ celles qui apparaissent au moins une fois dans un terme non différentiable. Les variables de $\mathcal{D}i$ restent optimisables par l'approche classique ; pour celles de $\mathcal{N}i$, nous introduisons une approche différente dans la section suivante.

3.2 OPTIMISATION PAR DESCENTE PROXIMALE

Afin de faire face aux pénalités non différentiables, cette section introduit un nouveau schéma d'optimisation pour les variantes du problème 3.1. Celui-ci sépare l'optimisation des variables des ensembles $\mathcal{D}i$ et $\mathcal{N}i$ définis précédemment en s'appuyant sur d'autres propriétés des pénalités G.

3.2.1 *Généralisation de l'optimisation alternée*

Le principe de l'optimisation alternée, mis en œuvre par les algorithmes de la section 2.3 p. 19, est de chercher le minimum de la fonction de coût en séparant la descente en chacune des variables U, C et W. Bien que cette méthode ne conduise pas forcément à un minimum global de la fonction de coût, elle donne de suffisamment bons résultats en pratique.

Le schéma d'optimisation que nous proposons généralise l'approche alternée et sépare l'optimisation de J selon les variables U, C et W en maintenant les deux autres fixées. Dans la suite, on note la fonction $J = F_K + G$ comme une fonction de la seule variable optimisée, i.e. $J(U)$, $J(C)$ ou $J(W)$ selon le contexte.

En outre, ce schéma d'optimisation distingue les variables $V \in \mathcal{D}i$ qui peuvent être optimisées à l'aide d'une simple équation de mise à jour, dérivée de façon standard, et

celles de $\mathcal{N}i$, pour lesquelles une autre approche est nécessaire. Dans cette section, V désigne un des paramètres de J qui appartient à $\mathcal{N}i$.

3.2.2 *Découpage proximal*

Si l'on fixe les paramètres autres que V, la fonction de coût s'écrit

$$J(V) = F(V) + \gamma G(V)$$

où F est convexe et différentiable en V mais pas G. Cette forme de problèmes de minimisation a rencontré un intérêt croissant ces dernières années, par exemple quand la fonction G est une pénalité inductrice de parcimonie (voir section 2.4 p. 26). En raison de la forme de la fonction J et de la différentiabilité de F, nous proposons d'utiliser le schéma d'optimisation proximale présenté en section 2.5 p. 34 comme l'une des étapes de l'optimisation alternée. Cette section détaille l'application de ce schéma au problème 3.1, notamment le calcul des gradients.

Gradients de F

La fonction F est différentiable et convexe en chacune de ses variables. Ces propriétés la rendent particulièrement adaptée à l'optimisation par descente de gradient utilisée par l'algorithme de descente proximale.

Dans le reste de la thèse, les deux paramètres concernés par des pénalités non différentiables sont les paramètres U et W. La proposition suivante donne l'expression des deux gradients de F :

Proposition 3.2.1 *Les gradients* $\nabla F(U) \in \mathbb{R}^{c\times n}$ *et* $\nabla F(W) \in \mathbb{R}^{c\times d}$ *sont :*

$$[\nabla F(U)]_{ri} = 2u_{ri} \sum_{p=1}^{d} w_{rp}^2 (x_{ip} - c_{rp})^2 \tag{3.2}$$

$$[\nabla F(W)]_{rp} = 2w_{rp} \sum_{i=1}^{n} u_{ri}^2 (x_{ip} - c_{rp})^2 \tag{3.3}$$

Proximabilité de G

Comme le résume la section 2.5 p. 34, quand G est elle aussi convexe, le découpage proximal permet de séparer l'optimisation des termes $F(V)$ et $G(V)$ en deux étapes, en s'appuyant sur la descente de gradient pour F et sur l'opérateur proximal de G, ce dernier étant défini par le sous-problème d'optimisation convexe 2.37 p. 36, rappelé ci-dessous :

$$\text{prox}_G(V) = \underset{V'}{\arg\min} \left\{ \frac{1}{2} \|V - V'\|^2 + G(V') \right\}$$

Pour que ce schéma d'optimisation soit efficace, il est nécessaire que la fonction G soit *proximable*, c'est-à-dire qu'il doit exister une façon efficace de calculer l'opérateur proximal de G.

Dans cette thèse, nous ne supposons pas que les pénalités G sont convexes. Par conséquent, leur opérateur proximal peut ne pas être correctement défini, notamment

parce qu'il peut admettre plusieurs solutions V' pour un V donné. En revanche, nous supposons quand même qu'elles sont proximables, dans le sens où il existe une procédure telle que, pour tout V, il existe au moins une solution V' à ce problème, qu'il est possible de calculer efficacement, soit analytiquement, soit algorithmiquement.

Le schéma de descente proximale pour la variable V est donné par l'équation 2.38 p. 36, c'est-à-dire

$$V^{t+1} = \text{prox}_{\frac{\gamma}{L}G}\left(V^t - \frac{1}{L} \cdot \nabla F(V^t)\right) \tag{3.4}$$

La constante $\frac{1}{L}$ est le pas de la descente de gradient (Ruder 2016). La section suivante détaille l'importance du choix de la constante L.

3.2.3 *Choix d'une constante L*

La descente de gradient proximale, présentée en section 2.5.1 p. 35, est sensible au choix du pas de descente : un pas trop petit ralentit la convergence de l'algorithme, alors qu'un pas trop grand peut empêcher d'atteindre le minimum recherché. Dans la pratique, il est courant d'ajouter à l'algorithme d'optimisation une étape de sélection du pas telle que la recherche linéaire (Beck et Teboulle 2010) et de faire varier L pendant l'exécution.

Cependant le cadre que nous proposons demande que L satisfasse une hypothèse particulière afin de faciliter la définition des pénalités G, détaillée ci-dessous. En retour, les restrictions que nous avons faites sur la fonction F dans le problème 3.1 autorisent une définition analytique de L, plus efficace qu'une procédure de recherche.

Restriction à une partie de l'espace

Comme les contraintes $(C1)$, $(C2)$ et $(C3)$ peuvent être retirées d'une instance du problème 3.1 si celui-ci contient des pénalités non différentiables sur U ou W, plus rien ne contraint en théorie les variables à rester positives et à garder des valeurs comprises entre 0 et 1. Pour garantir que les solutions calculées satisfassent cette propriété, il est nécessaire que les pénalités G considérées et la constante L utilisée lors de la descente proximale vérifient des hypothèses supplémentaires.

On suppose d'abord que les opérateurs proximaux étudiés sont définis pour des V dont toutes les composantes sont positives. On suppose également qu'ils sont eux-mêmes à valeurs positives, c'est-à-dire que les coordonnées des vecteurs calculés $\text{prox}_G(V)$ sont elles-mêmes positives.

Afin de simplifier la définition des pénalités en question, on suppose enfin que l'étape de descente du gradient elle-même ne produit jamais de coordonnées négatives. Pour un paramètre $V \in \mathbb{R}^{a\times b}$, on considère l'ensemble suivant, qui contient les solutions intermédiaires produites par la descente du gradient :

$$\text{pour tout } k \in [\![1,a]\!],\ \Delta_{\leq 1} = \left\{ V_k \in \mathbb{R}^{+b} \;\middle|\; v_{kl} \in [0,1] \wedge \sum_{l=1}^{b} v_{kl} \leq 1 \right\} \tag{3.5}$$

c'est-à-dire l'ensemble des vecteurs colonnes dont les composantes sont positives et somment à 1.

L'ensemble $\Delta_{\leq 1}$ fait le lien entre la descente du gradient ∇F, l'opérateur proximal de G utilisé lors de l'optimisation proximale de la variable V (voir l'équation 3.4) et les contraintes $(C1)$ et $(C2)$ (lorsque $V = U$) ou $(C3)$. Il peut être vu comme une forme assouplie de ces contraintes, qui apparaît naturellement dans le cadre de travail que nous avons choisi et dans lequel on souhaite restreindre les calculs. La proposition suivante fait le lien entre la descente de gradient proximale et $\Delta_{\leq 1}$:

Proposition 3.2.2 *Étant donné $\Delta_{\leq 1}$ défini comme précédemment, soit $V_k^t \in \Delta_{\leq 1}$. Il existe L tel que*

$$V_k^t - \frac{1}{L}\nabla F(V_k^t) \in \Delta_{\leq 1}$$

Démonstration *Il suffit que L vérifie*

$$\forall t,\ v_{kt} - \left[\frac{1}{L}\nabla F(V_k)\right]_{kt} \in [0,1]$$

Pour W, d'après la proposition 3.2.1, cela revient à

$$\forall r,p,w_{rp} \cdot \left(1 - \frac{2}{L}\sum_{i=1}^{n} u_{ri}^2 (x_{ip} - c_{rp})^2\right) \in [0,1]$$

Il suffit donc de prendre

$$L = \max_{r,p}\left\{2\sum_{i=1}^{n} u_{ri}^2 (x_{ip} - c_{rp})^2\right\} \tag{3.6}$$

De même pour U :

$$L = \max_{r,i}\left\{2\sum_{p=1}^{d} w_{rp}^2 (x_{ip} - c_{rp})^2\right\} \tag{3.7}$$

Ces équations permettent donc de calculer un pas de descente $\frac{1}{L}$ avant le début de la procédure d'optimisation proximale et s'assure que les vecteurs V calculés restent bien définis.

3.2.4 *Bilan : liste des hypothèses*

Nous avons énuméré plusieurs hypothèses nécessaires sur les fonctions de pénalité G non différentiables, qui permettent à la fois de définir plus facilement ces pénalités et de garantir que le schéma que nous proposons produira des résultats qui ont du sens. Ces 4 hypothèses sont :

- P1 : les pénalités G ne dépendent que d'une des trois variables de $\mathcal{A}$ à la fois ;
- P2 : les pénalités G sont positives, à valeur dans $\mathbb{R}^+ \cup \{+\infty\}$;
- P3 : les pénalités G sont proximables, dans le sens où il existe une procédure pour calculer efficacement la solution du problème 2.37 p. 36 ;
- P4 : l'opérateur proximal de G s'écrit comme un produit de fonctions qui laissent chacune $\Delta_{\leq 1}$ stable, au sens de la stabilité d'une application sur un ensemble.

Algorithme 1 PSFCM$(X, (\gamma_v), c, \varepsilon)$

1 : Initialisation : $U, C \leftarrow \text{FCM}(X, c, \varepsilon)$
2 : $\forall r, W_r \leftarrow \left[\frac{1}{d}, \cdots, \frac{1}{d}\right]$
3 : **répéter**
4 : **répéter**
5 : **pour chaque** $V \in \mathcal{D}i$:
6 : Mettre à jour V grâce à son équation de mise à jour
7 : **jusqu'à** *convergence*$(\{V \mid V \in \mathcal{D}i\}, \varepsilon)$
8 : **pour chaque** $V \in \mathcal{N}i$:
9 : Calculer L avec les équations 3.6 ou 3.7
10 : **répéter**
11 : $V_{temp} \leftarrow V - \frac{1}{L}\nabla F(V)$
12 : $V \leftarrow \text{prox}_{\frac{\gamma}{L}G}\left(V_{temp}\right)$
13 : **jusqu'à** *convergence*$(\{V\}, \varepsilon)$
14 : **jusqu'à** *convergence*$(\{U, C, W\}, \varepsilon)$

La quatrième hypothèse est à mettre en lien avec la proposition 2.5.1 p. 37 et s'explique ainsi : après optimisation, les composantes de V doivent être positives et comprises entre 0 et 1. D'après la proposition 3.2.2, on peut choisir le pas de descente de façon à ce que la descente de gradient respecte cette contrainte, il est donc nécessaire que l'opérateur proximal de G préserve cette propriété.

3.3 UN ALGORITHME DE SUBSPACE CLUSTERING GÉNÉRIQUE

Les sections précédentes ont respectivement posé le problème 3.1 et proposé un schéma d'optimisation original, en faisant la liste des différentes hypothèses nécessaires sur les fonctions de pénalité G. Nous présentons maintenant une approche générique pour minimiser le problème 3.1 : un algorithme d'optimisation alternée, combinant des équations de mise à jour directes pour les paramètres de l'ensemble $\mathcal{D}i$, et un algorithme de descente proximale pour ceux contenus dans $\mathcal{N}i$.

L'algorithme 1, baptisé *Proximal Subspace Fuzzy c-Means* (PSFCM) présente le schéma général des algorithmes étudiés dans cette thèse. En tant qu'algorithme d'optimisation alternée, il met à jour chaque variable indépendamment, les autres étant maintenues fixées. PSFCM prend en argument la matrice des données X, un vecteur d'hyperparamètres (γ_V) pour les différentes pénalités considérées, le nombre de clusters cherchés c et un réel $\varepsilon > 0$ déterminant le critère d'arrêt.

3.3.1 *Initialisation*

Les variables C et U sont initialisées par l'algorithme des c-moyennes floues (ligne 1). La variable W est initialisée comme attribuant un poids égal à chaque dimension pour chaque cluster (ligne 2).

Cette procédure d'initialisation est inspirée de l'algorithme de Borgelt (voir section 2.4.1 p. 27) et a une influence inégale selon les variables qui sont contenues dans $\mathcal{N}i$:

quand $\mathcal{N}i = \{W\}$ comme dans le chapitre 4, nous observons qu'elle a peu d'influence en comparaison de l'alternative, qui consiste à initialiser U et C comme le fait l'algorithme AWFCM. En revanche, quand $U \in \mathcal{N}i$ comme dans le chapitre 5, les résultats de l'algorithme diffèrent beaucoup selon que l'on initialise ainsi les variables ou non.

Ceci est dû au fait que l'algorithme converge vers des minima différents en utilisant les deux procédures d'initialisation. Cette sensibilité à l'initialisation est courante chez les algorithmes de partitionnement itératif. L'amélioration de cette procédure d'initialisation et l'étude de son influence sur les résultats fait partie de nos perspectives de recherche.

3.3.2 *Structure de l'algorithme*

Deux boucles imbriquées (lignes 4 à 7 et 8 à 13) sont respectivement chargées d'itérer la mise à jour des variables différentiables et non différentiables. La première est itérée jusqu'à convergence de toutes les variables de $\mathcal{D}i$ pendant que celles de $\mathcal{N}i$ sont maintenues constantes. La mise à jour des variables de $\mathcal{D}i$, ligne 6, est immédiate : si $V \in \mathcal{D}i$, minimiser $J(V)$ en maintenant les autres variables fixées est obtenu par définition des équations de mise à jour.

La seconde boucle (lignes 8 à 13) maintient constantes les variables de $\mathcal{D}i$ et met à jour les variables de $\mathcal{N}i$ en les parcourant chacune une seule fois : la descente proximale étant un algorithme approché, son exécution est sensiblement plus longue, en particulier à cause des appels aux opérateurs proximaux des pénalités concernées. Une boucle supplémentaire imposant la convergence de toutes les variables de l'ensemble $\mathcal{N}i$ risquerait d'être trop coûteuse.

Les lignes 9 à 13 décrivent la procédure de descente proximale pour une variable V. Celle-ci demande de calculer le pas de descente $\frac{1}{L}$, puis d'itérer la descente jusqu'à convergence relativement à V. L'opérateur proximal de G est évalué une fois par étape de descente ; il doit donc être connu avant l'exécution de l'algorithme et son évaluation doit être aussi efficace que possible.

3.3.3 *Critère de convergence*

L'algorithme PSFCM repose sur un critère de convergence simple, utilisé pour décider l'arrêt de l'itération sur les variables de $\mathcal{D}i$, ainsi que de la descente proximale et de l'algorithme. Ce critère est défini pour tout ensemble de variables A par

$$\textit{convergence}(A, \varepsilon) \text{ si et seulement si } \sum_{V \in A} \left\| V^{t+1} - V^{t} \right\| < \varepsilon$$

où la valeur de V correspondant à l'itération courante est comparée à la valeur de l'itération précédente. L'itération prend fin si la valeur de toutes les variables comparées ainsi se stabilise.

3.4 CONCLUSION

Ce chapitre a présenté la forme générale des problèmes d'optimisation auxquels nous nous intéressons par la suite. Ces problèmes s'appuient tous sur la fonction de

coût de l'algorithme AWFCM pour chercher des solutions au problème du subspace clustering, auquel ils ajoutent différentes pénalités pour faire émerger certaines propriétés des solutions.

Certaines de ces pénalités sont des fonctions non différentiables, qui ne peuvent donc pas être optimisées par les méthodes classiques. Nous introduisons un algorithme de mise à jour, qui s'appuie sur un découpage de la fonction de coût en deux parties : un terme F, différentiable, qui est optimisé par l'approche standard, et un terme G, non différentiable, qui est optimisé par descente proximale, sous certaines conditions.

Comme ce schéma est générique et que les fonctions G ne sont pas encore instanciées, il ne peut être validé expérimentalement. Les chapitres suivants proposent des études expérimentales des algorithmes dérivés qui valident l'efficacité de l'approche. Un travail théorique supplémentaire est cependant envisageable, qui donnerait des garanties de convergence de cet algorithme en fonction des hypothèses formulées sur la ou les fonctions G et s'inspirerait de la théorie générale des fonctions convexes en chacune de leur variable, mais pas simultanément en l'ensemble de ces variables (voir Gorski et al. (2007) pour l'exemple des fonctions biconvexes).

De même, le schéma de descente proximale que nous utilisons est simple et convient aux besoins des chapitres suivants. Une étude plus approfondie permettrait une meilleure compréhension du choix de la constante L, en particulier du point de vue de l'optimisation alternée : comment choisir une valeur de L garantissant une convergence efficace pour un pas de descente proximale sans pour autant risquer de tomber trop vite dans un minimum local de la fonction ? Une telle étude, qui fournirait un cas pratique d'étude de la théorie de l'optimisation des fonctions non convexes, pourrait également apporter des gains en performance.

4

DES SOUS-ESPACES PARCIMONIEUX

Le subspace clustering vise à établir une description adaptée des sous-espaces dans lesquels sont contenus les clusters. Afin de produire une telle description, il est nécessaire d'identifier les dimensions qui ne contribuent pas significativement à un sous-espace et de les mettre de côté, produisant ainsi une représentation parcimonieuse des clusters. Dans le formalisme que nous avons choisi, cela revient à attribuer un poids nul aux dimensions superflues.

Dans ce chapitre, nous instancions le problème d'optimisation générique du chapitre précédent : nous proposons une pénalité originale pour induire de la parcimonie dans la variable W, qui contient les poids des dimensions pour chaque cluster. La fonction correspondante est non différentiable, non continue et non convexe ; elle contraint le paramètre concerné à un ensemble de solutions admissibles et induit dans les solutions un certain niveau de parcimonie, en fonction de l'hyperparamètre associé.

Après avoir présenté la nouvelle fonction de coût et la pénalité associée dans la section 4.1, nous proposons dans la section 4.2 un opérateur proximal associé à cette pénalité, présenté sous forme algorithmique : à partir d'un W intermédiaire produit par descente de gradient, cet opérateur calcule une représentation parcimonieuse de W correspondant au niveau de parcimonie choisi par l'utilisateur.

L'algorithme générique PSFCM issu du chapitre précédent est instancié avec cet opérateur, résultant en un nouvel algorithme baptisé Prosecco, décrit dans la section 4.3. Afin de prouver la correction de l'opérateur proximal et de l'algorithme Prosecco, nous proposons un cadre géométrique original, adapté aux particularités du subspace clustering flou.

Prosecco est ensuite évalué sur des données artificielles dans la section 4.4, visant à tester sa capacité à reconnaître la dimensionnalité de sous-espaces simples. Enfin, nous proposons en section 4.5 un exemple détaillé d'utilisation de l'algorithme Prosecco sur des données réelles, qui illustre l'utilisation pratique des algorithmes de subspace clustering.

L'algorithme Prosecco a été présenté à la conférence ECDA 2018 (sans actes) ainsi que dans un article associé, soumis au journal *Archives of Data Science, Series A*.

4.1 FONCTION DE COÛT PROPOSÉE

Cette section particularise le problème de minimisation 3.1 pour introduire une pénalité sur le paramètre W. Cette pénalité, non différentiable, ajoute un coût variable en fonction de la parcimonie de chaque vecteur W_r.

4.1.1 *Forme générale*

La nouvelle fonction de coût et les contraintes actives sont présentées dans le problème 4.1, faisant intervenir un hyperparamètre $\gamma > 0$.

$$J(C,U,W) = \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^2 \sum_{p=1}^{d} w_{rp}^2 (x_{ip} - c_{rp})^2 + \gamma \sum_{p=1}^{d} G(W_r)$$

$$\text{sous les contraintes} \quad (C1)\ \forall i \in [\![1,n]\!],\ \sum_{r=1}^{c} u_{ri} = 1, \tag{4.1}$$

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

Cette fonction de coût fait apparaître une fonction de pénalité G, que nous explicitons ci-dessous, portant sur les vecteurs de poids W_r. Elle contraint indépendamment les poids de chaque cluster. L'hyperparamètre γ module l'influence de l'ensemble des pénalités G.

Alors que les contraintes $(C1)$ et $(C2)$ du problème 3.1 p. 40 restent actives, la contrainte $(C3)$ portant sur les poids W ne peut être exprimée. En effet, la fonction de pénalité que nous proposons est non différentiable et le schéma que nous proposons dans la section 3.2 p. 42 ne s'applique pas à des problèmes d'optimisation sous contrainte.

Cette contrainte servait notamment à interdire la solution triviale $W = 0$. La fonction G doit donc pénaliser les solutions de deux façons : une contrainte stricte de sommation à 1 pour remplacer $(C3)$, ainsi qu'une pénalité décourageant les solutions non parcimonieuses.

4.1.2 *Pénalisation des sous-espaces*

Dans la suite de cette section et la suivante, nous utilisons la notation V pour désigner un vecteur W_r quelconque afin d'alléger les notations, en omettant la dépendance à un cluster C_r particulier. Nous proposons la fonction G suivante, définie sur $\mathbb{R}^d$ et à valeurs positives :

$$G(V) = \begin{cases} +\infty & \text{si } \sum_{p=1}^{d} v_p \neq 1 \\ \|V\|_0 & \text{sinon} \end{cases} \tag{4.2}$$

La fonction G ne prend des valeurs finies que pour les vecteurs V respectant une contrainte de sommation, interdisant de fait les autres solutions du fait de leur coût infini. De plus, le schéma d'optimisation détaillé dans le chapitre précédent et notamment les restrictions faites sur le pas de descente, présentées dans la section 3.2.3 p. 44, contraignent les coordonnées des vecteurs considérés à rester positives.

La contrainte $(C3)$ est donc intégrée dans la pénalité G. Les V sont restreints à un ensemble particulier, le simplexe unitaire Δ, défini par

$$\Delta = \left\{ V \in \mathbb{R}^{+d} \;\middle|\; \sum_{p=1}^{d} V_p = 1 \right\} \tag{4.3}$$

Celui-ci contient les vecteurs de $\mathbb{R}^{+d}$ dont la somme des composantes vaut 1. Il est donc inclus dans l'ensemble $\Delta_{\leq 1}$ introduit dans l'équation 3.5 p. 44 et correspond au cas

où la condition (C3) est satisfaite. La section 4.2 montre comment l'opérateur proximal que nous proposons pour la pénalité de l'équation 4.2 permet de passer de $\Delta_{\leq 1}$ à Δ.

Pour distinguer les V qui respectent cette contrainte en fonction de leur degré de parcimonie, la fonction G attribue en outre un coût égal au nombre de valeurs non nulles aux vecteurs V, comptabilisées par la « norme » ℓ_0 :

$$\|V\|_0 = \sum_{\substack{p=1 \\ v_p \neq 0}}^{d} 1 \tag{4.4}$$

La pénalité que nous proposons porte donc uniquement sur la variable W et elle est à valeurs positives. Elle vérifie donc les hypothèses P1 et P2 de la section 3.2.4 p. 45. Dans la suite de ce chapitre, nous proposons son opérateur proximal, montrant ainsi qu'elle satisfait également les propriétés P3 et P4.

4.1.3 *Dérivation d'un opérateur proximal*

Afin d'utiliser l'algorithme de la section 3.3, un opérateur proximal efficace pour G doit être proposé, qui doit résoudre le problème suivant :

$$\begin{aligned} \text{prox}_{\gamma G}(V^0) &= \underset{V \in \mathbb{R}^d}{\text{argmin}} \, \tfrac{1}{2}\|V - V^0\|^2 + \gamma G(V) \\ &= \underset{V \in \Delta}{\text{argmin}} \, \tfrac{1}{2}\|V - V^0\|^2 + \gamma \|V\|_0 \end{aligned} \tag{4.5}$$

En effet, la fonction G ne prenant des valeurs finies que sur le simplexe Δ, la définition de l'opérateur proximal de G se réécrit en l'équation 4.5, suivant le même raisonnement que l'opérateur proximal de la fonction caractéristique d'un ensemble convexe, présenté en section 2.5.2 p. 36. Ce problème peut être interprété ainsi : en partant d'un V^0 donné, on cherche sur Δ le vecteur V le plus proche, dont le niveau de parcimonie est déterminé par γ.

Ce problème de minimisation est mal posé : en effet, la fonction $\|\cdot\|_0$ n'est pas convexe et, selon la valeur de γ, le problème 4.5 peut donc admettre plusieurs solutions. Cependant, toutes ces solutions potentielles présentent le même niveau de parcimonie et sont toutes d'également bonnes approximations de V^0. Par conséquent, n'importe laquelle convient et nous nous autorisons donc à raisonner comme s'il n'y en avait qu'une.

La fonction G n'est en particulier pas convexe. Cependant nous allons montrer qu'elle est proximable, c'est-à-dire que, en dépit du fait qu'il est mal posé, un opérateur proximal efficace pour G peut néanmoins être proposé. Nous généralisons la propriété de séparabilité (proposition 2.5.1 p. 37) pour obtenir l'opérateur proximal de la pénalité considérée dans le problème 4.1 à partir de celui correspondant à l'équation 4.5, étendant ses hypothèses au cas où G n'est pas convexe :

Proposition 4.1.1 *Soit $H : \mathbb{R}^{a \times b} \to \mathbb{R} \cup \{+\infty\}$ une fonction proximable. Si H s'écrit comme une somme séparable de fonctions $(H_k) : \mathbb{R}^b \to \mathbb{R} \cup \{+\infty\}$, c'est-à-dire $H(Z) = \sum_{k=1}^{a} H_k(Z_k)$ qui sont elles-mêmes proximables, alors*

$$\text{prox}_H(Z) = (\text{prox}_{H_1}(Z_1), \text{prox}_{H_2}(Z_2), \ldots)$$

Démonstration

$$\begin{aligned}\text{prox}_H(Z) &= \underset{Z' \in \mathbb{R}^{a\times b}}{argmin} \left\{ \frac{1}{2}\|Z - Z'\|^2 + H(Z') \right\} \\ &= \underset{Z' \in \mathbb{R}^{a\times b}}{argmin} \left\{ \frac{1}{2} \sum_{k=1}^{a} \|Z_k - Z'_k\|^2 + \sum_{k=1}^{a} H_k(Z'_k) \right\} \end{aligned} \tag{4.6}$$

L'équation 4.6 fait apparaître un problème sur $\mathbb{R}^{a\times b}$ dont toutes les composantes sont indépendantes et peuvent être minimisées indépendamment. Les fonctions (H_k) étant elles-mêmes supposées proximables, on définit l'opérateur proximal de H comme indiqué.

Dans le cas de la somme de pénalités du problème 4.1, les fonctions (H_k) sont toutes égales à la fonction G. La proposition 4.1.1 permet donc d'appliquer le schéma d'optimisation proposé dans le chapitre 3 à partir de l'opérateur proximal de G, que nous proposons dans la section suivante.

4.2 PROPOSITION D'UN OPÉRATEUR PROXIMAL ALGORITHMIQUE

Afin d'utiliser l'algorithme PSFCM pour résoudre le problème 4.1, il est nécessaire de dériver un opérateur proximal pour la fonction G précédente, formellement défini par l'équation 4.5. La section 4.2.1 pose les fondations du cadre théorique que nous utilisons par la suite. Nous proposons celui-ci dans la section 4.2.2 sous la forme d'un algorithme et non d'une d'expression analytique.

En conséquence, il est nécessaire de prouver que cet algorithme est correct, c'est-à-dire qu'il résout le problème 4.5. Nous nous plaçons à cet effet dans un cadre basé sur la géométrie de Δ. Nous prouvons ensuite que la solution au problème existe, puis qu'elle peut être identifiée par un algorithme d'exploration simple.

4.2.1 *Projections sur Δ*

Pour résoudre le problème 4.5, nous le reformulons comme un problème de projection sur Δ. L'ensemble Δ est fermé et convexe. Le problème 4.5 a la forme du problème 2.40 p. 37, la pénalité $\|\cdot\|_0$ en plus. Nous proposons donc un cadre de travail géométrique, basé sur les projections sur le simplexe Δ et ses faces.

Les faces du simplexe

Soit $s \subsetneq \{1, \ldots, d\}$ un ensemble d'indices, les *faces* de Δ sont les ensembles définis par :

$$\Delta_s = \left\{ V \in \Delta \;\middle|\; \forall p \in s, v_p = 0 \right\}$$

Les indices contenus dans s indiquent les composantes nulles des vecteurs V. Notons que comme $\Delta_s \subset \Delta$, la contrainte sur la somme des composantes des vecteurs $V \in \Delta$ impose que ces composantes ne peuvent pas être toutes nulles, imposant $s \neq [\![1, d]\!]$.

On note $\pi(V)$ (respectivement $\pi_s(V)$) la projection euclidienne de V sur Δ (respectivement Δ_s). L'opérateur proximal introduit dans la section suivante, donné sous la forme d'un algorithme, va consister à explorer Δ face par face pour trouver la meilleure

projection possible, celle qui résout le problème 4.5. Cela revient à chercher le minimum, sur Δ, de la fonction de coût suivante :

$$cost_{\gamma,V^0}(V) = \frac{1}{2}\|V - V^0\|^2 + \gamma\|V\|_0 \tag{4.7}$$

Les projections euclidiennes sur Δ et Δ_s n'ont pas d'expression analytique en général. Cependant, pour les points de $\Delta_{\leq 1}$, les formules suivantes peuvent être proposées (Wang et Carreira-Perpinán 2013) :

Proposition 4.2.1 *Soit $V \in \Delta_{\leq 1}$ et $s \subsetneq \{1..d\}$. Les projections $\pi(V)$ et $\pi_s(V)$ sont données par les expressions suivantes :*

$$\pi(V)_p = V_p + \frac{1}{d}\left(1 - \sum_{q=1}^{d} V_q\right) \tag{4.8}$$

$$\pi_s(V)_p = \begin{cases} 0 & \text{si } p \in s \\ V_p + \frac{1}{d-\text{card}(s)}\left(1 - \sum\limits_{\substack{q=1 \\ q\notin s}}^{d} V_q\right) & \text{sinon} \end{cases} \tag{4.9}$$

Ces formules permettent de chercher la meilleure approximation d'un V^0 sur Δ. Les projections vérifient en outre la propriété suivante :

Proposition 4.2.2 *Soient $s_1, s_2 \subsetneq \{1..d\}$ telles que $s_1 \subset s_2$ et $V \in \Delta_{\leq 1}$, alors*

$$\pi_{s_2}(\pi_{s_1}(V)) = \pi_{s_2}(V)$$

De plus, $\forall V' \in \mathbb{R}^d$:

$$\|\pi_{s_2}(V') - V'\|^2 = \|\pi_{s_2}(V') - \pi_{s_1}(V')\|^2 + \|\pi_{s_1}(V') - V'\|^2$$

Démonstration *Comme $s_1 \subset s_2$, on a $\pi_{s_2}(V) \in \Delta_{s_2} \subset \Delta_{s_1}$ et le vecteur $\overrightarrow{\pi_{s_1}(V)\pi_{s_2}(V)}$ appartient à Δ_{s_1}. Par orthogonalité de la projection, on a $\overrightarrow{\pi_{s_1}(V)V} \perp \overrightarrow{\pi_{s_1}(V)\pi_{s_2}(V)}$. Les deux résultats sont la conséquence du théorème de Pythagore.*

Existence d'un minimum

Dans la suite de cette section, on considère que V^0 et γ sont fixés et on s'autorise à noter simplement *cost* la fonction à minimiser définie dans l'équation 4.7. D'après le schéma présenté dans le chapitre 3, le vecteur V^0 est produit par descente du gradient. Les résultats de la section 3.2.3 p. 44 permettent de choisir le pas de descente $\frac{1}{L}$ de façon à contraindre V^0 à appartenir à l'ensemble $\Delta_{\leq 1}$, dont on rappelle la définition :

$$\Delta_{\leq 1} = \left\{ V \in \mathbb{R}^{+d} \;\middle|\; v_p \in [0,1] \wedge \sum_{p=1}^{d} V_p \leq 1 \right\} \tag{4.10}$$

Du fait de la non-convexité de la « norme » ℓ_0, la fonction *cost* peut avoir plusieurs minima sur Δ, pour certaines valeurs de γ et de V^0. Cependant, ces minima sont des

approximations équivalentes de V^0, et il est suffisant de savoir qu'il en existe au moins un, correspondant à la projection de V^0 sur une face Δ_s particulière. Soit S l'ensemble des projections possibles suivant :

$$S = \left\{ \pi_s(V^0) \;\middle|\; s \subsetneq \{1..d\} \right\}$$

Nous établissons la proposition suivante garantit que le minimum de *cost* sur Δ est bien dans S :

Proposition 4.2.3 *Soit $V^0 \in \Delta_{\leq 1}$, il existe s^0 tel que $\pi_{s^0}(V^0) \in S$ est un minimum de cost sur Δ, i.e. :*

$$\forall V \in \Delta,\ cost(\pi_{s^0}(V^0)) \leq cost(V)$$

Démonstration *Soit $V \in \Delta$ et $s \subsetneq \{1..d\}$ le plus grand ensemble tel que $V \in \Delta_s$, montrons que $cost(V) = cost(\pi_s(V^0)) + A$ où $A \geq 0$. Par définition de $cost(\pi_s(V^0))$,*

$$cost(\pi_s(V^0)) = \frac{1}{2}\|\pi_s(V^0) - V^0\|^2 + \gamma\|\pi_s(V^0)\|_0$$

D'autre part,

$$\begin{aligned} cost(V) &= \frac{1}{2}\|V - V^0\|^2 + \gamma\|V\|_0 \\ &= \frac{1}{2}\|V - \pi_s(V^0) + \pi_s(V^0) - V^0\|^2 + \gamma\|V\|_0 \\ &= \frac{1}{2}\|V - \pi_s(V^0)\|^2 + \frac{1}{2}\|\pi_s(V^0) - V^0\|^2 + \gamma\|V\|_0 \end{aligned}$$

par application du théorème de Pythagore, car V et $\pi_s(V^0)$ sont sur Δ_s et $\pi_s(V^0)$ est la projection orthogonale de V^0 sur Δ_s. D'autre part, comme $\pi_s(V^0) \in \Delta_s$,

$$\|\pi_s(V^0)\| \leq \mathrm{card}(s) = \|V\|_0$$

par hypothèse sur le choix de s. D'où enfin :

$$\begin{aligned} cost(V) &= \frac{1}{2}\|V - \pi_s(V^0)\|^2 + \frac{1}{2}\|\pi_s(V^0) - V^0\|^2 + \gamma\|V\|_0 \\ &\geq \frac{1}{2}\|V - \pi_s(V^0)\|^2 + cost(\pi_s(V^0)) \\ &\geq cost(\pi_s(V^0)) \end{aligned}$$

Ainsi, pour tout $V \in \Delta$, il existe s tel que $cost(\pi_s(V^0)) \leq cost(V)$, et $\pi_s(V^0) \in S$. Comme S est fini, il existe un minimum global $\pi_{s^0}(V^0)$, non nécessairement unique.

Cette proposition garantit que l'espace des solutions S peut être exploré et que le minimum peut être identifié. Une exploration naïve énumérant toutes les projections $\pi_s(V^0)$ imposerait cependant une complexité exponentielle. Nous proposons dans la section suivante un algorithme qui explore S en temps polynomial.

Algorithme 2 Opérateur proximal pour G

procédure MIN$\ell_0(V^0, \gamma)$
  $s \leftarrow \emptyset$
  $V^{sol} \leftarrow \pi(V^0)$ ▷ Projection initiale sur Δ
  $V \leftarrow V^{sol}$
  tant que $s \neq [\![1, d]\!]$:
    $p_0 \leftarrow \text{argmin}_{p \in \{1..d\}} \left\{ V^0_p \;\middle|\; p \notin s \right\}$
    $s \leftarrow s \cup \{p_0\}$
    $V \leftarrow \pi_s(V)$
    si $cost_{\gamma, V^0}(V) \leq cost_{\gamma, V^0}(V^{sol})$ **alors**
      $V^{sol} \leftarrow V$
  return V^{sol}

4.2.2 *Expression algorithmique de l'opérateur proximal*

Nous proposons l'opérateur proximal de la fonction G sous la forme de la procédure MINℓ_0, présentée dans l'algorithme 2. Cet algorithme résout le problème 4.5 ; il prend en en argument l'hyperparamètre $\gamma > 0$ et le vecteur V^0 dont on veut calculer l'approximation parcimonieuse sur Δ. L'algorithme consiste à ordonner les composantes du vecteur V^0 et à retirer progressivement les plus petites valeurs pour faire décroître la valeur de *cost*.

La proposition suivante prouve que l'algorithme MINℓ_0 minimise *cost* :

Proposition 4.2.4 *Soit $\{v^0_{p_1}, ..., v^0_{p_k}\}$ les k plus petites composantes de V^0, $s = \{p_1, ..., p_k\}$ et $V \in \Delta_s$ tel que V contient exactement $k = \text{card}(s)$ zéros. Soit V' tel que $v'_p = v_p$ pour tout p sauf pour un couple (q, q') avec $q \in s, q' \notin s$, et $v_q = v'_{q'} = 0$, $v_{q'} = v'_q$. Alors $cost(V) \leq cost(V')$.*

Démonstration *Soient deux tels V et V' : V et V' ont les mêmes composantes, sauf pour q et q' qui sont inversées. Par hypothèse on a $v^0_q \leq v^0_{q'}$ et V et V' ont le même nombre de composantes nulles. Alors :*

$$\begin{aligned} cost(V) - cost(V') &= \sum_{p=1}^{d} (v_p - v^0_p)^2 - \sum_{p=1}^{d} (v'_p - v^0_p)^2 \\ &= (v_{q'} - v^0_{q'})^2 + (v^0_q)^2 - (v'_q - v^0_q)^2 - (v^0_{q'})^2 \\ &= 2v'_q(v^0_q - v^0_{q'}) \leq 0 \end{aligned}$$

Dans la proposition 4.2.4, nous montrons que, lors de la construction de la solution parcimonieuse V, rajouter un 0 doit être nécessairement fait en remplaçant la plus petite composante non nulle de la solution courante ; V peut donc être construit composante par composante, de façon itérative. En effet, à chaque étape, étant donné deux s et s' qui diffèrent uniquement par deux indices q et q', celui qui conduit au minimum est celui qui correspond à la composante de V^0 la plus petite dont l'indice n'est pas encore dans s. Comme l'algorithme 2 conserve le minimum à chaque itération, il est certain de trouver le minimum de *cost*.

Algorithme 3 L'algorithme Prosecco

1 : **procédure** Prosecco($X, c, \gamma, \varepsilon$)
2 : Initialisation : $U, C \leftarrow \text{FCM}(X, c, \varepsilon)$
3 : $\forall r, W_r \leftarrow \left[\frac{1}{d}, \cdots, \frac{1}{d}\right]$
4 : **répéter**
5 : **répéter**
6 : Mettre à jour U d'après l'équation 2.12 p. 21 avec $m = v = 2$
7 : Mettre à jour C d'après l'équation 2.11 p. 21 avec $m = 2$
8 : **jusqu'à** *convergence*(C, U, ϵ)
9 : Calculer L d'après l'équation 3.6 p. 45
10 : **répéter**
11 : $W_{\text{temp}} \leftarrow W - \frac{1}{L} \cdot \nabla F(W)$ d'après l'équation 3.3 p. 43
12 : $W \leftarrow \text{Min}\ell_0(W_{\text{temp}}, \frac{\gamma}{L})$
13 : **jusqu'à** *convergence*(W, ε)
14 : **jusqu'à** *convergence*(C, U, W, ε)

Théorème 4.2.1 *Soient V^0 et γ fixés. L'algorithme 2 Minℓ_0 calcule la solution du problème 4.5.*

Démonstration *L'algorithme 2 minimise cost(V) à chaque itération t, avec t zéros fixés : la proposition 4.2.3 garantit que le minimum peut être atteint par projections successives et la proposition 4.2.4 prouve que l'on peut le construire itérativement. Enfin, si l'on trouve un meilleur candidat pour le minimum, la ligne 10 le mémorise et il est renvoyé à la ligne 13.*

L'algorithme Minℓ_0 est de complexité polynomiale : il consiste à trier la liste des composantes de V^0 et réalise d itérations pour construire la meilleure approximation parcimonieuse de V^0. À chaque itération, la composante la plus faible est annulée, le vecteur V est projeté sur la face correspondante et son coût est comparé à celui de la solution. Toutes ces opérations ont un coût linéaire, la complexité de Minℓ_0 est donc quadratique ; quand d est élevé, cette procédure est donc trop coûteuse et des travaux futurs consisteront à en proposer une version plus efficace.

Enfin, la procédure Minℓ_0 fournit un opérateur proximal pour la pénalité du problème 4.1 et celui-ci associe bien un point de Δ, donc de $\Delta_{\leq 1}$, à tout point $V^0 \in \Delta_{\leq 1}$. Nous vérifions donc bien les propriétés P3 et P4 de la section 3.2.4 p. 45.

4.3 PROSECCO, UN ALGORITHME PARCIMONIEUX

Nous avons établi dans la section précédente un opérateur proximal, sous forme algorithmique, de complexité polynomiale, qui vérifie les hypothèses du chapitre précédent : à partir d'un vecteur $V^0 \in \Delta_{\leq 1}$, la procédure Minℓ_0 produit un vecteur dont les composantes sont positives et dont la somme vaut 1 ; le résultat Min$\ell_0(V^0)$ appartient donc bien à Δ. L'algorithme PSFCM 1 appliqué à la résolution du problème 4.1 fournit un nouvel algorithme de subspace clustering flou.

Baptisé Prosecco, cet algorithme est présenté en algorithme 3. Comme dans la section 3.3, il s'agit d'un algorithme d'optimisation alternée : les trois variables sont opti-

misées séparément. De plus, l'optimisation « exacte » des variables U et C est séparée de celle « inexacte » de la variable W, par descente proximale.

Prosecco est initialisé grâce à l'algorithme des c-moyennes floues. Dans la pratique, nous n'observons pas de différence avec ou sans cette initialisation, la première boucle remplissant déjà ce rôle. Le paramètre γ doit être choisi en fonction du degré de parcimonie attendu dans les solutions. Nous observons expérimentalement que les résultats de l'algorithme sont stables quand γ varie peu. Il peut néanmoins être nécessaire de relancer l'algorithme avec différentes valeurs pour régler le niveau de parcimonie, par exemple avec une valeur initiale de 1 que l'on multiplie ou divise par 10 à chaque itération.

Prosecco se distingue des algorithmes parcimonieux de l'état de l'art, présentés en section 2.4 p. 26, en s'appuyant sur un opérateur qui explore une structure discrète, celle des faces de Δ, plutôt que sur une équation de mise à jour. Cette approche est prouvée correcte grâce aux théorèmes de la section précédente.

4.4 VALIDATION EXPÉRIMENTALE : ESTIMATION DE LA DIMENSIONNALITÉ

Afin d'évaluer la pertinence de l'algorithme Prosecco en tant qu'algorithme de subspace clustering flou, nous l'étudions expérimentalement sur différents jeux de données. L'évaluation d'algorithmes pour des tâches non supervisées est un problème épineux, en particulier pour le subspace clustering. En effet, un algorithme d'optimisation peut parfaitement produire des solutions qui sont de très bons minima de la fonction de coût mais qui ne correspondent pas aux groupes attendus par l'utilisateur, qui eux-mêmes pourraient ne pas correspondre aux minima de la fonction.

Pour pallier ce problème, nous comparons d'abord les différents algorithmes de subspace clustering de l'état de l'art et Prosecco sur des données artificielles. Celles-ci correspondent à des nuages de points générés par des distributions simples, présentant des caractéristiques idéales, qui permettent d'illustrer les comportements des différents algorithmes.

La section 4.4.1 considère des données en dimension 3 permettant de visualiser les résultats obtenus par Prosecco et les algorithmes des c-moyennes floues et AWFCM, respectivement décrits dans les section 2.1.2 p. 13 et section 2.3.2 p. 20. Cette expérience est ensuite généralisée à des données de dimension supérieure selon un protocole présenté en section 4.4.2. Les résultats de l'expérience sont présentés et commentés en section 4.4.3.

4.4.1 *Un exemple visuel*

La figure 4.1 propose un exemple de jeu de données artificielles en dimension 3, composées de deux groupes de points : le premier est généré selon une loi uniforme mais avec une très faible épaisseur ($|z| < 0{,}2$). Le second est généré selon une loi normale et forme un ellipsoïde plein. L'intersection des deux groupes est non vide, créant une ambiguïté dans les données.

Cet exemple illustre plusieurs caractéristiques du subspace clustering flou : premièrement, le plan est situé dans un espace à deux dimensions, alors que l'ellipsoïde existe en trois dimensions. En outre, les points situés à l'intersection des deux volumes ap-

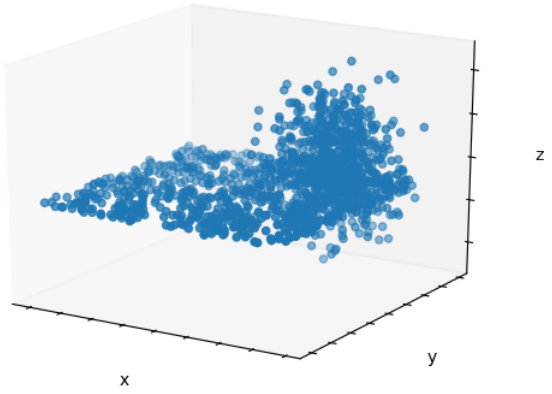
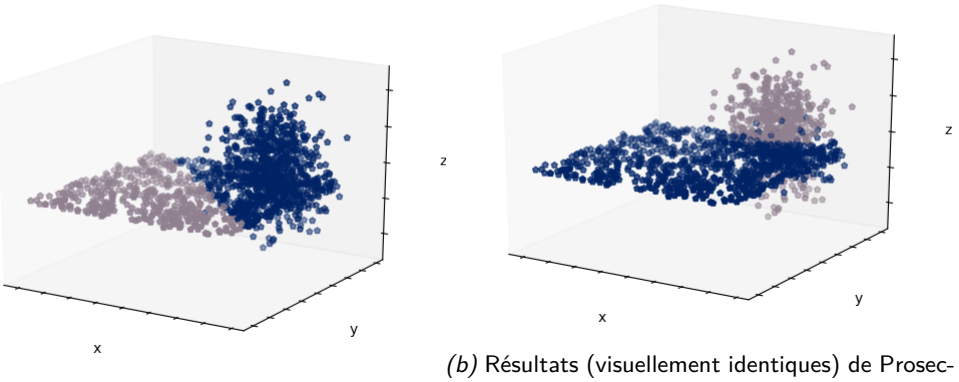


FIG. *4.1 :* Données artificielles tridimensionnelles composées d'un plan et d'un ellipsoïde



(a) Résultat de FCM

(b) Résultats (visuellement identiques) de Prosecco et AWFCM

FIG. *4.2 :* Résultats des algorithmes des c-moyennes floues (FCM), AWFCM et Prosecco.

partiennent aux deux clusters simultanément ce qui justifie l'utilisation du paradigme flou.

La figure 4.2 représente les résultats des algorithmes FCM, AWFCM et Prosecco sur ces données, avec les paramètres $m = 2$ pour FCM et AWFCM, $v = 2$ pour AWFCM et $\gamma = 1$ pour Prosecco, ainsi que $c = 2$, correspondant au nombre de clusters cherchés. Les points x_i originaux sont colorés selon le cluster C_r maximisant u_{ri}.

L'algorithme FCM n'apprend pas les sous-espaces dans lesquels se trouvent les clusters. Il identifie des clusters sphériques, comme le montre la figure 4.2a : l'ellipsoïde et le plan ne sont pas reconnus correctement, certains points du second étant affectés au premier. Sur la figure 4.2b, qui représente le résultat visuellement identiques des algorithmes Prosecco et AWFCM, le plan et l'ellipsoïde sont davantage délimités. Certains points générés selon la loi normale sont majoritairement affectés au plan et sont colorés en bleu ; situés à l'intersection des deux nuages de points, ces données sont ambiguës et sont partiellement affectées aux deux clusters. Sur cet exemple, les deux algorithmes produisent des affectations très proches, ne se distinguant que par les vecteurs de poids calculés.

	CLUSTER	AXE x	AXE y	AXE z
Prosecco	plan	0	0	1
	ellipsoïde	$3{,}41 \cdot 10^{-1}$	$3{,}71 \cdot 10^{-1}$	$2{,}88 \cdot 10^{-1}$
AWFCM	plan	$5{,}90 \cdot 10^{-3}$	$5{,}83 \cdot 10^{-3}$	$9{,}88 \cdot 10^{-1}$
	ellipsoïde	$3{,}41 \cdot 10^{-1}$	$3{,}72 \cdot 10^{-1}$	$2{,}87 \cdot 10^{-1}$

TAB. *4.1 :* Poids calculés par les algorithmes Prosecco et AWFCM sur les données de la figure 4.1.

Le tableau 4.1 donne les poids calculés par les deux algorithmes AWFCM et Prosecco. Les résultats montrent que l'algorithme Prosecco détermine une seule dimension significative pour le cluster bleu, qui est donc identifié comme étant un plan ; les poids attribués aux deux autres dimensions sont nuls. Pour le même plan, l'algorithme AWFCM produit un vecteur de poids non parcimonieux, dans lequel se trouvent des valeurs très faibles mais non nulles, représentant des informations superflues.

En revanche, Prosecco détermine des poids significatifs pour toutes les dimensions de l'ellipsoïde, avec un poids plus faible pour la dimension verticale, les points de l'ellipsoïde étant plus étalés dans cette dimension. Pour la valeur de γ utilisée ici, Prosecco calcule donc une représentation parcimonieuse du sous-espace du premier cluster, mais pas du second, qui correspondrait à une approximation trop coûteuse. Augmenter γ conduirait, au-delà d'une certaine valeur, à encourager également une représentation plus parcimonieuse pour le vecteur de poids de l'ellipsoïde.

4.4.2 *Protocole expérimental*

Nous comparons Prosecco à deux autres algorithmes de subspace clustering flou induisant de la parcimonie : l'algorithme de Borgelt (p. 27) et EWKM (p. 27). L'algorithme AWFCM n'induisant pas de parcimonie, il n'est pas inclus dans la comparaison. Le but de l'expérience est d'évaluer la capacité des algorithmes à retrouver des clusters simples, générés dans des sous-espaces de taille variable, et à évaluer correctement leur dimensionalité.

Génération des données

Les algorithmes sont comparés sur les données suivantes : des hyperplans de $\mathbb{R}^d$ de dimension variable, générés selon une loi uniforme. Plus précisément, pour d fixé, k clusters de dimensionalité aléatoire $d_r \in [\![1, d-4]\!]$ sont générés, de $n_r = 600$ points chacun. Lorsque d augmente, la dimensionnalité générée pour les clusters peut donc être plutôt importante.

Dans chacune des d_r dimensions choisies, une coordonnée centrale c_p est générée entre -10 et 10. Les données sont générées selon une loi uniforme avec des coordonnées comprises entre $c_p - 0{,}2$ et $c_p + 0{,}2$. Les coordonnées dans les $d - d_r$ dimensions restantes sont générées selon une loi uniforme dans l'intervalle $[-10, 10]$.

Ces expériences sont reproduites pour d variant de 2 en 2 entre 20 et 60, et $k \in \{2, 4, 6\}$.

Évaluation des algorithmes

Bien que l'algorithme EWKM n'induise pas en théorie de parcimonie (voir équation 2.25 p. 28), les valeurs des w_{rp} en-dessous de 10^{-10} sont considérées comme nulles après exécution pour le bien de cette expérience. Pour l'algorithme de Borgelt et Prosecco, les w_{rp} non nuls sont utilisés pour déterminer les dimensions considérées comme significatives par les algorithmes.

La capacité des algorithmes à estimer la dimensionalité des hyperplans est évaluée ainsi : étant donné un cluster $\hat{C}_r$ généré avec une dimensionnalité d_r, chaque algorithme doit affecter correctement 80% des points (après défuzzification de U) de chaque cluster, puis calculer W_r tel que $\|W_r\|_0 = d_r$ et que les w_{rp} non nuls correspondent aux dimensions significatives du cluster généré. La métrique utilisée pour la comparaison des algorithmes est le ratio ρ des estimations correctes.

Formellement, pour une expérience fixée avec k clusters générés, soient $D_1, ..., D_k$ les ensembles des dimensions significatives choisies, de sorte que $\text{card}(D_r) = d_r$. Pour un cluster $_r$ généré, un point x_i est affecté à $\hat{C}_r$ si $u_{ri} > 0{,}5$; un cluster est correctement identifié si 80% de ses points x_i lui sont affectés. Le vecteur produit W_r est alors comparé à D_r : seules doivent être nulles les dimensions $p \in D_r$.

En notant $I(\cdot)$ la fonction qui reçoit une condition en argument et vaut 1 si celle-ci est vraie, 0 sinon, on forme le ratio suivant :

$$\rho = \frac{1}{k} \sum_{r=1}^{k} I\left(\sum_{x_i \in C_r} I\left(u_{ri} > 0{,}5\right) > 0{,}8 \right) \cdot I(W_r = D_r) \tag{4.11}$$

La métrique utilisée pour comparer les algorithmes est le ratio ρ moyen pour 100 expériences.

Paramètres des algorithmes

Les algorithmes comparés requièrent tous les trois un hyperparamètre γ (pour Prosecco et EWKM) ou β (pour l'algorithme de Borgelt) qui définit l'influence du terme inducteur de parcimonie. Comme d'une part aucun de ces trois algorithmes ne propose de procédure pour choisir cet hyperparamètre, et que d'autre part la valeur idéale de celui-ci dépend des données considérées, pour chaque données X générées, l'algorithme de Borgelt et EWKM sont exécutés avec différentes valeurs de β et γ (respectivement). Le paramètre présentant les meilleurs résultats en moyenne sur l'ensemble des expériences est conservé.

Pour l'algorithme de Borgelt, on teste les paramètres $\beta = 10^{-i}$ pour $i \in \{3, 2, 1\}$. Pour EWKM, on teste $\gamma \in \{10^{-2}, 10^{-1}, 0.5, 10\}$. Ces deux plages de valeurs sont arbitraires et correspondent à l'intuition que nous nous sommes faite de ces différents algorithmes. Elles , ils sont cohérentes avec les valeurs mentionnées par les auteurs des algorithmes dans leurs articles. Prosecco est exécuté avec une seule valeur, $\gamma = 1$, ses résultats étant satisfaisants pour cette valeur. Enfin, pour chaque algorithme, $c = k$ est considéré comme connu et $\varepsilon = 10^{-4}$.

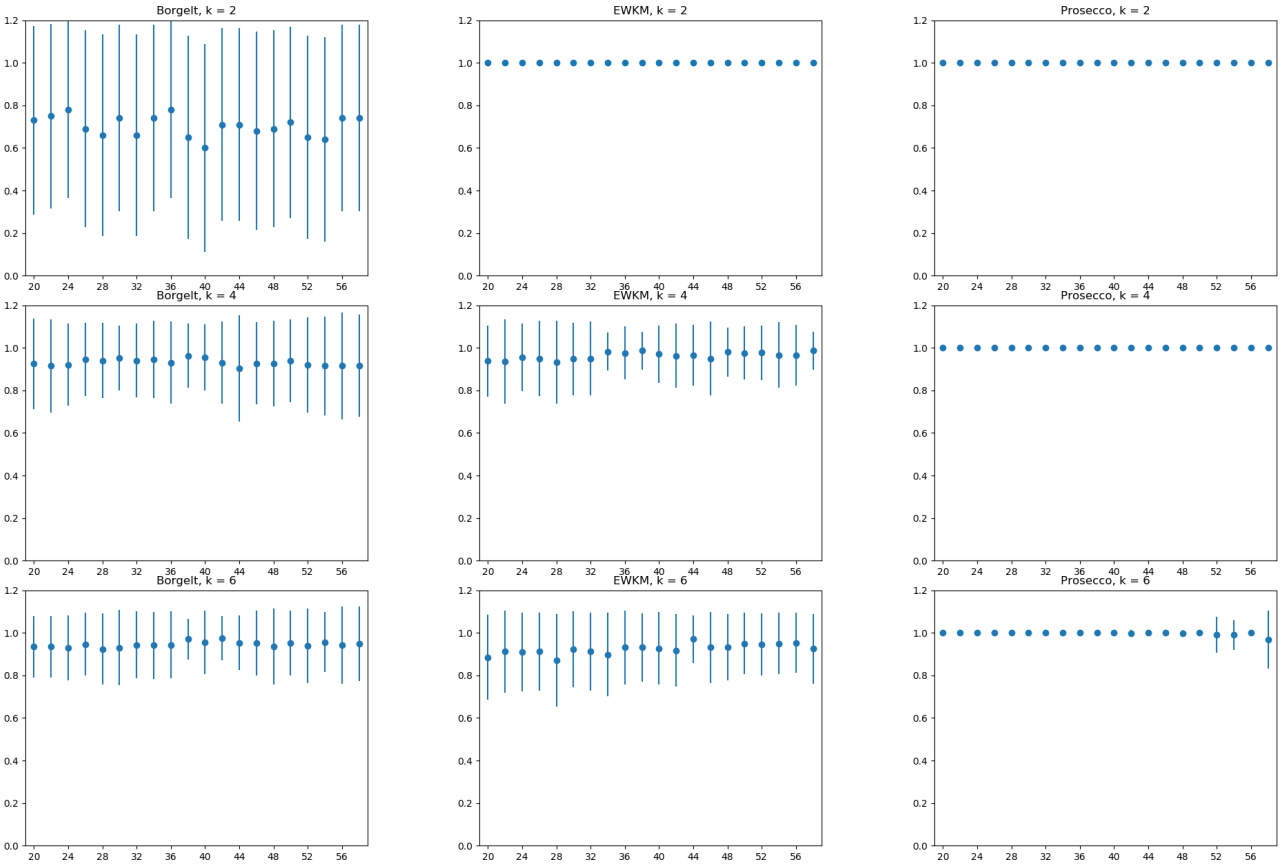


FIG. *4.3 :* Résultats de l'expérience sur des données artificielles pour les algorithmes de Borgelt, EWKM et Prosecco. Chaque ligne correspond à une valeur de $k \in \{2, 4, 6\}$. En abscisse de chaque image, la dimension globale d. En ordonnée, le ratio moyen de clusters dont la dimensionnalité a été correctement identifiée.

4.4.3 *Discussion des résultats expérimentaux*

Le score moyen des trois algorithmes, calculés sur 100 exécutions, est reporté dans la figure 4.3 pour les paramètres $k \in \{2, 4, 6\}$ et $d \in [\![20, 58]\!]$. Les hyperparamètres ayant conduit aux meilleurs résultats sur l'ensemble des données sont respectivement $\beta = 0{,}01$ pour l'algorithme de Borgelt et $\gamma = 0{,}5$ pour EWKM.

La figure présente les résultats comme une grille, dont les lignes correspondent à un k fixé et les colonnes à un algorithme. Chaque image montre l'évolution de la moyenne et de l'écart-type des ratios en fonction de la dimension d. L'algorithme de Borgelt obtient des résultats très variables pour $k = 2$ et semble s'améliorer lorsque k augmente, comme le montrent l'augmentation du ratio moyen et la diminution de l'écart-type lorsque k augmente. Ce résultat est inversé pour EWKM, qui conserve cependant un ratio moyen supérieur à celui de l'algorithme de Borgelt pour $k = 2$ et $k = 4$.

Prosecco a un ratio moyen de 1 pour $k = 2$ et $k = 4$, ce qui signifie qu'il identifie correctement tous les plans et leur dimension dans ces deux expériences. Pour $k = 6$,

CATÉGORIE DE PRODUIT	MOYENNE	ÉCART-TYPE
Produits frais	12000	12647
Produits laitiers	5796	7380
Épicerie	7951	9503
Produits congelés	3072	4855
Produits d'entretien	2881	4768
Épicerie fine	1525	2820

TAB. *4.2 :* Description des variables du jeu de données *"Wholesale"*.

une légère instabilité apparaît en haute dimension ($d > 52$), mais le ratio moyen reste très proche de 1.

Cette expérience tend à montrer que Prosecco estime plus fidèlement la dimensionnalité des hyperplans générés. Cependant, elle montre également la difficulté de choisir l'hyperparamètre de ces algorithmes, γ ou β. Même dans un contexte non supervisé, les résultats d'un algorithme sont fortement influencés par la valeur de cet hyperparamètre, qui lui-même produit des effets différents selon les données considérées, leur nombre n et, dans une moindre mesure d'après la figure 4.3, leur dimension d.

Cette caractéristique est à la fois une faiblesse pratique bien connue des algorithmes de fouille de données paramétrés de la sorte, mais également une difficulté de l'évaluation de ces algorithmes. Des extensions permettant de déterminer de bonnes valeurs de γ ou β permettraient à la fois de résoudre ce problème pratique et d'améliorer cette expérience.

4.5 UN EXEMPLE D'UTILISATION DE L'ALGORITHME PROSECCO

Les articles de clustering illustrent le plus souvent les performances des algorithmes proposés à l'aide de *benchmarks*, menant à une évaluation supervisée. Bien que ce mode opératoire permette de mettre en avant certaines des forces et des faiblesses des algorithmes présentés, il est très éloigné de la *pratique* du clustering, qui consiste à appliquer un algorithme sur des données sur lesquelles on ne dispose que de peu d'informations.

Cette section présente un cas concret d'utilisation de l'algorithme Prosecco sur des données réelles. Plutôt que de s'appuyer sur des critères de qualité, le résultat de Prosecco est analysé et comparé aux connaissances dont on dispose sur les données, ainsi qu'aux partitions formées par les algorithmes des k-moyennes et AWFCM. Cet exemple correspond à notre sens à la pratique du clustering, incluant par exemple la recherche du nombre de clusters et la visualisation des résultats.

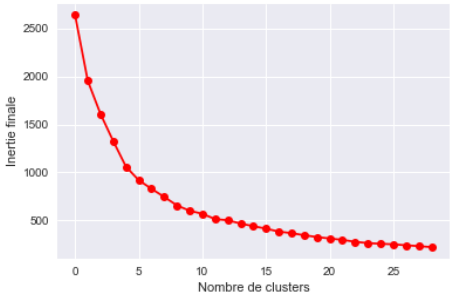


FIG. *4.4 :* Évolution de la valeur finale de la fonction de coût de l'algorithme des k-moyennes sur les données du tableau 4.2 lorsque k augmente.

4.5.1 *Protocole expérimental*

Les données utilisées dans cette section correspondent à des données de vente « en gros », dans une unité monétaire donnée, de $d = 6$ produits différents, à $n = 440$ clients situés au Portugal. Deux catégories sont de plus proposées pour ces clients : une variable `Channel` qui distingue deux catégories de clients (hôtels/restaurants/cafés d'un côté et revendeurs au détail de l'autre) et une autre variable `Region` qui désigne la ville du client : Lisbonne (77 clients), Porto (47 clients) ou autre (316). Ces données proviennent d'une thèse en marketing et sont distribuées par le site UCI sous le nom *"Wholesale Dataset"* (Lichman 2013).

Standardisation des données

Les 6 catégories de produits ainsi que la moyenne et l'écart-type des variables correspondantes sont présentées dans le tableau 4.2. Les écarts-types élevés correspondent à des valeurs extrêmes dans les données pour chaque variable. Comme les algorithmes que nous utilisons sont sensibles aux valeurs extrêmes de ce genre, nous choisissons de standardiser les données, de telle sorte que chaque variable est centrée en 0 et a un écart-type égal à 1.

Ce choix peut paraître contradictoire avec le fait de travailler avec des algorithmes de subspace clustering, qui pondèrent eux-mêmes l'importance des variables. Cependant les algorithmes AWFCM et Prosecco sont tous les deux initialisés avec des vecteurs de poids tous égaux : toutes les variables sont donc initialement importantes et des valeurs extrêmes, même pour certaines valeurs seulement, peuvent faire converger les algorithmes vers des minima locaux moins intéressants. De plus, le chapitre suivant montre que ces algorithmes restent sensibles au bruit. Standardiser les variables nous paraît donc pertinent même pour des algorithmes de subspace clustering.

Choix du nombre de clusters

Les trois algorithmes que nous utilisons dépendent du nombre de clusters cherchés, k ou c. Plusieurs méthodes plus ou moins empiriques peuvent être utilisées pour déterminer, a priori, un nombre de clusters pertinents. Parmi celles-ci, la « méthode du coude », couramment utilisée conjointement à l'algorithme des k-moyennes, consiste

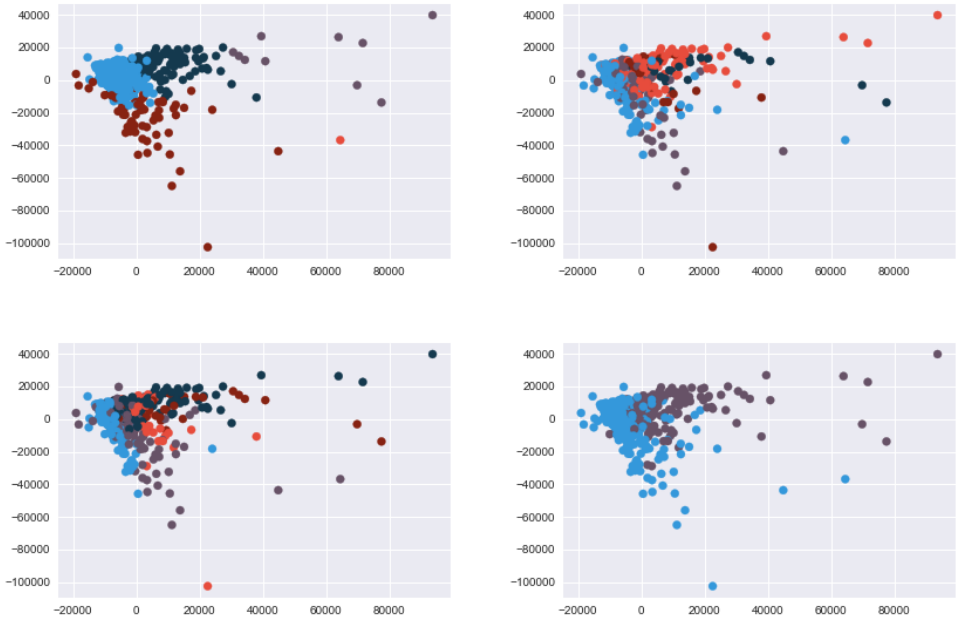


FIG. *4.5 :* Visualisation du jeu de données *"Wholesale"* : en haut et en bas à gauche, résultats des algorithmes des k-moyennes, AWFCM et Prosecco sur les données *"Wholesale"*, pour $k = c = 5$. En bas à droite, visualisation de la catégorie correspondant à la variable `Channel`.

à relancer l'algorithme en augmentant le paramètre k et en traçant l'évolution de la valeur finale de la fonction de coût en fonction de k. Celle-ci décroît au fur et à mesure que l'on augmente le nombre de clusters ; on retient alors le dernier k qui produit un changement de valeur finale significatif.

Cette méthode peut également être employée pour l'algorithme AWFCM. Pour Prosecco en revanche, la dépendance à un autre hyperparamètre, γ, dont le choix est partiellement influencé par le nombre de clusters c, décourage l'utilisation de cette méthode. Afin de faciliter les comparaisons avec les deux autres algorithmes, nous choisissons donc le même paramètre c.

La figure 4.4 présente l'évolution de la valeur finale de la fonction de coût pour l'algorithme des k-moyennes, qui fait apparaître une inflexion autour de $k = 5$; en utilisant l'algorithme AWFCM on observe une courbe similaire. Nous choisissons donc $c = 5$ pour les algorithmes AWFCM et Prosecco. En outre, nous observons que le choix $\gamma = 1$ produit des résultats satisfaisants, avec des vecteurs de poids parcimonieux. La suite de cette section présente donc les résultats des trois algorithmes pour ce choix de paramètres.

4.5.2 *Visualisation des résultats*

Il est courant en fouille des données de visualiser les clusters pour se faire une idée de leur organisation. La dimension des données que nous étudions empêche une visualisation directe ; cependant, des techniques comme le *MultiDimensional Scaling* (MDS, voir

Cox et Cox (2000)) permettent de se faire une première idée. Ces techniques consistent à projeter les données en respectant au mieux les distances deux à deux ; combinées à une coloration des points en fonction du cluster auquel elles sont majoritairement affectées, elles permettent de se faire une première idée des clusters identifiés.

La figure 4.5 présente une telle visualisation : la visualisation du résultat de l'algorithme des k-moyennes est située en haut à gauche, celle d'AWFCM en haut à droite et celle de Prosecco sur la deuxième ligne de la figure. Cette figure permet de constater quelques différences entre les 3 algorithmes ; les clusters produit par les k-moyennes sont plus « sphériques » que les autres dans le sens où ils forment des blocs uniformes, sans mélange. De plus, les k-moyennes produisent 3 clusters majeurs, représentés en bleu foncé, bleu clair et marron, ainsi que deux clusters mineurs, en mauve et orange, ce dernier ne comptant d'ailleurs qu'un seul point, qui correspond à un extrême au sein des données.

Les algorithmes Prosecco et AWFCM semblent produire des clusters plus équilibrés, répartis différemment. Les clusters produits regroupent des points extrémaux (tel que le point isolé par l'algorithme des k-moyennes) et des points plus « normaux ». La visualisation proposée par MDS étant construite de façon à préserver les distances euclidiennes originales, il est normal qu'elle soit plus confuse pour ces deux algorithmes de subspace clustering. Borgelt (2010) propose également de visualiser les clusters dans les sous-espaces sélectionnés par les algorithmes parcimonieux ; cependant, comme ces sous-espaces sont propres aux clusters, une telle solution ne permet pas de voir les données dans leur globalité.

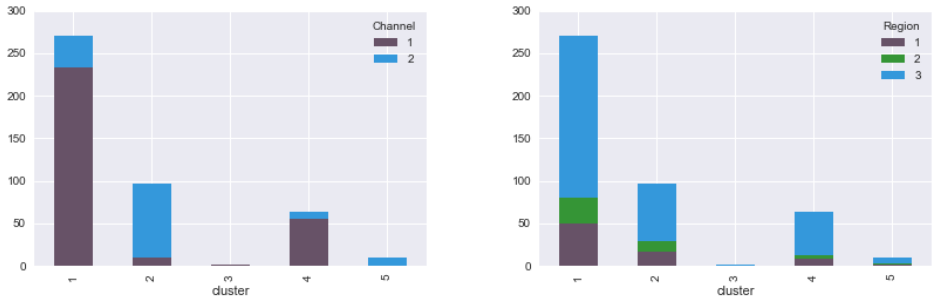
4.5.3 *Discussion des résultats obtenus*

Il est donc nécessaire de se pencher un peu plus sur les partitions formées pour pouvoir interpréter les résultats.

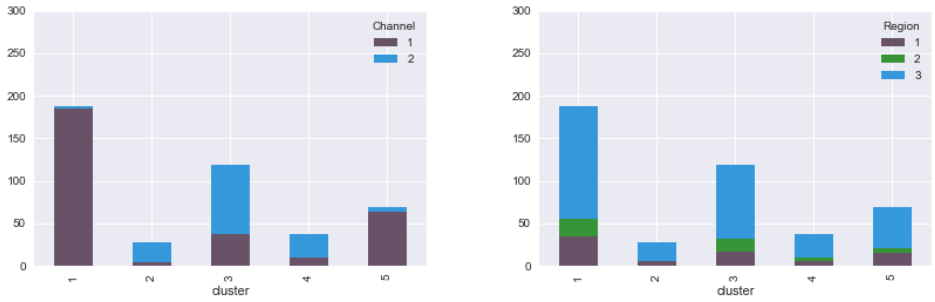
Répartition des catégories

Les catégories fournies avec les données, à savoir le type de commerce et la ville dans laquelle il se trouve, peuvent par exemple être comparées aux regroupements produits par les algorithmes. La figure 4.6 présente des histogrammes, séparés en ligne selon les 3 algorithmes étudiés et en colonne selon que les données sont colorées d'après le type de commerce du client de la vente en gros (colonne de gauche) ou d'après la ville dans laquelle il se trouve (colonne de droite). Tous les histogrammes étant à la même échelle, on constate à la hauteur des colonnes que la partition produite par l'algorithme des k-moyennes est plus déséquilibrée. En revanche, l'ordre des colonnes n'a pas d'importance et correspond simplement aux étiquettes produites par l'algorithme de clustering.

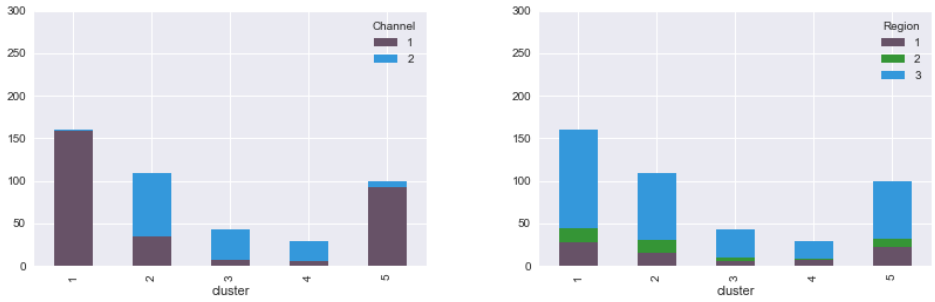
Les trois figures de droite montre la répartition des clients des différentes villes (selon la variable `Region`) et ne fait pas apparaître d'information particulière ; pour chaque algorithme, les couleurs sont réparties au sein des différentes colonnes sans qu'on constate de motif particulier. Dans la colonne de gauche en revanche, correspondant à la variable `Channel`, on aperçoit qu'à chaque fois deux clusters contiennent significativement plus de données correspondant aux hôtels et restaurants. Le plus gros



(a) Répartition des clusters pour les k-moyennes



(b) Répartition des clusters pour AWFCM



(c) Répartition des clusters pour Prosecco

FIG. *4.6 :* Répartition des données au sein des clusters, en séparant le type de commerce (à gauche) et la ville dans laquelle il est situé (à droite).

cluster, numéroté 1 pour les trois algorithmes, est à chaque fois quasi-exclusivement composé de points de points de cette catégorie (variable `Channel` égale à 1). En comparant ces 3 clusters deux à deux grâce à l'*Adjusted Rand Index*, on constate cependant que le cluster 1 de l'algorithme des k-moyennes est très différent des deux autres (ARI proche de 0), alors que les mêmes clusters pour AWFCM et Prosecco sont assez similaires (ARI supérieur à 0,5). Les deux algorithmes de subspace clustering produisent donc des partitions différentes de l'algorithme des k-moyennes, dont nous allons voir qu'il apporte des informations supplémentaires.

CLUSTER	Produits					
	FRAIS	LAITIERS	ÉPICERIE	CONGELÉS	ENTRETIEN	ÉPICERIE FINE
1	0	$1{,}13 \cdot 10^{-1}$	$2{,}66 \cdot 10^{-1}$	0	$6{,}20 \cdot 10^{-1}$	0
2	$1{,}60 \cdot 10^{-1}$	0	0	$7{,}45 \cdot 10^{-1}$	0	$9{,}43 \cdot 10^{-2}$
3	0	$4{,}33 \cdot 10^{-2}$	$3{,}01 \cdot 10^{-1}$	0	$6{,}55 \cdot 10^{-1}$	0
4	$9{,}23 \cdot 10^{-2}$	0	0	$6{,}47 \cdot 10^{-1}$	0	$2{,}60 \cdot 10^{-1}$
5	0	0	$8{,}00 \cdot 10^{-2}$	0	$9{,}20 \cdot 10^{-1}$	0

TAB. *4.3 :* Poids calculés par l'algorithme Prosecco.

Utilisation des vecteurs de poids

L'algorithme Prosecco, exécuté ici sur les données précédentes avec le paramètre $\gamma = 1$, produit une description parcimonieuse des sous-espaces qui témoigne des variables sélectionnées pour former le cluster. Ceci nous informe quant aux tendances prédominantes au sein des données.

Le tableau 4.3 donne les poids calculés par l'algorithme. La forme des lignes et la disposition des 0 nous montre que Prosecco a formé les clusters en suivant majoritairement deux variables, celle correspondant aux produits congelés et aux produits d'entretien. Les clusters 1, 3 et 5 ont donc des profils similaires en matière d'achat de produits d'entretien : la variance des données selon la variable correspondant aux produits d'entretien est beaucoup plus faible dans ces trois clusters.

En consultant la figure 4.6, on remarque que les clusters 1 et 5 contiennent majoritairement des hôtels et des restaurants plutôt que des commerces. Ce n'est cependant pas le cas de la totalité des clients de la catégorie `Channel = 1` ; cette analyse fournit donc une nouvelle catégorie au sein des données, qui permet de distinguer de nouveaux sous-groupes qui n'apparaissent pas aussi nettement dans les résultats d'un algorithme de clustering classique.

4.6 CONCLUSION

Ce chapitre a présenté un nouveau terme de pénalité qui contraint les solutions à appartenir au simplexe unitaire Δ, et qui pénalise également les solutions non parcimonieuses. L'opérateur proximal correspondant à cette pénalité est proposé sous forme algorithmique, et une preuve de sa correction est apportée, qui s'inscrit dans un cadre géométrique. L'algorithme de subspace clustering flou obtenu, Prosecco, est évalué sur des données artificielles. Il est comparé à l'algorithme de Borgelt et à EWKM ; Prosecco se montre plus stable que ces deux algorithmes mais, comme ces deux algorithmes, il reste dépendant d'un hyperparamètre que l'utilisateur doit choisir pour spécifier le niveau de parcimonie attendu.

Différentes améliorations et extensions peuvent être envisagées. La première porte sur la recherche du minimum de la meilleure approximation sur Δ, qui repose sur un simple algorithme de recherche linéaire. Un algorithme qui ne calculerait pas toutes les projections, qui plus est à chaque itération de la descente de gradient, permettrait de gagner en performances.

Plus généralement, les expériences conduites montrent que Prosecco et les algorithmes similaires dépendent d'un hyperparamètre dont le choix dépend des données et a un effet important sur le résultat des algorithmes. Dans nos expériences, nous nous limitons à un choix de paramètres parmi une plage de valeurs arbitraires, inspirées par notre pratique personnelle. En recherchant des critères de sélection de ces hyperparamètres, on améliorerait donc à la fois l'intérêt pratique de ces algorithmes et les expériences que nous conduisons.

Enfin, notons encore une fois que l'algorithme EWKM n'induit pas en théorie de parcimonie dans les solutions. C'est seulement en pratique que la précision limitée des nombres flottants ainsi que le post-traitement que nous utilisons dans les expériences suffisent à induire cet effet. Prosecco se distingue de cette approche et, dans une moindre mesure, de celle de l'algorithme de Borgelt, en combinant l'approche *continue* des algorithmes de subspace clustering flou à une exploration *discrète* d'une structure particulière, celle des faces (Δ_s), qui se rapproche un peu du principe de l'algorithme CLIQUE (voir section 2.2 p. 16).

5

UNE PÉNALITÉ POSSIBILISTE

L'approche du chapitre précédent consiste à traduire une contrainte de sommation à 1, qui permet d'éviter des solutions triviales du point de vue du subspace clustering, par l'attribution d'une pénalité infinie aux candidats ne respectant pas cette contraintes. Du point de vue du cadre théorique proposé dans le chapitre 3, cela revenait à projeter les vecteurs sur une surface particulière Δ pendant l'optimisation.

Cette contrainte peut cependant être affaiblie, autorisant une valeur de sommation comprise entre 0 et 1. Ce principe rappelle les algorithmes de clustering possibiliste, une variante des algorithmes de clustering flou qui ne contraignent pas la somme des degrés d'appartenance des points à valoir 1. Le degré calculé par l'algorithme pour un point donné est alors représentatif de l'importance de ce point au sein du cluster, fournissant une description plus informative des données.

Dans ce chapitre, nous présentons une version affaiblie de la pénalité du chapitre précédent, qui associe un coût fini aux vecteurs qui n'appartiennent pas à Δ. Nous explorons ainsi une autre approche, qui se distingue de la précédente en ce qu'elle pénalise les vecteurs solutions qui ne respectent pas la contrainte de sommation mais ne les interdit pas tout à fait. Dans un sens, cette pénalité introduit donc un comportement possibiliste dans les solutions.

Nous dérivons un opérateur proximal associé à cette pénalité, cette fois sous forme analytique, qui enrichit la fonction de projection sur le simplexe introduite dans le chapitre précédent. Contrairement au chapitre précédent, la pénalité n'introduit pas de parcimonie et rien ne contraint donc les solutions à appartenir aux faces du simplexe Δ.

Nous appliquons d'abord cette nouvelle pénalité à la matrice des poids W, ce qui, grâce à l'opérateur proximal dérivé et au schéma d'optimisation du chapitre 3, nous fournit un nouvel algorithme de subspace clustering flou baptisé PFSCM, pour *Proximal Fuzzy Subspace C-Means*. Nous illustrons le comportement de cet algorithme sur des données artificielles, sur lesquelles il produit un résultat similaire à celui d'AWFCM tout en se montrant plus stable.

Nous proposons ensuite de l'appliquer à la variable U, obtenant un nouveau problème de minimisation avec une contrainte relâchée sur U. En appliquant l'algorithme PSFCM à ce nouveau problème, nous obtenons un nouvel algorithme de subspace clustering flou qui s'autorise à ne pas affecter totalement les points : ce nouvel opérateur introduit un comportement possibiliste dans les algorithmes dans lesquels il est utilisé.

Nous intégrons cet opérateur à plusieurs algorithmes de subspace clustering flou, proposant ainsi des versions possibilistes des algorithmes AWFCM, EWKM, de Borgelt et Prosecco, baptisées respectivement WPPCM, PEWKM, PBorgelt et Possecco. En nous appuyant sur des exemples visuels dans un premier temps, nous montrons que ces nouveaux algorithmes se comportement effectivement comme des algorithmes possibilistes. Nous proposons ensuite une généralisation de l'expérience du chapitre précédent pour montrer qu'ils résistent également davantage au bruit que les algorithmes originaux.

La section 5.1 présente notre terme de pénalité possibiliste et propose l'opérateur proximal correspondant. La section 5.2 applique ce terme à la variable W, puis s'appuie sur les résultats du chapitre 3 pour constituer l'algorithme PFSCM. Dans la section 5.3, le même opérateur est appliqué à la variable U, ce qui permet d'obtenir plusieurs algorithmes de subspace clustering possibilistes. La section 5.4 propose enfin une généralisation de l'expérience du chapitre précédent qui confronte ces algorithmes à des données bruitées et montre que l'opérateur possibiliste que nous proposons permet de lutter partiellement contre ce bruit.

L'algorithme PFSCM a été présenté aux conférences IPMU 2016 et LFA 2016 (Guillon et al. 2016a ; Guillon et al. 2016b) ainsi que dans dans le journal FSS 2018 (Guillon et al. 2018a).

5.1 UNE NOUVELLE FONCTION DE PÉNALITÉ POSSIBILISTE

La fonction de pénalité introduite dans le chapitre 4 joue deux rôles : se restreindre aux solutions qui respectent une contrainte de sommation et favoriser les solutions parcimonieuses. Du point de vue du cadre de travail géométrique que nous présentons, cela revient à restreindre les solutions au simplexe Δ et à chercher une solution de coût minimum sur une face Δ_s de ce simplexe. Le premier effet est obtenu en donnant un coût infini aux solutions qui ne satisfont pas la contrainte, et est donc indépendant du coefficient de pondération des deux termes, noté γ.

Dans cette section, nous travaillons sur une matrice M qui désignera par la suite soit W soit U et nous introduisons un terme de pénalité qui donne un coût *fini* aux solutions qui ne respectent pas cette contrainte. Du point de vue de la minimisation de la fonction de coût, de telles solutions sont alors pénalisées mais autorisées : selon les données X, certains minima peuvent apparaître, dans lesquels certaines lignes ou colonnes de M peuvent ne plus sommer à 1.

La nouvelle pénalité est proposée dans la section 5.1.1. Dans la section 5.1.2, nous dérivons l'opérateur proximal correspondant et le présentons comme un enrichissement de la projection π sur le simplexe (équation 4.8 p. 53). La section 5.1.3 décrit les solutions qu'il produit et la section 5.1.4 prouve qu'il vérifie les quatre hypothèses du chapitre 3.

5.1.1 *Une nouvelle fonction de pénalité*

Étant donné une variable $M \in \mathbb{R}^{a\times b}$, nous commeçons par définir la fonction suivante :

$$G(M_k) = \left|\sum_{l=1}^{b} m_{kl} - 1\right| \tag{5.1}$$

La fonction de pénalité que nous étudions dans ce chapitre est donnée par la somme

$$\sum_{k=1}^{a} G(M_k)$$

Celle-ci exprime un coût nul pour les M_k dont la somme des composantes vaut 1 et un coût strictement positif pour les autres. Du fait de l'utilisation de la valeur absolue, la fonction G est non différentiable.

Étant donné un hyperparamètre $\gamma > 0$, le problème de minimisation que nous instancions dans les sections 5.2 et 5.3 est donc le suivant :

$$\begin{aligned} J(C,U,W) &= \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^2 \sum_{p=1}^{d} w_{rp}^2 (x_{ip}-c_{rp})^2 + \gamma \sum_{k=1}^{a} G(M_k) \\ \text{sous les contraintes} \quad & (C1*)\ \forall i \in [\![1,n]\!],\ \sum_{r=1}^{c} u_{ri} = 1, \\ & (C2*)\ \forall r \in [\![1,c]\!],\ \sum_{i=1}^{n} u_{ri} > 0, \\ & (C3*)\ \forall p \in [\![1,d]\!],\ \sum_{r=1}^{c} w_{rp} = 1 \end{aligned} \tag{5.2}$$

Dans une instance de ce problème, selon que M est remplacée par U ou par W, les contraintes $(C1*)$ et $(C2*)$ ou la contrainte $(C3*)$ doivent être supprimées pour que le schéma d'optimisation du chapitre 3 soit appliqué.

L'hyperparamètre γ est choisi par l'utilisateur. Des valeurs de γ élevées garantissent que la contrainte de sommation à 1 est respectée. Pour des valeurs plus faibles, il est cependant possible que des minima de J soient atteints pour des solutions telles que tous les vecteurs M_k ne somment pas nécessairement à 1.

5.1.2 *Un nouvel opérateur proximal*

La pénalité G est la somme de fonctions convexes, elle est donc convexe. Par conséquent, son opérateur proximal est bien défini. Alors que l'opérateur proximal du chapitre précédent était proposé sous la forme d'un algorithme qui devait être prouvé correct a posteriori, celui de notre nouvelle pénalité G peut être dérivé à l'aide de règles de calcul usuelles sur les opérateurs proximaux ; il est donc correct par construction.

Avec $M \in \mathbb{R}^{a\times b}$, on note K le vecteur $(1,\ 1, \dots 1)$ de $\mathbb{R}^{1\times b}$, tel que $K \cdot K^\top = b$. Afin d'alléger les notations, on note également $V = M_k$ dans la suite de cette section.

Théorème 5.1.1 *Soient* $G(V) = \left|\sum_{l=1}^{b} v_l - 1\right|$ *et* $L > 0$, *alors :*

$$\operatorname{prox}_{\frac{\gamma}{L}G}(V) = V + \frac{1}{b}K^\top \cdot \left(1 + \operatorname{prox}_{\frac{\gamma b}{L}|\cdot|}(K\cdot V - 1) - K \cdot V\right) \tag{5.3}$$

où $\operatorname{prox}_{\lambda|\cdot|}(x) = \operatorname{sign}(x)\cdot\max(|x|-\lambda, 0)$ *est l'opérateur de la fonction valeur absolue.*

Démonstration *La formule de l'équation 5.3 est obtenue en utilisant les propriétés et expressions des opérateurs proximaux établis par Combettes et Pesquet (2011) : $G(V) = \varphi(K \cdot V)$ où $\varphi(x) = |x - 1|$, d'où :*

$$\begin{aligned}
\text{prox}_{\varphi}(y) &= 1 + \text{prox}_{|\cdot|}(y - 1) \\
\text{prox}_{G}(V) &= V + \frac{1}{b} K^{\top} \cdot \Big(\text{prox}_{\varphi}(K \cdot V) - K \cdot V\Big) \\
&= V + \frac{1}{b} K^{\top} \cdot (1 + \text{prox}_{b|\cdot|}(K \cdot V - 1) - K \cdot V)
\end{aligned}$$

La formule de $\text{prox}_{\frac{\gamma}{L}G}(V)$ est donnée par la propriété de post-composition.

L'opérateur proximal obtenu en équation 5.3 est à rapprocher de l'équation 4.8 de la projection sur le simplexe Δ de $\mathbb{R}^b$, donnée par

$$\begin{aligned}
\forall l, \pi(V)_l &= v_l + \frac{1}{b}\left(1 - \sum_{m=1}^{b} v_m\right) \\
\text{soit } \pi(V) &= V + \frac{1}{b} K^{\top} \cdot (1 - K \cdot V)
\end{aligned}$$

Ce nouvel opérateur proximal apparaît donc comme une extension de la projection euclidienne de $\Delta_{\leq 1}$ sur Δ : un nouveau terme apparaît, qui peut empêcher la somme $\sum_{l=1}^{b} v_l$ de valoir 1, selon les valeurs de γ, b et L.

En revanche, aucune généralisation des projections π_s du chapitre précédent n'est proposée : la nouvelle pénalité n'introduit aucune parcimonie dans les solutions.

5.1.3 *Propriété des solutions*

La section précédente propose un opérateur proximal pour la pénalité G, donné par l'équation 5.3. Elle montre aussi que cet opérateur proximal enrichit l'expression de l'opérateur de projection sur Δ du terme suivant :

$$\text{prox}_{\frac{\gamma b}{L}|\cdot|}(K \cdot V - 1) \tag{5.4}$$

qui fait intervenir le vecteur V, la grandeur b ainsi que les constantes γ et L, cette dernière étant calculée d'après la section 3.2.3 p. 44.

Dans cette section nous explicitons l'effet de ce terme sur les solutions calculées lors de l'itération de la descente proximale et nous montrons que la fonction de pénalité et son opérateur proximal satisfont bien les hypothèses formulées dans le chapitre 3, nécessaires pour appliquer l'algorithme PSFCM.

Influence de γ sur la sommation à 1

L'opérateur proximal de la valeur absolue rencontré dans l'équation 2.42 p. 37 est rappelé ci-dessous, pour un réel $\lambda > 0$:

$$\begin{aligned}\text{prox}_{\lambda|\cdot|}(v) &= \underset{v' \in \mathbb{R}}{\text{argmin}} \left\{ \frac{1}{2} (v - v')^2 + \lambda |v'| \right\} \\ &= \begin{cases} v - \lambda & \text{si } v \geq \lambda \\ 0 & \text{si } -\lambda < v < \lambda \\ v + \lambda & \text{si } v \leq -\lambda \end{cases} \end{aligned} \tag{5.5}$$

En vertu de la proposition 3.2.2, à chaque itération de la descente proximale, l'étape de descente selon le gradient $\nabla F(V^t)$ vérifie la propriété suivante :

Proposition 5.1.1 *Soit $V^t \in \Delta_{\leq 1}$. Avec L tel que donné par l'équation 3.7 p. 45, on a*

$$V^t - \frac{1}{L} \cdot \nabla F(V^t) \in \Delta_{\leq 1} \quad \textit{et} \quad K \cdot V^t - 1 \leq 0$$

D'après l'équation 5.5 et la proposition précédente, le terme de l'équation 5.4 est négatif ou nul. Au sein de l'équation 5.3, ce terme agit donc comme une pénalité supplémentaire qui, quand elle est non nulle, empêche les composantes de V de sommer à 1.

Proposition 5.1.2 *Soient $V \in \Delta_{\leq 1}$ et $\gamma, L > 0$, $\text{prox}_{\gamma G}(V) \in \Delta_{\leq 1}$.*

Démonstration *Par hypothèse, $V \in \Delta_{\leq 1}$ donc $1 - K \cdot V_k > 0$. Soit $\lambda = \frac{\gamma b}{L}$, on distingue les cas selon la valeur de équation 5.4. D'après l'équation 5.5, deux cas sont possibles :*

- *si $K \cdot V - 1 \leq -\lambda$ alors*

  $$\text{prox}_{\frac{\gamma b}{L}|\cdot|}(K \cdot V - 1) \leq K \cdot V - 1$$

  et le membre droit de l'équation 5.3 est positif mais ne somme pas à 1 ;

- *si $-\lambda < K \cdot V - 1 < \lambda$ alors*

  $$\text{prox}_{\frac{\gamma b}{L}|\cdot|}(K \cdot V - 1) = 0$$

  et $\text{prox}_G(V) \in \Delta$.

Dans les deux cas, on a bien $\text{prox}_{\gamma G}(V) \in \Delta_{\leq 1}$.

Ce terme peut également s'annuler quand la valeur de $\lambda = \frac{\gamma c}{L}$ est trop grande, c'est-à-dire quand γ est choisi très grand par l'utilisateur. Dans ce cas, $\text{prox}_{\frac{\gamma}{L} G}(V) = \pi(V)$, ce qui correspond aux solutions intuitives du problème 5.8 pour une valeur de γ élevée.

5.1.4 *Propriétés de G et de son opérateur proximal*

Afin d'appliquer l'algorithme PSFCM, nous vérifions que les propriétés de la section 3.2.4 p. 45 sont bien satisfaites :

P1 : la pénalité introduite dans le problème 5.2 est bien définie sur une variable M qui désigne au choix U ou W ;

P2 : cette pénalité est une somme de termes positifs, elle est donc positive ;

P3 : le théorème 5.1.1 donne son opérateur proximal ;

P4 : l'ensemble $\Delta_{\leq 1}$ est stable par cet opérateur proximal d'après la proposition 5.1.2.

Dans les deux sections 5.2 et 5.3 suivantes, les résultats généraux de cette section sont appliqués successivement aux variables W et U et l'algorithme PSFCM est instancié pour obtenir de nouveaux algorithmes de subspace clustering flou.

5.2 APPLICATION AUX POIDS DES DIMENSIONS

Dans cette section nous appliquons le terme de pénalité introduit dans la section 5.1 à la matrice des poids W. Nous obtenons ainsi un problème de minimisation proche de celui de l'algorithme AWFCM, excepté que la contrainte $(C3)$ forçant la somme $\sum_{p=1}^{d} w_{rp}$ à valoir 1 est relâchée sous la forme d'une pénalité.

La section 5.2.1 présente la nouvelle fonction de coût appliquant la pénalité précédente à la matrice W et l'opérateur proximal associé. Dans la section 5.2.2, nous proposons l'algorithme PFSCM, pour *Proximal Fuzzy Subspace c-Means*, qui est l'instance de l'algorithme PSFCM du chapitre 3 minimisant notre nouvelle fonction de coût. Nous proposons ensuite un exemple de résultat produit par l'algorithme dans la section 5.2.3.

5.2.1 *Fonction de coût proposée et opérateur proximal*

En remplaçant la variable M du problème 5.2 par W, on obtient le problème suivant :

$$J(C,U,W) = \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^2 \sum_{p=1}^{d} w_{rp}^2 (x_{ip}-c_{rp})^2 + \gamma \sum_{r=1}^{c} \left| \sum_{p=1}^{d} w_{rp} - 1 \right|$$

$$\text{sous les contraintes} \quad (C1)\ \forall i \in [\![1,n]\!],\ \sum_{r=1}^{c} u_{ri} = 1, \tag{5.6}$$

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

Dans le problème 5.6, les deux contraintes $(C1)$ et $(C2)$ portant sur la variable U sont maintenues et le terme de pénalité non différentiable précédent est appliqué à la variable W.

L'opérateur proximal dérivé dans la section 5.1.2 fait intervenir la variable $M \in \mathbb{R}^{a\times b}$. Un soin particulier doit être apporté aux dimensions de M quand on la remplace

Algorithme 4 L'algorithme PFSCM

procédure PFSCM($X, c, \gamma, \varepsilon$)
  Initialisation : $U, C \leftarrow \text{FCM}(X, c, \varepsilon)$
    $\forall r, W_r \leftarrow \left[\frac{1}{d}, \cdots, \frac{1}{d}\right]$
  répéter
    répéter
      Mettre à jour C d'après l'équation 2.11 p. 21
      Mettre à jour U d'après l'équation 2.12 p. 21
    jusqu'à *convergence*($\{C, U\}, \epsilon$)
    Calculer L d'après l'équation 3.6 p. 45
    répéter
      $W_{\text{temp}} \leftarrow W - \frac{1}{L} \cdot \nabla F(W)$
      $W \leftarrow \text{prox}_{\frac{\gamma}{L}G}(W_{\text{temp}})$ d'après l'équation 5.7
    jusqu'à *convergence*($\{W\}, \varepsilon$)
  jusqu'à *convergence*($\{C, U, W\}, \varepsilon$)

par $W \in \mathbb{R}^{c \times d}$. Nous explicitons l'opérateur proximal correspondant à la pénalité du problème 5.6 pour lever toute ambiguïté :

Théorème 5.2.1 *Soit $L > 0$, l'opérateur proximal de la pénalité du problème 5.6 est*

$$\text{prox}_{\frac{\gamma G}{L}}(W_r) = W_r + \frac{1}{d} K^\top \cdot (1 + \text{prox}_{\frac{\gamma d}{L}|\cdot|}(K \cdot W_r - 1) - K \cdot W_r) \tag{5.7}$$

5.2.2 *L'algorithme PFSCM*

Nous appliquons l'algorithme générique PSFCM 1 p. 46 au problème 5.6 pour obtenir un nouvel algorithme de subspace clustering flou, PFSCM, présenté par l'algorithme 4. Comme dans les chapitres précédents, le résultat est un algorithme d'optimisation alternée séparant l'optimisation des variables U et C, qui sont mises à jour à l'aide des mêmes équations que les algorithme AWFCM et Prosecco, et de la variable W, mise à jour par descente de gradient proximale.

Ici encore, l'hyperparamètre $\gamma > 0$ module le degré auquel les poids (W_r) somment à 1. Nous observons expérimentalement que des valeurs élevées sont nécessaires pour exercer cette contrainte, de l'ordre de $\gamma = 1000$. Des valeurs supérieures ne changent pas les résultats de PFSCM. En revanche, lorsque γ diminue la contrainte de sommation n'est plus respectée et les valeurs des (w_{rp}) tendent vers 0.

5.2.3 *Illustration de l'algorithme*

Nous considérons un jeu de données artificielles inspiré de l'expérience du chapitre précédent, composé de deux plans générés selon des lois uniformes dan un espace en trois dimensions. Les deux plans sont sécants et sont représentés sur la figure 5.1, à gauche, colorés en fonction du plan dans lequel ils ont été générés.

Sur la même figure, à droite, le résultat de l'algorithme PFSCM est représenté comme dans le chapitre précédent : les points sont colorés en fonction du cluster auquel ils

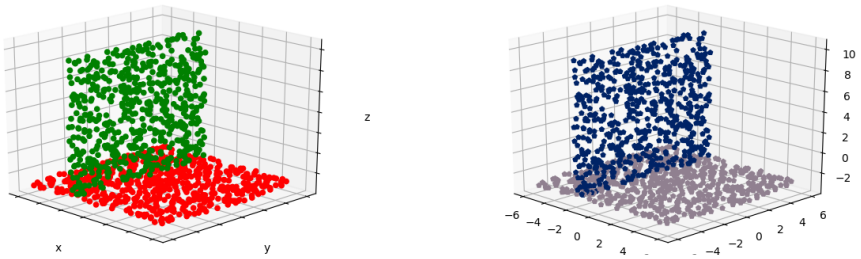


FIG. *5.1 :* Deux plans générés dans un espace en trois dimensions (à gauche) et le résultat de l'algorithme PFSCM sur ces données (à droite).

γ	CLUSTER	AXE x	AXE y	AXE z	$\sum(w_{rp})$
1000	bleu	$4{,}90 \cdot 10^{-4}$	$4{,}76 \cdot 10^{-4}$	$9{,}99 \cdot 10^{-1}$	2
	gris	$9{,}99 \cdot 10^{-1}$	$2{,}74 \cdot 10^{-4}$	$2{,}29 \cdot 10^{-4}$	
100	bleu	$5{,}41 \cdot 10^{-4}$	$5{,}26 \cdot 10^{-4}$	$9{,}98 \cdot 10^{-1}$	2
	gris	$3{,}93 \cdot 10^{-1}$	$6{,}06 \cdot 10^{-4}$	$4{,}18 \cdot 10^{-4}$	
10	bleu	$6{,}88 \cdot 10^{-4}$	$6{,}66 \cdot 10^{-4}$	$2{,}04 \cdot 10^{-1}$	0,49
	gris	$2{,}29 \cdot 10^{-1}$	$7{,}11 \cdot 10^{-4}$	$6{,}06 \cdot 10^{-4}$	
1	bleu	$6{,}85 \cdot 10^{-5}$	$6{,}51 \cdot 10^{-5}$	$3{,}90 \cdot 10^{-2}$	0,17
	gris	$1{,}25 \cdot 10^{-1}$	$7{,}27 \cdot 10^{-5}$	$6{,}38 \cdot 10^{-5}$	
AWFCM	bleu	$2{,}56 \cdot 10^{-4}$	$2{,}50 \cdot 10^{-4}$	$9{,}99 \cdot 10^{-1}$	—
	gris	$9{,}99 \cdot 10^{-1}$	$2{,}74 \cdot 10^{-4}$	$2{,}29 \cdot 10^{-4}$	

TAB. *5.1 :* Résultats des algorithmes PFSCM et AWFCM sur les données de la figure 5.1.

sont majoritairement affectés. L'algorithme retrouve les deux plans générés. Ici, les paramètres $\gamma = 1000$ et $c = 2$ ont été utilisés pour produire la figure ; cependant nous observons expérimentalement que l'affectation des points ne change pas lorsque γ décroît.

Ceci nous permet d'observer l'effet de γ sur les vecteurs de poids. Le tableau 5.1 représente les vecteurs de poids W_r calculés par l'algorithme PFSCM pour des valeurs de γ décroissantes, ainsi que ceux calculés par l'algorithme AWFCM. Par défaut, l'algorithme AWFCM converge cependant vers une solution de clustering complètement différente ; il est donc nécessaire de l'initialiser avec le résultat de l'algorithme PFSCM pour pouvoir comparer les vecteurs de poids calculés.

Le tableau 5.1 montre que les vecteurs W_r ne somment plus à 1 quand γ diminue. Ils n'atteignent cependant pas la valeur 0 et ne semblent pas diminuer à la même vitesse.

L'intérêt pratique de PFSCM quand γ est faible semble limité. Pour des valeurs de γ élevées, il fournit cependant un autre algorithme de subspace clustering flou alternatif à AWFCM qui produit des solutions plus intuitives. Le chapitre 6 propose une expérience mettant en valeur quelques-unes de leurs différences.

5.3 APPLICATION AUX DEGRÉS D'APPARTENANCE

En appliquant le terme de pénalité et de l'opérateur proximal de la section 5.1 à la variable U, nous montrons qu'il permet de former des algorithmes de subspace clustering possibiliste, qui produisent des degrés d'appartenance u_{ri} reflétant l'importance du point x_i au sein du cluster C_r. Mieux, cet opérateur est générique, pouvant ainsi être intégré aux algorithmes étudiés dans les chapitres précédents pour en produire des versions possibilistes.

Dans cette section nous introduisons l'algorithme *Weighted Proximal Possibilistic c-Means* (WPPCM), qui résout le problème 5.2 et produit des solutions possibilistes. En outre, nous intégrons cet opérateur aux algorithmes de Borgelt, EWKM et Prosecco, produisant des versions possibilistes de ces algorithmes, que nous baptisons respectivement PBorgelt, PEWKM et Possecco.

La section 5.3.1 détaille la fonction de coût qui nous intéresse dans cette section et la façon dont la pénalité et l'opérateur proximal du début de la section 5.1 sont appliqués à U. Cet opérateur est utilisé dans la section 5.3.2 dans l'algorithme WPPCM et dans la section 5.3.3 pour présenter trois nouveaux algorithmes de subspace clustering possibilistes, dont nous illustrons le comportement sur des données en faible dimension afin d'en développer l'intuition en section 5.3.4.

5.3.1 *Fonction de coût et pénalité sur les degrés d'appartenance*

Inspirés par les algorithmes possibilistes présentés dans la section 2.1.2 à la page 14, nous proposons d'appliquer la pénalité de la section 5.1 à la variable U dans le but d'obtenir des degrés d'appartenance liés à la représentativité des points. Étant donné un hyperparamètre γ, le problème de minimisation devient

$$J(C, U, W) = \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^2 \sum_{p=1}^{d} w_{rp}^2 (x_{ip} - c_{rp})^2 + \gamma \sum_{i=1}^{n} \left| \sum_{r=1}^{c} u_{ri} - 1 \right| \tag{5.8}$$

$$\text{sous les contraintes} \quad (C3)\ \forall p \in [\![1, d]\!],\ \sum_{r=1}^{c} w_{rp} = 1$$

Contrairement à la section 5.2, ce sont maintenant les deux contraintes $(C1)$ et $(C2)$ portant sur U qui sont absentes du problème 5.8.

De plus, la pénalité, qui porte maintenant sur la variable U, présente plusieurs différences avec ce qui précède. Premièrement, la pénalité G est n'est plus appliquée au vecteur colonne W_r mais au vecteur *ligne* $U_{\cdot i}$, ce qui signifie que les résultats de la section 5.1 et notamment l'opérateur proximal calculé doivent être adaptés. Ensuite, l'ordre des sommes du terme de pénalité est échangé en comparaison du problème 2.8 : ici le terme pénalise la somme des $\sum_{r=1}^{c} u_{ri}$ et tous les (u_{ri}) ne sont donc pas indépendants.

L'énoncé du théorème 5.1.1 devient le suivant, correspondant à une transposition de la variable $M \in \mathbb{R}^{a \times b}$ de la section 5.1.2 :

Algorithme 5 L'algorithme WPPCM

procédure WPPCM($X, c, \gamma, \varepsilon$)
  Initialisation : $U, C \leftarrow \text{FCM}(X, c, \varepsilon)$
    $\forall r, W_r \leftarrow \left[\frac{1}{d}, \cdots, \frac{1}{d}\right]$
  répéter
    répéter
      Mettre à jour C d'après l'équation 2.11 p. 21
      Mettre à jour W d'après l'équation 2.13 p. 22
    jusqu'à *convergence*$(\{C, W\}, \epsilon)$
    Calculer L d'après l'équation 3.7 p. 45
    répéter
      $U_{\text{temp}} \leftarrow U - \frac{1}{L} \cdot \nabla F(U)$
      $U \leftarrow \text{prox}_{\frac{\gamma}{L}G}(U_{\text{temp}})$ (voir équation 5.9)
    jusqu'à *convergence*$(\{U\}, \varepsilon)$
  jusqu'à *convergence*$(\{C, U, W\}, \varepsilon)$

Théorème 5.3.1 *Soit $L > 0$, l'opérateur proximal de la pénalité du problème 5.8 est donné par :*

$$\text{prox}_{\frac{\gamma}{L}G_i}(U_{\cdot i}) = U_{\cdot i} + \frac{1}{c}K^\top \cdot \left(1 + \text{prox}_{\frac{\gamma c}{L}|\cdot|}(K \cdot U_{\cdot i} - 1) - K \cdot U_{\cdot i}\right) \tag{5.9}$$

où $\text{prox}_{\lambda|\cdot|}(x) = \text{sign}(x) \cdot \max(|x| - \lambda, 0)$.

5.3.2 *L'algorithme WPPCM*

Nous appliquons l'algorithme générique 1 p. 46 au problème 5.8 pour obtenir un nouvel algorithme de subspace clustering flou, WPPCM, présenté par l'algorithme 5. Cet algorithme procède encore par optimisation alternée mais cette fois-ci ce sont les variables C et W qui sont optimisées ensemble à l'aide d'équations de mise à jour.

L'optimisation de la variable U est analogue à celle de la variable W dans Prosecco. L'algorithme est initialisé de la même façon, grâce aux c-moyennes floues ; cependant, contrairement à l'algorithme Prosecco ou à PFSCM, nous observons expérimentalement que cette initialisation est cruciale. Sans elle, l'algorithme peut converger vers des minima locaux très différents.

Comme pour Prosecco, la valeur de γ influence grandement les résultats. Nous proposons 0,1 comme valeur initiale typique. Une valeur trop grande forçant les $U_{\cdot i}$ à sommer à 1, nous préconisons de diminuer progressivement la valeur de γ en contrôlant l'évolution de la somme $\sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}$.

L'algorithme WPPCM peut également être vu comme une combinaison de l'opérateur de l'équation 5.9 et de l'algorithme AWFCM, dont il reprend deux des équations de mise à jour.

5.3.3 *Intégration de l'opérateur possibiliste aux algorithmes parcimonieux*

En appliquant le procédé du chapitre 3, l'opérateur de la section 5.1.2 peut être intégré à d'autres algorithmes tels que l'algorithme Prosecco du chapitre précédent. Cela

Algorithme 6 L'algorithme Possecco

procédure POSSECCO($X, c, \gamma_U, \gamma_W \varepsilon$)
  Initialisation : $U, C \leftarrow \text{FCM}(X, c, \varepsilon)$
    $\forall r, W_r \leftarrow [\frac{1}{d}, \cdots, \frac{1}{d}]$
  répéter
    Mettre à jour C d'après l'équation 2.11 p. 21
    Calculer L d'après l'équation 3.7 p. 45
    répéter
      $U_{\text{temp}} \leftarrow U - \frac{1}{L} \cdot \nabla F(U)$
      $U \leftarrow \text{prox}_{\frac{\gamma_U}{L} G}(U_{\text{temp}})$ (voir équation 5.9)
    jusqu'à *convergence*(U, ε)
    Calculer L d'après l'équation 3.6 p. 45
    répéter
      $W_{\text{temp}} \leftarrow W - \frac{1}{L} \cdot \nabla F(W)$
      $W \leftarrow \text{MIN}\ell_0(W_{\text{temp}}, \frac{\gamma_W}{L})$ (voir algorithme 2 p. 55)
    jusqu'à *convergence*(W, ε)
  jusqu'à *convergence*(C, U, W, ε)

revient à résoudre des problèmes d'optimisation différents, dans lesquels les contraintes $(C1)$ et $(C2)$, portant sur U, ont été remplacées par la pénalité de l'équation 5.1.

L'algorithme 6 présente Possecco, le résultat de cette transformation pour l'algorithme Prosecco. Celui-ci correspond à l'optimisation du problème suivant :

$$\sum_{i=1}^{n} \sum_{r=1}^{c} u_{ri}^2 \sum_{p=1}^{d} w_{rp}^2 (x_{ip} - c_{rp})^2 + \gamma_W \sum_{p=1}^{d} G(W_r) + \gamma_U \sum_{i=1}^{n} \left| \sum_{r=1}^{c} (u_{ri}) - 1 \right| \quad (5.10)$$

où G est la fonction définie par l'équation 4.2 p. 50.

L'algorithme Possecco repose sur le même principe que les précédents et s'appuie sur l'optimisation proximale pour les deux variables U et W. La première boucle intérieure a disparu : seule la variable C étant mise à jour par une équation, la convergence est immédiate et sa valeur ne change pas entre deux itérations.

Enfin, deux hyperparamètres γ_U et γ_W sont nécessaires. En fonction des données et de l'importance souhaitée pour les deux pénalités, des valeurs complètement différentes doivent être utilisées. Bien que les deux effets puissent être contrôlés indépendamment, cela complexifie l'utilisation de l'algorithme, comme le montre la section expérimentale suivante.

Les algorithmes PBorgelt et PEWKM sont dérivés selon les mêmes principes et résolvent respectivement chacun des problèmes suivants :

$$F_{PB}(C, U, W) = \sum_{i=1}^{n} \sum_{r=1}^{c} h(u_{ri}) \sum_{p=1}^{d} g(w_{rp})(x_{ip} - c_{rp})^2 + \gamma_W \sum_{p=1}^{d} w_{rp} \log w_{rp} + \gamma_U \sum_{i=1}^{n} \left| \sum_{r=1}^{c} u_{ri} - 1 \right|$$

$$\text{sous les contraintes} \quad \forall r \in [\![1, c]\!], \sum_{p=1}^{d} w_{rp} = 1$$

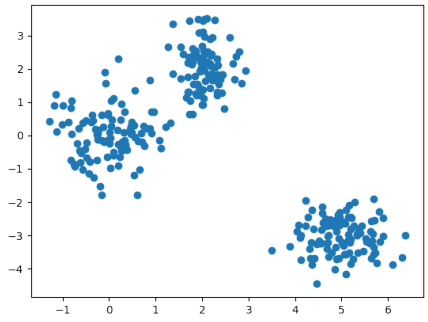


FIG. *5.2 :* Trois gaussiennes bidimensionnelles échantillonnées.

où h et g sont les fonctions décrites dans la section 2.4.1 p. 27, et

$$F_{PE}(C,U,W) = \sum_{i=1}^{n}\sum_{r=1}^{c} u_{ri}^2 \sum_{p=1}^{d} w_{rp}(x_{ip}-c_{rp})^2 + \gamma_W \sum_{p=1}^{d} w_{rp}\log w_{rp} + \gamma_U \sum_{i=1}^{n}\left|\sum_{r=1}^{c} u_{ri} - 1\right|$$

$$\text{sous les contraintes} \quad \forall r \in [\![1,c]\!],\ \sum_{p=1}^{d} w_{rp} = 1$$

5.3.4 *Illustration du comportement possibiliste*

Les algorithmes de clustering possibiliste étant aussi difficiles à évaluer que les algorithmes de subspace clustering classiques, nous privilégions ici encore une approche graphique dans un premier temps afin de développer l'intuition de leur comportement.

Un algorithme de clustering possibiliste est en mesure de caractériser la typicalité d'un point relativement à son cluster. L'algorithme Possecco, exécuté sur ces points, produit une matrice U telle que les colonnes $U_{.i}$ ne somment pas à 1. On peut alors *équeuter* le nuage de points en enlevant du résultat les points (X_i) tels que $\forall r, u_{ri} \leq \eta$, pour un paramètre $\eta \in [0,1]$ fixé.

En deux dimensions

On illustre le comportement possibiliste de l'opérateur de l'équation 5.9 à l'aide d'un exemple visuel simple. La figure 5.2 représente trois gaussiennes bidimensionnelles multivariées. La plupart des points de chaque gaussienne sont concentrés au milieu des gaussiennes, mais un certain nombre de points sont générés loin du centre.

La figure 5.3 représente une telle opération, les points équeutés étant représentés en noir. On constate que les points les moins représentatifs des gaussiennes, i.e. ceux qui sont placés en périphérie, sont équeutés en premier. On ne conserve alors que les points les plus représentatifs de chaque cluster.

Le cluster bleu clair, identifié par Possecco comme étant unidimensionnel (le poids de la dimension verticale est fixé à 0), est équeuté en priorité dans sa dimension la

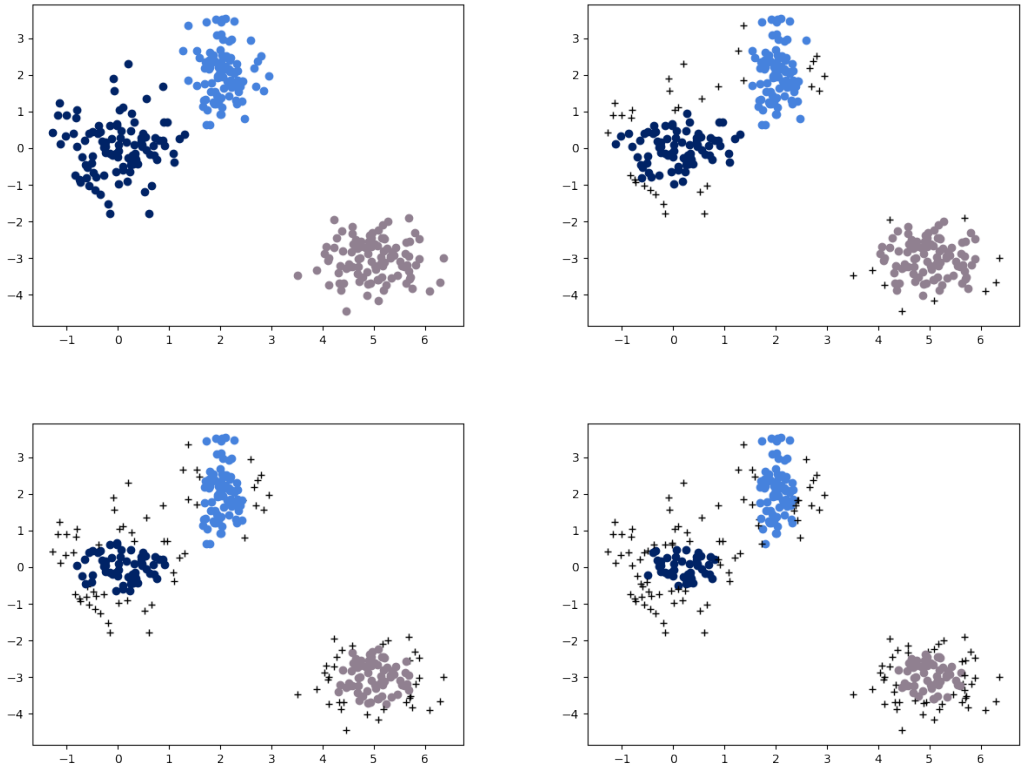


FIG. *5.3 :* Application de l'algorithme Possecco et équeutage du résultat pour $\eta \in \{0,0,3,0,6,0,9\}$.

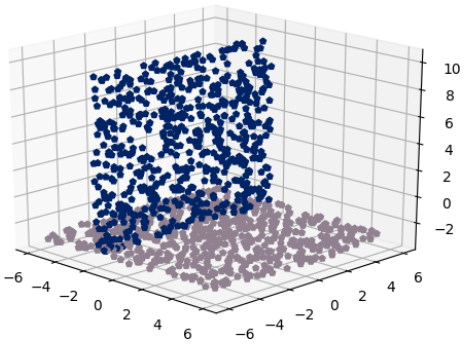
moins caractéristique. Cette interaction entre les effets parcimonieux et le possibiliste permet donc de simplifier encore les informations présentées à l'utilisateur par l'algorithme de clustering.

En trois dimensions

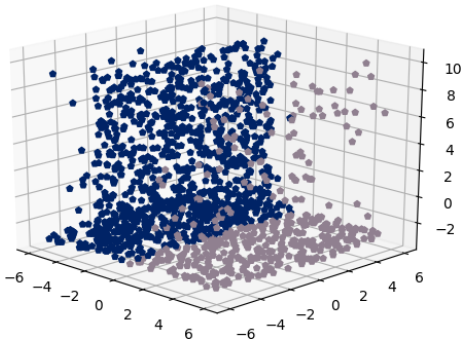
La figure 5.4 présente des données générées avec des règles similaires à celles du chapitre précédent, mais en trois dimensions et en ajoutant du bruit. La figure en haut à gauche représente le résultat de l'algorithme Prosecco, exécuté avec les paramètres $\gamma = 1$ et $c = 2$, avant l'ajout de bruit dans les données. Les points étant encore colorés en fonction du cluster auquel ils sont majoritairement affectés, on constate que Prosecco identifie les deux plans attendus.

En haut à droite, les données précédentes sont bruitées : de nouveaux points sont ajoutés, générés selon une loi uniforme dans le cube formé par les données. Ce bruit change considérablement le résultat de l'algorithme Prosecco, exécuté avec les mêmes paramètres, qui identifie des clusters très différents. Ce résultat reste le même lorsque l'on augmente γ : l'approche parcimonieuse du chapitre précédent, qui élimine les dimensions superflues, a un effet sur la matrice W produite mais empêche l'algorithme d'identifier les deux plans.

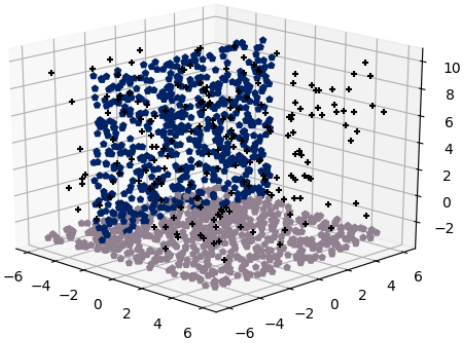
La deuxième ligne de la figure 5.4 représente le résultat de l'algorithme Possecco sur ces mêmes données bruitées, exécuté avec les paramètres $c = 2$, $\gamma_U = 0{,}1$ et $\gamma_W = 1$.



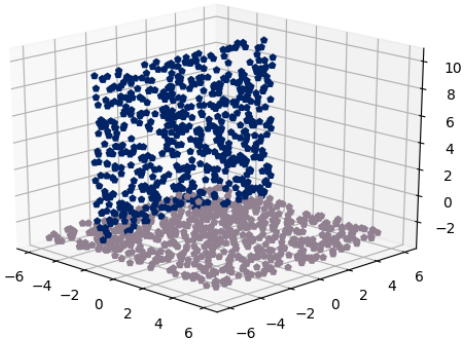
(a) Deux plans identifiés par l'algorithme Prosecco



(b) Résultat de Prosecco en ajoutant 20% de bruit



(c) Résultat de Possecco sur les données bruitées, points équeutés ($\eta = 0{,}1$) représentés en noir



(d) Résultat de Possecco sur les données bruitées, points équeutés ($\eta = 0{,}1$) masqués

FIG. *5.4 :* Comparaison graphique de Prosecco et Possecco sur des données bruitées.

Sur la figure de gauche, les points équeutés ($\forall r, u_{ri} \leq 0{,}1$) sont représentés en noir. Sur la figure de droite, ils sont masqués, pour produire une représentation plus lisible. On constate que la solution proposée par l'algorithme affecte des valeurs de u_{ri} très faibles à la plupart des points « bruit ».

Ces points jouent alors un rôle plus faible dans le calcul des coordonnées des centres et des poids des dimensions. En conséquence, Possecco est moins sensible au bruit ajouté aux données et il identifie les deux plans trouvés par Prosecco dans les données non bruitées, à l'exception de quelques points situés à la base du cluster gris et ajoutés au cluster bleu.

5.4 VALIDATION EXPÉRIMENTALE : RÉSISTANCE AU BRUIT

Nous proposons une généralisation de l'expérience sur données artificielles du chapitre précédent, présentée en section 4.4 p. 57 qui consiste à ajouter un bruit uniforme aux données pour observer le comportement des algorithmes. Nous montrons que l'utilisation de l'opérateur possibiliste améliore considérablement la résistance au bruit des algorithmes de Borgelt, EWKM et Prosecco, justifiant l'intérêt pour le subspace clustering possibiliste.

5.4.1 *Protocole expérimental*

Les données utilisées sont les mêmes que dans l'expérience du chapitre précédent, à savoir des clusters générés dans des sous-espaces de dimensionnalité très variable comportant 600 points chacun. La dimension d de l'espace $\mathbb{R}^d$ varie entre 20 et 58, la dimensionnalité des sous-espaces entre 1 et $d-4$. Les règles exactes de génération des données sont précisées dans la section 4.4.2 p. 59.

À ces données est ensuite ajouté un bruit Y, correspondant à des points générés selon une loi uniforme, dans les bornes formées par les données. Le nombre de points de bruit générés varie comme un pourcentage ψ du nombre de points $n = \text{card}(X)$, allant de 5% à 40% des données.

Critères d'évaluation

Les algorithmes sont évalués comme dans l'expérience précédente, le bruit n'étant pas pris en compte : les algorithmes doivent donc retrouver les clusters générés et leur dimensionnalité, le degré d'appartenance des points de Y étant ignoré dans le calcul du ratio donné par l'équation 4.11 p. 60.

Choix des hyperparamètres

Enfin, les hyperparamètres utilisés pour les différents algorithmes sont adaptés à cette nouvelle expérience :

- les algorithmes de Borgelt et PBorgelt utilisent les valeurs de β suivante : 10^{-3}, 10^{-2} et 10^{-1} ;
- les algorithmes EWKM (respectivement PEWKM) et Prosecco (respectivement Possecco) utilisent les valeurs de γ (respectivement γ_W) suivantes : 10^{-2}, 10^{-1}, 0.5, 10 et 50 ;
- les algorithmes PBorgelt, PEWKM et Possecco utilisent les valeurs de γ_U suivantes : 10^{-2} et 10^{-1}.

L'évaluation des algorithmes étant supervisée, il est possible de sélectionner le ou les paramètre(s) ayant conduit aux meilleurs résultats. Ceci est fait pour chaque valeur de k et de ψ : sont retenus les paramètres qui conduisent au meilleur ratio moyen pour un k et un ψ donnés.

Les algorithmes utilisant deux hyperparamètres sont donc exécutés pour toutes les combinaisons de valeurs de ces paramètres ainsi que pour tous couples (k, ψ), ce qui explique le faible espace de recherche du paramètre γ_U. Cette dépendance à plusieurs hyperparamètres limite à la fois l'utilisation des algorithmes possibilistes que nous proposons, mais aussi leur étude, et fait partie de nos perspectives de recherche.

5.4.2 *Interprétation des résultats expérimentaux*

La moyenne des résultats de chaque algorithme sur 100 exécutions pour $k \in \{2, 4, 6\}$ sont respectivement donnés par les figures 5.5, 5.6 et 5.7, en fonction de la dimension d

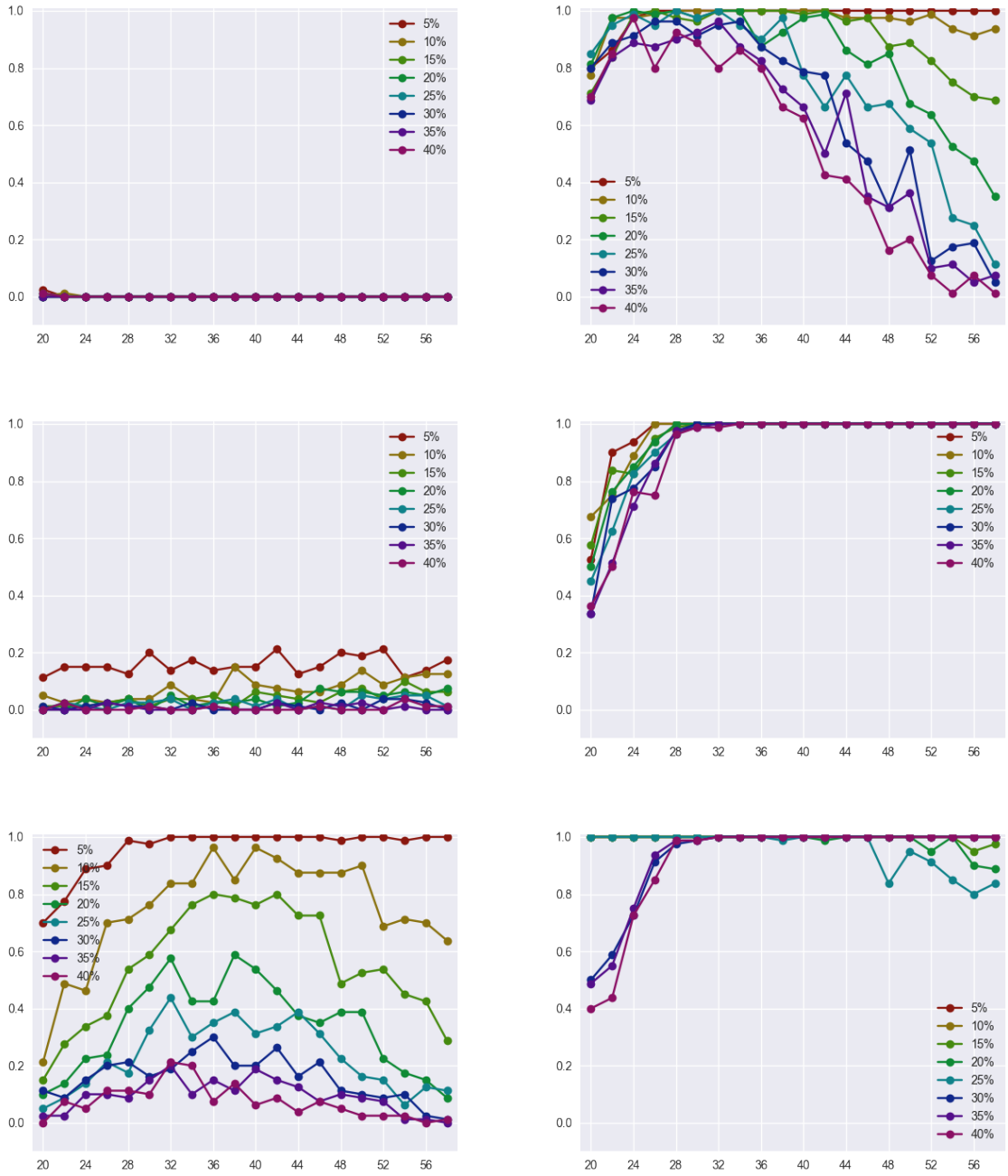


FIG. *5.5 :* Résultats de l'expérience pour $k = 2$, pour les algorithmes de Borgelt, EWKM et Prosecco (colonne de gauche) et leurs versions possibilistes (colonne de droite).

et du pourcentage de bruits ajouté aux données. Pour des raisons de lisibilité l'écart-type n'est pas représenté.

Sur chacune des trois figures, six graphes sont représentés, qui donnent l'évolution du ratio de bonne estimation de la dimensionnalité en fonction de la dimension d totale, pour différents paramètres de bruit. La colonne de gauche représente les résultats

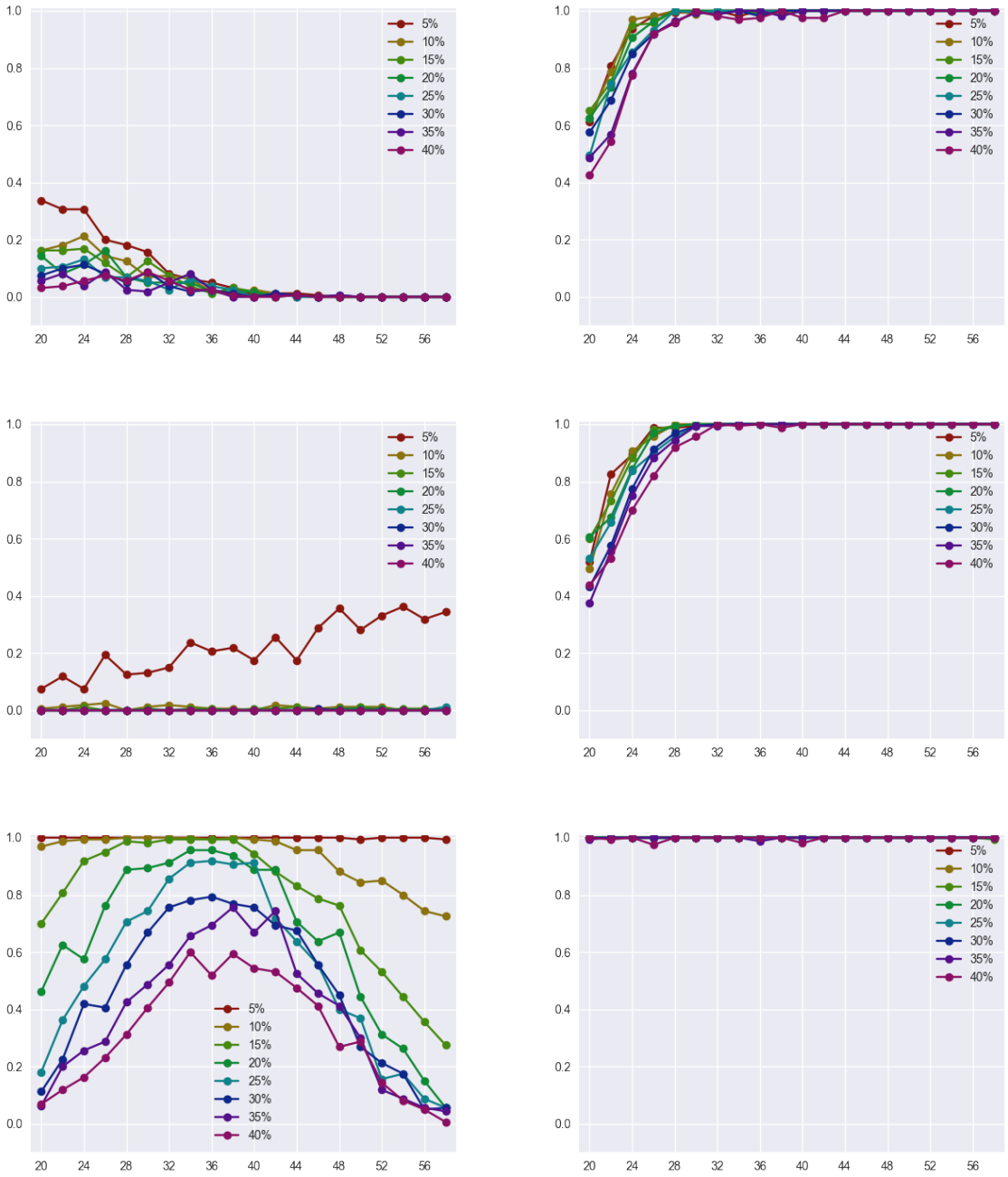


FIG. *5.6 :* Résultats de l'expérience pour $k = 4$, pour les algorithmes de Borgelt, EWKM et Prosecco (colonne de gauche) et leurs versions possibilistes (colonne de droite).

des algorithmes parcimonieux déjà étudiés dans le chapitre précédent, dans l'ordre suivant : algorithme de Borgelt en haut, EWKM au milieu et Prosecco en bas. La colonne de droite représente les résultats des versions possibilistes de ces algorithmes, dans le même ordre : algorithme PBorgelt en haut, PEWKM au milieu et Possecco en bas. Nous commençons par commenter les résultats des expériences pour $k = 2$ et $k = 4$.

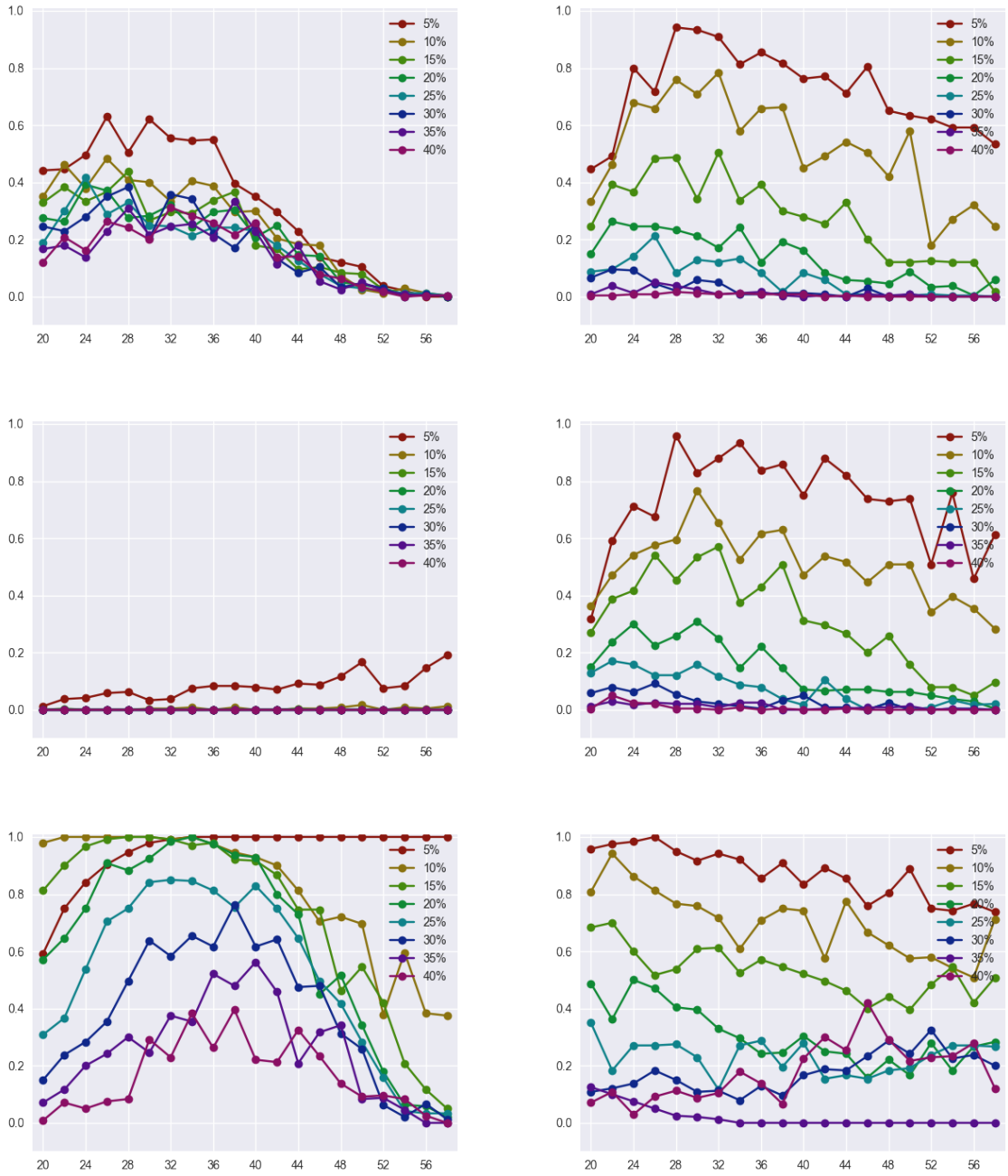


FIG. *5.7 :* Résultats de l'expérience pour $k = 6$, pour les algorithmes de Borgelt, EWKM et Prosecco (colonne de gauche) et leurs versions possibilistes (colonne de droite).

Algorithmes de Borgelt et EWKM

Sur les figures 5.5 et 5.6, on observe que les ratios des algorithmes de Borgelt et EWKM sont proches de 0, à plus forte raison quand le bruit dans les données augmente. Nous interprétons ainsi ces résultats : la présence de bruit dans les données, même faible, induit les algorithmes en erreur et les fait converger vers des solutions dans

lesquelles les points ne sont pas attribués aux bons clusters comme dans la figure 5.4b, ou alors les algorithmes ne retrouvent pas la dimensionnalité attendue.

Les versions possibilistes de ces algorithmes obtiennent de bien meilleurs résultats : les ratios sont très proches de 1, sauf dans le cas de PBorgelt lorsque $k = 2$. Comme dans l'expérience du chapitre 4, un mauvais paramétrage pourrait être la cause de ces résultats, qui tendent cependant à montrer que les deux algorithmes sont très sensibles au bruit dans les données et que l'opérateur possibiliste corrige ce défaut.

Algorithmes Prosecco et Possecco

L'algorithme Prosecco semble relativement plus résistant au bruit que les autres. Pour $k = 2$ et $k = 4$, les figures montrent un ratio supérieur aux algorithmes de Borgelt et EWKM, en particulier quand le bruit ajouté reste faible. Les courbes semblent présenter un maximum local, correspondant à une dimension d_1 particulière, différente pour chaque courbe, pour laquelle Prosecco obtient de meilleurs résultats. Ce phénomène peut s'explique par la façon dont la valeur de γ_W est sélectionnée : selon la valeur de d, Prosecco obtient peut-être de meilleurs résultats avec d'autres γ_W mais ceux-ci ne sont pas affichés. Cette expérience permet de montrer que, lorsque les données sont bruitées, le choix de la valeur de γ_W est plus crucial que lorsqu'elles ne le sont pas, comme dans le chapitre 4.

Comme pour les algorithmes précédents, l'adjonction de l'opérateur possibiliste permet de rendre l'algorithme plus résistant au bruit. Les deux derniers graphiques, situés en bas à droite des figures 5.5 et 5.6, montrent un ratio moyen très proche de 1 dans la plupart des situations, témoignant du fait que Possecco reconnaît la quasi-totalité des clusters et leur dimensionnalité même lorsque d varie ou que le bruit augmente.

Résultats pour $k = 6$

La figure 5.7 montre les résultats de l'expérience dans le cas où $k = 6$ et mène à des conclusions légèrement différentes. En effet, si les courbes de l'algorithme EWKM sont comparables à celles obtenues pour $k = 2$ et $k = 4$, la colonne de droite de la figure montre que les versions possibilistes sont beaucoup moins efficaces contre le bruit que dans les cas précédents, et ce d'autant plus que le bruit est accru. Les trois algorithmes possibilistes sont désormais équivalents, le ratio moyen de bonne identification étant inférieur à 0,5 pour 20% de bruit et plus.

En particulier, le graphe situé en bas à gauche de la figure montre que l'algorithme Prosecco obtient de meilleurs résultats que sa version possibiliste Possecco, quel que soit le niveau de bruit. Ceci peut s'expliquer par un mauvais choix du paramètre γ_U pour Possecco, mais aussi par le grand nombre de points générés pour ces expériences et l'augmentation du degré de liberté qui en découle pour former les solutions : pour un nombre élevé de points, le nombre de groupes denses au sein des données augmente et de nouveaux clusters peuvent être formés, qui peuvent donc induire l'algorithme en erreur.

De façon paradoxale, les algorithmes possibilistes sont sensibles à ce phénomène car ils accordent plus d'importance à ces groupes denses en masquant les autres points (Krishnapuram 1994), ce qui est le pendant de leur plus grande résistance au bruit et aux outliers. Un travail supplémentaire doit être fourni pour déterminer si ce phéno-

mène est accru dans le cas du subspace clustering, du fait de la sélection d'attributs et de l'utilisation de distances pondérées localement à chaque cluster.

Détermination des hyperparamètres

Ici encore, l'évaluation des algorithmes est supervisée et permet de sélectionner, en fonction des résultats, la valeur des hyperparamètres γ_W et β portant sur le terme inducteur de parcimonie, ainsi que celle de γ_U portant sur le terme possibiliste. Nous observons que, selon les algorithmes, les meilleures valeurs de γ_W ou β peuvent grandement varier en fonction du nombre de clusters cherchés k et du bruit dans les données. En revanche celle de γ_U varie uniquement d'un algorithme à l'autre, et non selon le bruit ou la dimension.

5.5 CONCLUSION

Dans ce chapitre, nous avons considéré un nouveau problème de minimisation avec une pénalité non différentiable qui relâche la contrainte de sommation à 1. Nous montrons que cette pénalité est proximable et que l'opérateur associé est une généralisation de la projection sur le simplexe unitaire utilisée dans le chapitre précédent. Celui-ci n'introduit pas de parcimonie dans les solutions mais il a pour effet d'autoriser la somme des composantes des vecteurs projetés à être strictement plus petite que 1.

Cet opérateur est appliqué aux variables W puis U, ce qui nous permet de dériver de nouveaux algorithmes de subspace clustering flou. Lorsqu'il est appliqué à U, nous montrons qu'il permet d'obtenir des algorithmes de subspace clustering possibiliste, qui produisent des degrés d'affectation U qui tiennent compte de l'importance d'un point au sein d'un cluster, ce qui permet à la fois d'obtenir une description plus riche des clusters formés en obtenant des degrés plus représentatifs, mais aussi de rendre ces algorithmes plus résistants au bruit. Nous montrons dans une expérience comment les 3 algorithmes parcimonieux étudiés dans le chapitre précédent deviennent plus résistants au bruit lorsqu'on leur ajoute ce nouvel opérateur.

Dans la pratique, les algorithmes possibiliste obtenus présentent certains défauts classiques des algorithmes de clustering possibiliste, comme par exemple le fait d'identifier des clusters trop proches, au détriment du reste des points. Des stratégies avancées, comme la fusion de clusters voisins (Krishnapuram 1994), pourraient être une solution à ce problème.

6

PRISE EN COMPTE DE LA GÉOMÉTRIE DES DONNÉES

Le paradigme de subspace clustering étudié dans cette thèse s'appuie sur une distance euclidienne pondérée localement à chaque cluster. Des points éloignés selon la distance euclidienne non pondérée peuvent se retrouver rapprochés du centre d'un cluster par la distance pondérée et être ainsi affectés ensemble. Inversement, des points voisins dans l'espace $\mathbb{R}^d$ peuvent être éloignés dans un sous-espace et se retrouver affectés à des clusters distincts.

Ceci fait apparaître une opposition entre les informations apportées par la distance non pondérée et les informations propres aux sous-espaces sur lesquelles repose le subspace clustering : en adaptant localement la pondération des attributs, de nouveaux clusters peuvent être formés, rassemblant des données qui ne se ressemblent que partiellement et qui sont donc séparées pour la distance non pondérée. Cette opposition entre informations globales et locales peut cependant être reformulée localement : l'affectation aux clusters ne tient pas compte des voisinages des points formés par la distance non pondérée. Ainsi, la localité des sous-espaces s'oppose à celle des voisinages, qui constitue une information supplémentaire que les algorithmes que nous étudions ne prennent pas en compte.

Dans ce chapitre, nous soulignons ce phénomène au moyen d'un exemple de données artificielles, qui illustre comment des points partageant certains attributs seulement se retrouvent affectés à un même cluster au détriment de points pourtant plus proches selon la distance non pondérée. Nous en proposons une interprétation géométrique : dans ce chapitre, nous nous intéressons aux voisinages des points, définis selon la distance euclidienne dite *globale* et nous illustrons le fait qu'ils ne sont pas pris en compte par les algorithmes lors du calcul de l'affectation des points aux clusters.

Afin de corriger ce problème, nous proposons d'introduire un terme de régularisation laplacienne dans la fonction de coût. Cette fonction de pénalité, exprimée sur les degrés d'affectation, est notamment au cœur des algorithmes de clustering spectral. Elle possède une interprétation géométrique : chaque point est relié à ses voisins, dans le sens où plus des points sont proches dans $\mathbb{R}^d$, plus cette fonction pénalise leur séparation, c'est-à-dire leur affectation à des clusters distincts.

Contrairement aux fonctions de coût proposées dans les chapitres précédents, celle que nous étudions ici est différentiable et peut être optimisée sous les contraintes usuelles. Nous dérivons un nouvel algorithme de subspace clustering flou, baptisé *Weighted Laplacian Fuzzy Clustering* (WLFC), dont nous montrons qu'il prend en compte les voisinages des points pour construire les clusters.

Nous proposons enfin une expérience utilisant des données artificielles générées selon des gaussiennes multivariées et s'appuyant sur des critères de qualité originaux, qui comparent le comportement des algorithmes WLFC et AWFCM, ainsi que l'algorithme PFSCM du chapitre précédent. Cette expérience met en avant certaines similarités entre les trois algorithmes, qui suggèrent que ces algorithmes basés sur des équa-

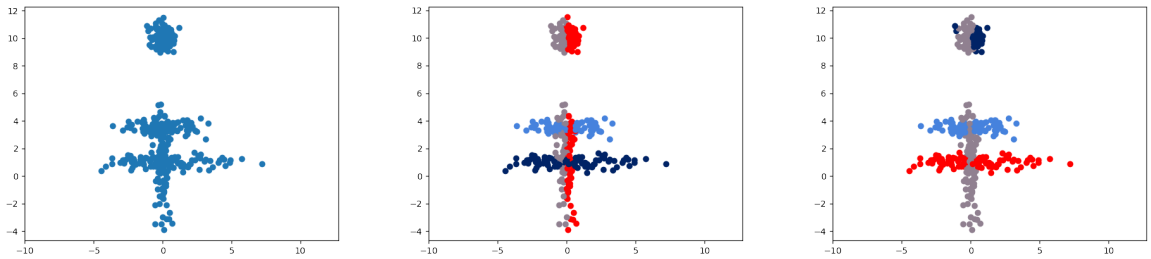


FIG. *6.1 :* Exemple de nuage de points (à gauche) conduisant à des clusters contre-intuitifs pour les algorithmes AWFCM (au centre) et Prosecco (à droite).

tions de mise à jour convergent vers des minima qui nuisent à leur représentation des données.

La section 6.1 présente l'exemple de données servant de fil rouge dans ce chapitre : sur un exemple simple, nous montrons que les algorithmes utilisés jusqu'à présent ne tiennent pas compte des informations de voisinage des données. Nous proposons un nouveau terme de régularisation en section 6.2. Ce terme étant différentiable, une nouvelle équation de mise à jour est dérivée et implémentée dans un nouvel algorithme, WLFC (*Weighted Laplacian Fuzzy Clustering*) dans la section 6.3, qui est étudié en section 6.4. L'algorithme WLFC a été présenté aux conférences FUZZIEEE 2017 et LFA 2018 (Guillon et al. 2017 ; Guillon et al. 2018b).

6.1 GÉOMÉTRIE DES DONNÉES ET VOISINAGES

Cette section illustre un défaut des algorithmes de subspace clustering étudiés jusqu'à présent : sur un jeu de données artificielles dans $\mathbb{R}^2$, les algorithmes AWFCM et Prosecco produisent des résultats contre-intuitifs. En effet, les minima vers lesquels ces algorithmes convergent produisent une affectation des points aux clusters qui ne tient pas compte des voisinages de ces points, au sens de la métrique de $\mathbb{R}^2$.

La figure 6.1 donne un exemple en deux dimensions de cette situation : le nuage de points est formé de points échantillonnés de quatre gaussiennes. Trois de ces gaussiennes sont essentiellement unidimensionnelles et sont partiellement confondues, alors que la quatrième est bidimensionnelle. Les points de la gaussienne restante sont situés au sommet de la figure et sont séparés des autres ; ils sont en outre dispersés également dans les deux dimensions, formant un cluster sphérique.

Au centre et à droite de la figure 6.1, les résultats des algorithmes AWFCM et Prosecco sont représentés, les algorithmes étant exécutés avec le paramètre $c = 4$. Ils correspondent à des minima typiques de ceux vers lesquels ces deux algorithmes convergent sur ce jeu de données. Les résultats des algorithmes représentés sur la figure peuvent être interprétés ainsi :

- AWFCM identifie quatre clusters unidimensionnels, deux horizontaux et deux verticaux. Ces deux derniers, rouge et gris, contiennent des points du nuage central ainsi que du nuage sphérique en haut de la figure. L'affectation des points aux clusters est donc calculée sans tenir compte de la séparation entre le cluster sphérique et les autres ;

- Prosecco, exécuté avec le paramètre $\gamma = 10$, identifie trois clusters unidimensionnels et un cluster quasi-sphérique. Cependant, comme pour AWFCM, le cluster vertical « grignote » les points du cluster du haut.

Ces observations s'expliquent facilement dans le paradigme de subspace clustering que nous étudions : l'utilisation de distances euclidiennes pondérées rapproche ici les points du cluster sphérique alignés du ou des clusters verticaux et ces points sont donc considérés comme plus proches de ceux du bas que de certains de leurs voisins directs. Ce résultat persiste d'ailleurs selon que l'on augmente ou que l'on diminue le nombre de clusters c cherchés, la gaussienne bidimensionnelle n'étant jamais clairement séparée des autres au sein des solutions.

Cependant, ce phénomène apparaît comme peu intuitif et contredit les objectifs du subspace clustering : si le subspace clustering a pour but d'utiliser des informations locales pour découvrir les clusters, pourquoi ne pas utiliser également les voisinages des points ou le fait que les deux nuages sont séparés pour construire ces clusters ?

6.2 PRISE EN COMPTE DES VOISINAGES DES POINTS

Cette section présente un nouveau terme de pénalité inspiré du clustering spectral qui fait intervenir une information basée sur la distance euclidienne de $\mathbb{R}^d$: cette information *globale* s'oppose à la recherche de distances euclidiennes pondérées *locales* aux sous-espaces, utilisées dans les algorithmes de subspace clustering que nous avons étudiés jusqu'à présent.

Plutôt que comme une contradiction, ce terme de pénalité est une tentative de tempérer l'enthousiasme du modèle de subspace clustering initial, comparable aux techniques de régularisation proposées pour l'algorithme de Gustafson-Kessel (section 2.3.5 p. 24) : nous présentons une nouvelle fonction de coût qui fait intervenir des informations locales, liées aux sous-espaces, mais aussi globales, liées à la géométrie de $\mathbb{R}^d$. Cet ajout permet de mettre en avant de nouvelles solutions de clustering, enrichissant donc l'expressivité du modèle initial.

Contrairement aux chapitres précédents, cette nouvelle fonction de coût peut être minimisée sous les contraintes usuelles. Elle est de plus différentiable en chacune de ses variables, ce qui permet de dériver des équations de mise à jour par la méthode usuelle des multiplicateurs de Lagrange.

6.2.1 *Terme de régularisation laplacienne*

Le terme de régularisation que nous proposons pénalise les variations entre des degrés d'affectation $U_{\cdot i}$ et $U_{\cdot j}$ pour deux points voisins X_i et X_j. Pour tenir compte de cette notion de voisinage, ce terme fait intervenir une relation de similarité $S \in \mathbb{R}^{+n \times n}$, définie dans la section 6.2.3 p. 92, qui apporte des informations sur la géométrie des points dans l'espace $\mathbb{R}^d$.

La fonction de pénalité est alors donnée par :

$$\begin{aligned} G(U) &= \sum_{i,j=1}^{n} \left\| U_{\cdot i} - U_{\cdot j} \right\|^2 \cdot s_{ij} \\ &= \sum_{i,j=1}^{n} \sum_{r=1}^{c} (u_{ri} - u_{rj})^2 \cdot s_{ij} \end{aligned} \tag{6.1}$$

Ce terme pénalise les solutions U qui affectent des points voisins, au sens de la similarité $S = (s_{ij})$, à des clusters (C_r) différents. En effet, étant donné deux points X_i et X_j, le coût induit est proportionnel à la similarité s_{ij}, c'est-à-dire à la proximité des points, ainsi qu'à la distance entre $U_{\cdot i}$ et $U_{\cdot j}$.

Ce terme de pénalité est central dans la famille des algorithmes de clustering spectral, brièvement évoqués dans la section 2.1.2 p. 10, qui consistent à former une matrice particulière à partir de S, appelée matrice laplacienne, puis à la diagonaliser pour obtenir une nouvelle représentation des données adaptée à la recherche des clusters. Les matrices laplaciennes sont également au cœur de plusieurs algorithmes de *manifold learning* (Belkin et Niyogi 2003 ; Huo et al. 2007).

En ajoutant ce terme de pénalité à la fonction de coût F_K de l'algorithme AWFCM, présenté en section 2.3.2 p. 20, nous proposons donc un algorithme hybride, qui combine une fonction de coût issue des approches spectrales à celle d'un algorithme par partitionnement, tout en conservant la stratégie de minimisation de ce dernier.

6.2.2 *Nouvelle fonction de coût*

En intégrant le terme de régularisation précédent à la fonction de coût, le problème de minimisation 3.1 devient, avec $\gamma > 0$,

$$J(C, U, W) = \sum_{i=1}^{n} \sum_{r=1}^{c} u_{ri}^2 \sum_{p=1}^{d} w_{rp}^2 (x_{ip} - c_{rp})^2 + \gamma \sum_{i,j=1}^{n} \sum_{r=1}^{c} (u_{ri} - u_{rj})^2 \cdot s_{ij}$$

$$\text{sous les contraintes} \quad \begin{aligned} &\forall i \in [\![1, n]\!], \ \sum_{r=1}^{c} u_{ri} = 1, \\ &\forall r \in [\![1, c]\!], \ \sum_{i=1}^{n} u_{ri} > 0, \\ &\forall p \in [\![1, d]\!], \ \sum_{r=1}^{c} w_{rp} = 1 \end{aligned} \tag{6.2}$$

Le paramètre γ équilibre les deux termes de la fonction de coût. Lorsque γ augmente, les différences entre les paires (u_{ri}, u_{rj}) sont pénalisées et les valeurs des (u_{ri}) s'uniformisent au sein des composantes connexes du graphe formé par la similarité S. Cela contraint des points voisins à appartenir exactement aux mêmes clusters, indépendamment de la valeur du premier terme.

6.2.3 *Choix d'une mesure de similarité*

L'équation 6.1 et le problème 6.2 font intervenir une pénalité portant sur la variable U qui présuppose une relation de similarité S définissant la proximité des (X_i).

Cette similarité ne dépend pas des poids W et sert à contrebalancer le premier terme, responsable de l'apprentissage des sous-espaces. La fonction de coût regroupe donc les effets de deux mesures de similarité : l'une locale, apprise au fur et à mesure que les sous-espaces sont découverts, l'autre globale et constante.

De même que pour des algorithmes de clustering standard, le choix de la mesure de similarité joue un rôle essentiel dans l'identification des clusters : elle pénalise les solutions ne respectant pas les voisinages des points dans $\mathbb{R}^d$ et doit donc être indépendante de la forme des clusters. Elle doit de plus s'adapter à des voisinages de densités variables.

Pour satisfaire ces attentes, nous faisons le choix d'un noyau gaussien classique, faisant intervenir la distance euclidienne de $\mathbb{R}^d$ entre les points X_i et X_j. Afin de prévenir des variations de densité au sein des données, nous utilisons le raffinement proposé par Zelnik-Manor et Perona (2004) qui adapte le paramètre de largeur de bande σ localement :

$$s_{ij}^0 = \exp\left(\frac{-\left\|X_i - X_j\right\|^2}{\sigma_i \sigma_j}\right)$$

où $\sigma_i = \left\|X_i - X_i^K\right\|$ est la distance euclidienne entre X_i et son K-ième plus proche voisin, X_i^K.

Comme le but de la similarité est de forcer l'algorithme de cluster à tenir compte des voisinages des points, que l'on souhaite affecter à un même cluster autant que possible, nous faisons également le choix d'une similarité parcimonieuse, en ajoutant un seuil au-delà duquel elle s'annule :

$$s_{ij} = \begin{cases} s_{ij}^0 & \text{si } \left\|X_i - X_j\right\|^2 \leq \sigma_i \sigma_j \\ 0 & \text{sinon} \end{cases} \tag{6.3}$$

Pour des valeurs de K petites devant le nombre de points n, la matrice S est creuse. Le terme de pénalité de l'équation 6.1 est donc nul sauf pour les paires (X_i, X_j) de points suffisamment proches, ce qui permet de limiter l'effet du terme de régularisation aux voisinages directs de chaque point. Enfin, on remarque que la matrice S est bien symétrique.

6.2.4 *Dérivation d'un terme de mise à jour*

Dans cette section nous dérivons un terme de mise à jour pour la variable U de façon à permettre l'optimisation de la fonction de coût J définie dans l'équation 6.2. Comme ni W ni C n'apparaissent dans le terme de régularisation, il n'est pas nécessaire de dériver de nouvelle équation de mise à jour pour ces variables. En restreignant le lagrangien du problème 3.1 à la seule variable U, on obtient :

$$\begin{aligned} \mathcal{L}(U) &= \sum_{r=1}^{c}\sum_{i=1}^{n} u_{ri}^2 d_{ri}^2 + \gamma \sum_{i,j=1}^{n}\sum_{r=1}^{c} (u_{ri} - u_{rj})^2 \cdot s_{ij} + \sum_{i=1}^{n}\left(\lambda_i\left(\sum_{r=1}^{c} u_{ri} - 1\right)\right) \\ &= J_K + \gamma J_S + J_{C1} \end{aligned}$$

Les trois termes de cette fonction peuvent être dérivés séparément. On a d'abord :

$$\frac{\partial J_K}{\partial u_{ri}} = 2u_{ri}d_{ri}^2$$
$$\frac{\partial J_{C1}}{\partial u_{ri}} = \lambda_i$$

De façon à isoler les termes qui dépendent de u_{ri}, J_S peut être réécrite de la façon suivante :

$$J_S = \sum_{\substack{k=1\\k\neq i}}^{n}\left(\sum_{\substack{j=1\\j\neq i}}^{n}\sum_{t=1}^{c}(u_{tk}-u_{tj})^2 s_{kj} + \sum_{t=1}^{c}(u_{tk}-u_{ti})^2 s_{ki}\right) + \sum_{\substack{j=1\\j\neq i}}^{n}\sum_{t=1}^{c}(u_{ti}-u_{tj})^2 s_{ij} + \sum_{t=1}^{c}(u_{ri}-u_{ti})^2 s_{ii}$$

En différenciant par rapport à u_{ri} et en tenant compte du fait que S est symétrique :

$$\begin{aligned}\frac{\partial J_S}{\partial u_{ri}} &= -2\sum_{\substack{k=1\\k\neq i}}^{n}(u_{rk}-u_{ri})s_{ki} + 2\sum_{\substack{j=1\\j\neq i}}^{n}(u_{ri}-u_{rj})s_{ij} + 0\\ &= 4\sum_{\substack{j=1\\j\neq i}}^{n}(u_{ri}-u_{rj})s_{ij}\end{aligned}$$

Avec $\frac{\partial \mathcal{L}}{\partial u_{ri}} = \frac{\partial J_K}{\partial u_{ri}} + \gamma\frac{\partial J_S}{\partial u_{ri}} + \frac{\partial J_{C1}}{\partial u_{ri}} = 0$, on obtient une équation de mise à jour pour u_{ri} :

$$u_{ri} = \frac{4\gamma\sum_{\substack{j=1\\j\neq i}}^{n} u_{rj}s_{ij} - \lambda_i}{2d_{ri}^2 + 4\gamma\sum_{\substack{j=1\\j\neq i}}^{n} s_{ij}} \tag{6.4}$$

Ce terme de mise à jour dépend toujours de λ_i, qui peut être calculé séparément. Avec $\frac{\partial \mathcal{L}}{\partial \lambda_i} = \sum_{s=1}^{c} u_{is} - 1 = 0$, en écrivant le dénominateur $E_{ri} = 2d_{ri}^2 + 4\gamma\sum_{j\neq i} s_{ij}$ on obtient, en sommant sur tous les r :

$$\sum_{r=1}^{c} u_{ri} = \sum_{r=1}^{c}\left(\frac{4\gamma\sum_{j\neq i} u_{rj}s_{ij}}{E_{ri}} - \frac{\lambda_i}{E_{ri}}\right) = 1$$

D'où l'équation de mise à jour pour λ_i :

$$\lambda_i = \frac{\sum_{r=1}^{c}\frac{4\gamma\sum_{j\neq i} u_{rj}s_{ij}}{E_{ri}} - 1}{\sum_{r=1}^{c}\frac{1}{E_{ri}}} \tag{6.5}$$

Algorithme 7 L'algorithme WLFC

procédure WLFC($X, S, c, \gamma, \varepsilon$)
  Initialisation : $U, C \leftarrow \text{FCM}(X, c, \varepsilon)$
    $\forall r, W_r \leftarrow \left[\frac{1}{d}, \cdots, \frac{1}{d}\right]$
  répéter
    Mettre à jour de λ d'après l'équation 6.5
    Mettre à jour U d'après l'équation 6.4
    Mettre à jour C d'après l'équation 2.11 p. 21
    Mettre à jour W d'après l'équation 2.13 p. 22
  jusqu'à *convergence*(C, U, W, ε)

Les équations de mise à jour 6.5 et 6.4 peuvent être utilisées directement dans un algorithme d'optimisation alternée, comme les équations classiques des algorithmes du style des c-moyennes floues. Elles présentent néanmoins une légère différence avec ces dernières : comme l'affectation d'un point dépend de celle de ses voisins, les équations 6.5 et 6.4 servant à mettre à jour la variable u_{ri} font apparaître les autres (u_{rj}), ce qui n'était pas le cas pour toutes les autres équations de mise à jour rencontrées jusqu'à présent.

6.3 UN ALGORITHME RESPECTUEUX DU VOISINAGE

L'étude mathématique précédente permet de dériver deux équations de mise à jour qui peuvent être utilisées conjointement aux équations de l'algorithme AWFCM, section 2.3.2 p. 20, pour obtenir un nouvel algorithme de subspace clustering flou fonctionnant par optimisation alternée, présenté en section 6.3.1. Le comportement de cet algorithme est illustré graphiquement en section 6.3.2.

6.3.1 *L'algorithme WLFC*

Contrairement aux chapitres précédents, la fonction de coût du problème 6.2 est différentiable en chaque variable, la mise à jour de chaque variable utilise donc une équation de mise à jour directe. L'algorithme 7 présente cet algorithme d'optimisation alternée, nommé *Weighted Laplacian Fuzzy Clustering*. Celui-ci dépend des mêmes paramètres que d'habitude, à savoir les données X, le nombre de clusters cherchés c, l'hyperparamètre $\gamma > 0$ et le paramètre de convergence $\varepsilon > 0$, ainsi que de la matrice de similarité S.

Les équations 6.5 et 6.4 servant à mettre à jour la variable u_{ri} font intervenir les degrés d'appartenance des autres variables, (u_{rj}). Cette particularité doit être prise en compte lors de l'implémentation de l'algorithme : le calcul de la variable U à l'itération $t+1$ dépend de sa valeur à l'itération t précédente, qui doit donc être conservée. Une telle approche rappelle la descente de gradient, qui met à jour U d'après ses valeurs précédentes et pas seulement d'après les valeurs des autres variables W et C comme c'est habituellement le cas en optimisation alternée.

Le paramètre γ est choisi par l'utilisateur en fonction des données de façon à équilibrer les deux termes de la fonction de coût. Les expériences que nous conduisons

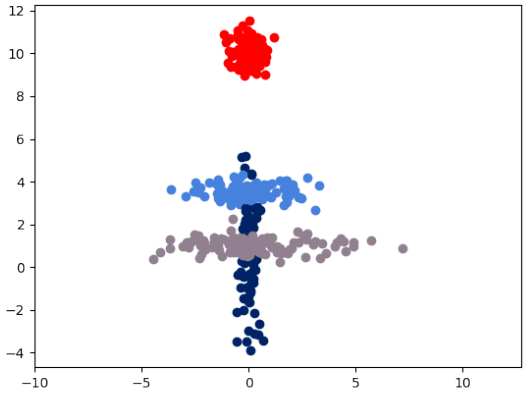


FIG. *6.2 :* Affectation des points produite par l'algorithme WLFC sur les données de la figure 6.1.

CLUSTER	POIDS	
	$w_{\Leftrightarrow}$	$w_{\Updownarrow}$
rouge	0,588	0,412
bleu clair	0,001	0,999
gris	0,001	0,999
bleu foncé	0,999	0,001

TAB. *6.1 :* Poids des dimensions calculés par l'algorithme WLFC sur les données de la figure 6.1.

montrent que le résultat de l'algorithme est assez sensible à la valeur de γ. Comme les clusters identifiés peuvent varier fortement lorsque γ augmente, il est conseillé de relancer l'algorithme plusieurs fois en changeant ce paramètre : différentes solutions correspondent à différents points de vue sur les données, les clusters formés pouvant mener à différentes interprétations.

La matrice de similarité S peut être pré-calculée par l'utilisateur de l'algorithme WLFC. Dans nos expériences, nous considérons uniquement la similarité définie dans la section 6.2.3, qui dépend d'un hyperparamètre supplémentaire $K > 0$, le nombre de voisins considérés pour calculer l'équation 6.3. Le paramètre K est généralement pris faible devant n de façon à ne considérer que des voisins directs de chaque point et à construire une matrice de similarité S creuse, permettant une implémentation plus efficace des équations de mise à jour de la section 6.2.4.

6.3.2 *Exemples illustratifs*

Nous proposons maintenant deux illustrations des résultats produits par WLFC.

Données bidimensionnelles

Le résultat de WLFC sur l'exemple de la section 6.1 est présenté en figure 6.2, pour les paramètres $\gamma = 0{,}5$, $K = 15$ et $c = 4$. La figure montre que le cluster vertical est correctement délimité et n'empiète pas sur le cluster sphérique. Le tableau 6.1 situé en-dessous présente les valeurs affectées aux deux dimensions pour chacun des clusters : les résultats montrent que trois clusters sont identifiés comme essentiellement unidimensionnels, alors que les poids du cluster du sommet de la figure sont tous les deux proches de 0,5, traduisant le fait que le cluster est identifié comme sphérique, c'est-à-dire bidimensionnel.

Bien que les poids calculés ici soient satisfaisants, nous observons expérimentalement que WLFC n'évalue pas toujours correctement la dimensionnalité des clusters, se comportant généralement comme AWFCM. L'étude expérimentale de la section suivante confirme cette observation. Notamment, à l'instar de l'algorithme AWFCM, WLFC produit des vecteurs de poids non parcimonieux ; l'application de la pénalité et de l'opérateur proximal du chapitre 4 à WLFC fait partie de nos perspectives.

Tores imbriqués

Nous proposons d'illustrer le comportement de WLFC par un deuxième exemple graphique, à partir de données constituées de deux tores imbriqués, parallèles à deux hyperplans perpendiculaires, représentées sur la figure 6.3. Comme le notent par exemple Allab et al. (2017), l'ajout d'un terme de régularisation laplacienne à un modèle prévu pour des données linéaire (dans leur cas, un modèle de factorisation de matrices combiné à l'analyse en composantes principales) permet d'identifier des structures non linéaires dans les données. De telles structures ne peuvent pas être décrites par les sous-espaces que nous considérions jusqu'à présent.

L'utilisation d'un terme de régularisation inspiré du clustering spectral est comparable à un algorithme de *manifold learning* « du pauvre », dont on peut tirer quelques jolies images. La figure 6.4 compare les résultats des algorithmes AWFCM et WLFC sur ces données. Les deux algorithmes produisent des résultats différents : sur l'image de gauche, on constate que des points du tore gris sont colorés en bleu, donc affectés majoritairement à l'autre cluster, et réciproquemment. Ceci témoigne du fait qu'AWFCM identifie les dimensions qui maximisent la similarité interne des deux clusters et affecte des points de chacun des tores à l'autre. WLFC calcule les mêmes poids pour les dimensions, mais l'utilisation d'une similarité basée sur la distance euclidienne globale pour estimer les voisinages des points dans $\mathbb{R}^d$ affecte les points des tores en respectant la structure des deux variétés.

6.4 VALIDATION EXPÉRIMENTALE

Cette section présente une étude expérimentale sur des données artificielles, à savoir des gaussiennes multivariées en faible et moyenne dimension et compare WLFC à AWFCM et l'algorithme PFSCM du chapitre précédent, présenté dans l'algorithme 4 p. 75. La nature hybride de WLFC, qui fait le pont entre différentes familles d'algorithmes de clustering comme évoqué dans la section 6.2.1, rend le choix des données expérimentales un peu délicat. Les gaussiennes multivariées utilisées ici avantagent

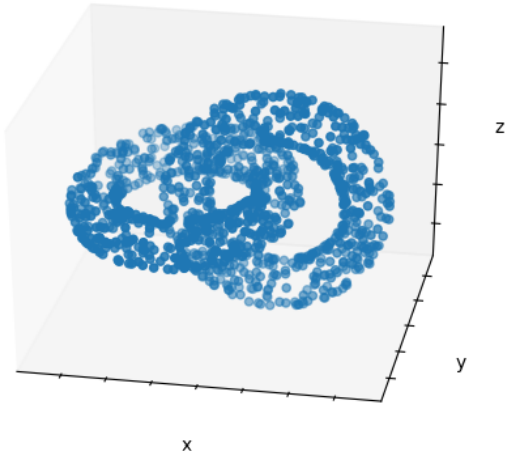


FIG. *6.3 :* Deux tores imbriqués, ainsi que le résultat des algorithmes AWFCM (au milieu) et WLFC (à droite).

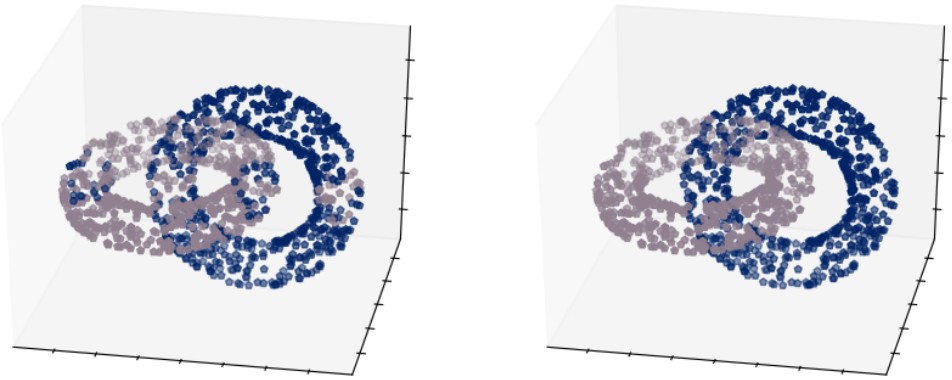


FIG. *6.4 :* Résultats des algorithmes AWFCM (à gauche) et WLFC (à droite) sur les données de la figure 6.3.

en effet considérablement les algorithmes de partitionnement basés sur des centroïdes et ne reflètent pas nécessairement au mieux l'apport de WLFC.

La section 6.4.1 détaille les données, les critères de qualité et le protocole expérimental utilisés dans cette expérience. Les résultats sont présentés dans la section 6.4.2.

6.4.1 *Protocole expérimental*

Données artificielles

Les données utilisées dans cette expérience sont une généralisation des données de la figure 6.1 en dimension supérieure $d \in \{5, 7, 9, 11, 13\}$. Pour chaque expérience, $\hat{c} = 4$ centres $\hat{C}_1, \ldots, \hat{C}_4$ sont générés aléatoirement avec une distance euclidienne minimum de 0,3 entre les centres. Ensuite, d_r dimensions $p_1, \ldots p_{d_r}$ sont tirées aléatoirement, avec d_r pris entre 1 et $d - 3$. Les dimensions $p_1, \ldots, p_{d_r}$ sont par la suite appelées dimensions pertinentes pour le cluster C_r. Pour chaque cluster, 100 points

d	WLFC	AWFCM	PFSCM
5	0.58 ± 0.09	1.18 ± 0.77	0.60 ± 0.11
7	0.72 ± 0.12	1.49 ± 1.03	0.73 ± 0.13
9	0.86 ± 0.14	1.98 ± 1.58	0.84 ± 0.13
11	0.99 ± 0.13	2.02 ± 1.36	0.98 ± 0.12
13	1.12 ± 0.11	1.77 ± 0.91	1.11 ± 0.10

d	WLFC	AWFCM	PFSCM
5	52%	51%	63%
7	53%	51%	68%
9	48%	46%	74%
11	48%	50%	78%
13	58%	56%	85%

TAB. *6.2 :* Résultats de l'expérience sur données artificielles : distances moyennes à gauche, pourcentages de sous-espaces correctement identifiés à droite.

sont générés selon une distribution gaussienne, avec une variance $v < 0{,}1$ pour les dimensions $p_1, ..., p_{d_r}$ et $v \in [0{,}5, 0{,}9]$ pour les autres dimensions.

Critères d'évaluation

Nous proposons de considérer deux critères pour évaluer la qualité des clusters retrouvés ainsi que les dimensions des sous-espaces identifiés : la distance aux centres générés, qui traduit la qualité des clusters obtenus par rapport aux données artificielles, ainsi que le pourcentage de sous-espaces correctement identifiés après une coupure explicite comparable à l'équeutage du chapitre 5, qui vise à caractériser les vecteurs de poids produits par les algorithmes.

Soit d'abord $\delta = \sum_{r=1}^{4} \|\hat{C}_r - C_r\|$ la somme des distances euclidiennes entre les centres générés $(\hat{C}_r)$ et obtenus (C_r) : elle constitue un critère de qualité standard pour évaluer les clusters produits. Une valeur faible signifie que les centres calculés sont proches des centres originaux.

Afin de proposer également un critère d'évaluation lié au subspace clustering, nous proposons un critère permettant de distinguer les dimensions retenues par un algorithme de celles qu'il considère comme non pertinentes. Comme aucun des trois algorithmes AWFCM, PFSCM et WLFC ne sélectionne explicitement les dimensions, nous proposons d'ajouter une phase de traitement des résultats qu'ils fournissent : étant donné la matrice W produite par un algorithme, on considère qu'une dimension p est jugée pertinente pour le cluster r si $w_{rp} > cut$, où cut est un réel choisi pour l'expérience. Nous estimons que le choix $cut = \frac{1}{2d}$ est une estimation pertinente de ce paramètre quand d croît. Le deuxième critère que nous choisissons, noté θ, est défini comme le pourcentage de clusters C_r pour lesquels toutes les dimensions retrouvées sont correctement identifiées par les algorithmes, c'est-à-dire le pourcentage de C_r tel que $\forall p, w_{rp} > cut \Leftrightarrow p \in \{p_1, ...p_{d_r}\}$,

Paramètre des algorithmes

L'algorithme AWFCM est initialisé avec des centres pré-calculés par l'algorithme des c-moyennes floues et utilise les paramètres $m = v = 2$ et $\alpha = 1$. PFSCM utilise le paramètre $\gamma = 1000$. Pour construire la relation de similarité de WLFC, on prend $K = 7$ voisins et on utilise de plus $\gamma = 5$. Enfin, les trois algorithmes utilisent le même critère de convergence, $\varepsilon = 10^{-4}$, et cherchent dans chaque cas $c = 4$ clusters.

6.4.2 *Résultats de l'expérience*

Le résultat de l'expérience pour $d \in \{5, 7, 9, 11, 13\}$ est présenté dans le tableau 6.2, ce choix de dimensions étant purement arbitraire. Les valeurs moyennes de δ sont données dans le tableau de gauche et le tableau de droite donne les valeurs de θ pour 100 exécutions de chaque algorithme.

Les faibles valeurs moyennes de δ pour WLFC et PFSCM montrent que les deux algorithmes identifient correctement les centres des clusters attendus alors que AWFCM est moins efficace. Le faible écart-type indique également une plus grande stabilité des deux premiers algorithmes. Les valeurs de θ montrent que WLFC se comporte de façon comparable à AWFCM en matière d'identification des dimensions de chaque cluster. L'algorithme PFSCM semble produire une meilleure estimation de ces dimensions. Ces conclusions sont toutefois tributaires du choix du paramètre *cut*, qui reste empirique.

Cette expérience montre que le terme de régularisation laplacienne, dont l'effet direct porte sur la variable U, a également un effet sur les clusters et les centres identifiés (variable C). En revanche, l'influence sur la variable W semble faible, l'algorithme WLFC présentant les mêmes caractéristiques que AWFCM.

L'expérience permet de caractériser les résultats de ces trois algorithmes : AWFCM produit des vecteurs de poids plus marqués que PFSCM, c'est-à-dire que les dimensions majoritaires reçoivent des poids plus élevés. WLFC, qui reprend les équations de mise à jour des centres et des poids de AWFCM, semble plus efficace du point de vue des clusters identifiés mais calcule des poids comparables à ceux d'AWFCM. PFSCM, enfin, qui s'appuie sur le schéma de mise à jour du chapitre 3, semble produire des résultats plus stables.

L'algorithme WLFC est, d'après notre expérience, sensiblement plus lent qu'AWFCM et PFSCM. La mise à jour de la variable U fait intervenir la matrice $S \in \mathbb{R}^{n \times n}$, dont la construction et le parcours peuvent prendre du temps. Cependant, l'utilisation de matrices creuses permet de conserver des performances et une occupation mémoire correctes, le facteur de remplissage de la matrice dépendant de K et des données.

6.5 CONCLUSION

Dans ce chapitre, nous faisons apparaître une opposition entre le principe du subspace clustering, qui sélectionne des attributs de façon à identifier des groupes denses, et le principe de similarité interne du clustering, qui dit que les groupes formés doivent contenir des point tous similaires. En intégrant à la fonction de coût 6.2 une relation de similarité basée sur la distance euclidienne non pondérée de $\mathbb{R}^d$, il est possible d'équilibrer ces deux principes et de faire apparaître de nouvelles solutions.

Contrairement aux fonctions de coût des chapitres précédents, cette fonction est différentiable et un algorithme peut être obtenu par la méthode usuelle des multiplicateurs de Lagrange. L'algorithme obtenu, nommé WLFC pour *Weighted Laplacian Fuzzy Clustering*, est paramétré par une relation de similarité qui fait intervenir la distance euclidienne standard et affecte les points à des clusters en tenant compte de leurs voisinages dans l'espace $\mathbb{R}^d$.

Une expérience sur des nuages de points constitués de gaussiennes multivariées permet d'évaluer l'influence de ce terme de pénalité : comparé à l'algorithme AWFCM,

WLFC retrouve des centres plus proches des centres originaux tout en calculant des poids similaires pour les dimensions. Les données considérées ne permettent cependant pas de mettre en avant les particularités de WLFC.

Cependant, d'autres utilisations sont envisageables, qui pourraient se révéler plus prometteuses. Premièrement, l'utilisation d'une relation de similarité liée à la géométrie de $\mathbb{R}^d$ conjointement à la distance euclidienne pondérée pourrait rapprocher WLFC des algorithmes de *manifold learning* (Huo et al. 2007), qui retrouvent des structures complexes dans les données et apportent des informations supplémentaires sur leur organisation intrinsèque, comme l'illustre l'expérience préliminaire sur les tores imbriqués. D'autre part, il semble également intéressant de se pencher sur l'utilisation de relations de similarité qui ne dériveraient pas de la distance euclidienne, apportant ainsi des informations complémentaires à la formation des clusters.

7

CONCLUSION ET PERSPECTIVES

Nous achevons ce manuscrit par une synthèse des différentes contributions, avant d'ouvrir sur une liste de perspectives de recherche qu'elles ouvrent.

7.1 RÉSUMÉ DES CONTRIBUTIONS DE LA THÈSE

Cette thèse étudie un modèle de subspace clustering flou, basé sur l'utilisation de distances euclidiennes pondérées, qui forme des clusters en identifiant les dimensions les plus pertinentes pour caractériser les données qu'ils contiennent. Plusieurs propriétés souhaitables des solutions sont formulées, qui sont absentes du modèle original : la parcimonie de la représentation des sous-espaces, qui permet d'estimer la dimensionnalité des clusters découverts ; la prise en compte et la description de la représentativité des données au sein d'un cluster, qui renforce la résistance de l'algorithme de clustering aux points marginaux ; enfin, le respect de la géométrie originale des nuages de points, qui permet de préserver des informations locales, utiles dans la découverte des clusters.

Afin de formaliser ces propriétés, nous les exprimons dans le vocabulaire de l'optimisation mathématique d'une fonction de coût, dont les minima correspondent aux solutions produites par l'algorithme et présentent un compromis entre les propriétés précédentes et les minima du modèle original. Nous proposons ainsi d'ajouter différents termes de pénalité qui changent les minima de la fonction de coût ; certains de ces termes de pénalité étant non différentiables, nous nous ancrons dans le paradigme de l'optimisation alternée, qui est la technique d'optimisation de référence en clustering flou, de sorte à utiliser des outils issus de la théorie de l'optimisation convexe. Nous introduisons un schéma d'optimisation enrichi qui intègre une étape de descente de gradient proximale. Ce schéma est par ailleurs générique : nous formulons une suite d'hypothèses relativement peu contraignantes sur les termes de pénalité qui peuvent être ajoutés à la fonction de coût. À notre connaissance, de tels outils sont encore nouveau dans la communauté du clustering flou.

En nous appuyant sur ce schéma, nous proposons la démarche suivante : pour implémenter les propriétés évoquées précédemment, nous repartons de la fonction de coût du modèle initial et introduisons des termes de pénalité qui les traduisent dans de nouvelles fonctions. L'étude de ces termes de pénalité et la prise en compte des contraintes propres au problème du subspace clustering flou permet à chaque fois de dériver des opérateurs ou équations de mise à jour qui sont implémentées dans de nouveaux algorithmes. Leurs particularités sont alors illustrées à l'aide de données pour cela ; nous les comparons enfin à différents algorithmes de l'état de l'art.

Le premier terme de pénalité proposé restreint les vecteurs de poids des dimensions à un domaine particulier tout en privilégiant les vecteurs parcimonieux. L'opérateur proximal correspondant à cette fonction de pénalité est donné sous forme algorithmique. Une preuve de correction de cet algorithme est proposée, qui s'appuie sur les

propriétés géométriques de l'espace des solutions. En s'appuyant sur le schéma d'optimisation précédent, un nouvel algorithme de subspace clustering flou est proposé, baptisé Prosecco. Celui-ci produit des descriptions parcimonieuses des sous-espaces découverts et nous montrons, en le comparant aux deux principaux algorithmes de l'état de l'art, qu'il est capable d'estimer la dimensionnalité de clusters et de sous-espaces simples. Enfin nous illustrons la pratique du subspace clustering en utilisant l'algorithme Prosecco sur des données réelles, qui nous permettent de faire apparaître ses différences avec des algorithmes classiques comme celui des k-moyennes.

Nous nous intéressons ensuite au problème du clustering possibiliste, qui vise à produire, en plus de la solution de clustering flou, une description de l'importance et de la représentativité de chaque point des données au sein de chaque cluster. Pour cela, nous proposons un deuxième terme de pénalité, qui relâche le précédent en élargissant le domaine des solutions cherchées. Nous l'appliquons successivement aux vecteurs de poids des sous-espaces et aux vecteurs d'appartenance des points aux clusters. Selon la valeur de l'hyperparamètre chargé de moduler l'influence de ce terme de pénalité, les vecteurs pénalisés sont projetés plus ou moins près du domaine précédent et ne somment plus nécessairement à 1. Nous dérivons un nouvel opérateur proximal, dont nous montrons expérimentalement qu'il ajoute un effet « possibiliste » aux algorithmes de clustering : les points représentatifs d'un cluster sont désormais différenciés des points périphériques, car ils reçoivent des valeurs significativement différentes. Nous intégrons cet opérateur à différents algorithmes parcimonieux pour en faire des versions possibilistes ; nous proposons notamment une variante possibiliste de l'algorithme Prosecco, baptisée Possecco. Nous montrons alors sur des données artificielles que les algorithmes possibilistes sont plus résistants aux données bruitées : l'opérateur possibiliste permet d'identifier les données qui ne font partie d'aucun cluster et de ne pas en tenir compte dans le calcul de la solution.

Enfin, le troisième terme de pénalité que nous proposons est un terme de régularisation laplacienne, qui s'inspire de la théorie spectrale des matrices d'adjacence des graphes et des algorithmes qui en sont issus, comme le clustering spectral. En étudiant un exemple particulier de données, nous montrons que les algorithmes de subspace clustering précédents ne tiennent pas compte du voisinage original des points lors de leur affectation aux clusters. La régularisation laplacienne nous permet de contraindre les solutions à respecter la géométrie originale des données : ce nouveau terme de pénalité contre-balance le terme de subspace clustering et ne dépend pas des vecteurs de poids appris. Moyennant le choix d'un hyperparamètre adapté, il équilibre l'influence des deux termes et permet de découvrir de nouvelles solutions.

Afin de conduire toutes nos expériences, nous avons implémenté tous les algorithmes concernés dans une même bibliothèque, qui s'appuie sur les structures de données proposées par NumPy (Walt et al. 2011). Cette bibliothèque vise à proposer un cadre générique pour implémenter les algorithmes de l'état de l'art et les nôtres, ainsi qu'à faciliter le développement de nouveaux algorithmes combinant des caractéristiques déjà implémentées, comme dans le cas de l'algorithme Possecco. Notre bibliothèque vise également à faciliter la comparaison entre les algorithmes implémentés et la visualisation des résultats.

Bien que notre travail porte sur la tâche du subspace clustering flou, le cadre dans lequel nous nous plaçons, celui de l'optimisation, est très général en apprentissage au-

tomatique. Cette thèse est donc à la fois une application de techniques d'optimisation convexe au problème du clustering et une illustration de l'intérêt de ce cadre technique, qui l'illustre et aide à en développer l'intuition.

7.2 PERSPECTIVES

Cette section présente quelques perspectives de recherche ouvertes par nos travaux en omettant les questions classiques telles que le choix du nombre de clusters, de l'utilisation de noyaux pour enrichir l'expressivité de nos algorithmes ou l'expérimentation sur des données réelles, ainsi que les perspectives déjà évoquées dans les conclusions des chapitres. Nous discutons ici de 6 perspectives originales, théoriques, logicielles ou expérimentales, qui concernent aussi bien le code de la bibliothèque écrite pendant la thèse que des perspectives d'apprentissage automatique plus générales.

7.2.1 *Preuve de la convergence de l'algorithme général PSFCM*

Comme évoqué en conclusion du chapitre 3, si elle a pu être observée expérimentalement dans les chapitres ultérieurs, la convergence de l'algorithme générique PSFCM n'a théoriquement pas été prouvée. Le fait d'alterner descente de gradient et descente selon un opérateur proximal est pourtant contre-intuitif, en particulier lorsque la pénalité ajoutée à la fonction de coût n'est pas convexe.

Les preuves de convergence des algorithmes d'optimisation alternée tels que les c-moyennes floues ou l'algorithme FSC présentés en section 2.3.3 p. 22 reposent généralement sur l'utilisation du théorème de Zangwill (1969). Les hypothèses de ce théorème semblent cependant trop fortes pour notre cadre de travail : il requiert notamment l'existence d'une fonction de coût continue, interdisant par exemple l'utilisation de la pénalité introduite au chapitre 4.

Les garanties d'un tel théorème de convergence sont essentiellement théoriques. En effet, les solutions des algorithmes de clustering par optimisation alternée ne sont pas nécessairement des minima locaux : Bezdek et al. (1987) donnent un exemple de situation dans lequel l'algorithme des c-moyennes floues converge vers un maximum local. Tous les algorithmes que nous présentons sont bien sûr susceptibles de présenter un comportement similaire.

Dans la pratique, il suffit d'implémenter ces algorithmes en bornant leur nombre maximum d'itérations pour s'assurer de leur terminaison, une pratique aussi connue sous le nom de « fuel » en preuve de programmes (Murphy et al. 2017). Sur le plan théorique, cependant, une telle preuve de convergence permettrait de mieux délimiter les fonctions de pénalité utilisables dans le cadre de travail que nous proposons. Huard (1979) présente des généralisations du théorème de Zangwill (1969) qui pourraient s'appliquer, en restreignant par exemple les hypothèses à la semi-continuité.

7.2.2 *Propriétés mathématiques et tests logiciels automatisés*

Les algorithmes de subspace clustering flou que nous implémentons et évaluons sont dérivés de fonctions de coût, dans le sens suivant : les équations et opérateurs

proximaux qu'ils font intervenir minimisent la fonction de coût étape par étape, selon des principes mathématiques tels que la descente du gradient.

L'implémentation logicielle de ces équations, opérateurs et algorithmes est cependant indifférente à ces principes mathématiques. Elle est par ailleurs en proie aux écueils du développement logiciels habituellement dénoncés par les partisans de la preuve de programme, qui sont renforcés, à notre sens, par la nature abstraite du code à implémenter : lorsqu'il traduit une équation dans le langage Python (ou autre), tout programmeur peut commettre une erreur, qui se traduira tôt ou tard par un bug logiciel.

Dans notre cas, une part importante de ces bugs pourraient être détectés automatiquement au moyen de tests générés à partir des principes mathématiques que nous évoquions. Des contraintes simples, comme la sommation à 1 ou la décroissance de la fonction de coût, peuvent être traduits à leur tour dans le code sous différentes formes.

Selsam et al. (2017) proposent de dériver des algorithmes d'apprentissage automatique dans le cadre confortable d'un assistant de preuves, obtenant ainsi des preuves de correction de leur implémentation relativement aux spécifications de ces algorithmes. Dans le cadre de nos algorithmes de clustering, de telles spécifications sont relativement faciles à écrire : par exemple, chaque itération doit faire décroître la valeur de la fonction de coût ; autre exemple, dans le cas d'une fonction triconvexe (convexe en chacun de ses trois arguments, les deux autres étant fixés, voir Gorski et al. (2007)) comme la fonction F du problème 2.10 p. 21, chaque équation de mise à jour conduit à un minimum de la fonction de coût.

Sans aller jusqu'à redévelopper notre bibliothèque à l'aide d'un assistant de preuves, il serait intéressant de fournir un moyen de traduire automatiquement les propriétés précédentes en tests unitaires, en utilisant par exemple une bibliothèque de test de propriétés comme la bibliothèque Hypothesis (MacIver 2018). Une telle politique de tests permettrait également de s'assurer automatiquement que certains cas « pathologiques » de données sont bien pris en compte, comme par exemple lorsque les centres des clusters sont confondus avec certains points, célèbre piège de l'algorithme des c-moyennes floues (voir équation 2.7 p. 13) mais qui n'arrive que très rarement en pratique.

7.2.3 *Énumération de clusterings alternatifs*

Les premiers algorithmes de subspace clustering tels que CLIQUE (section 2.2.1 p. 16) renvoient à l'utilisateur un ensemble de partitions possibles, qui contient toutes les solutions maximales, i.e. celles qui ne seraient pas subsumées par une autre. Les algorithmes de subspace clustering flou que nous étudions dans cette thèse, quant à eux, ne produisent qu'une seule solution, qui correspond à un partitionnement et à une estimation des sous-espaces donnés.

Ces modèles de subspace clustering ne sont pas convexes. Selon les données d'entrée, les fonctions de coût de ces algorithmes peuvent posséder plusieurs minima, qui correspondent à différents partitionnements des données, voire à l'identification de sous-espaces très différents. Ces minima sont autant de présentations différentes des données, et même des minima locaux peuvent intéresser l'utilisateur et lui apporter davantage d'informations sur les données. Les pénalités que nous étudions augmentent

potentiellement le nombre de minima locaux et donc de résultats possible ; c'est par exemple le cas pour les algorithmes possibilistes ou pour l'algorithme WLFC du chapitre 6.

En clustering classique, il est courant de relancer un algorithme de clustering non déterministe tel que les k-moyennes plusieurs fois sur les mêmes données pour éviter les minima locaux et chercher la meilleure solution. Ici, la problématique est différente : il s'agit non pas de chercher le « meilleur minimum », mais d'en retrouver plusieurs. Cependant, tous ces minima ne sont pas nécessairement accessibles depuis des initialisations aléatoires. Est-il possible de repartir d'une solution de subspace clustering pour en découvrir d'autres ? Des techniques de recherche des points-selles pourraient être adaptées, telles que la *gentlest ascent dynamics* (Weinan et Zhou 2011 ; Gu et Zhou 2018), qui permettrait de partir d'un minimum pour remonter vers un point-selle voisin et redescendre ensuite vers un autre minimum. D'autres travaux récents de physique théorique développent le concept d'*energy landscapes* (Ballard et al. 2017) qui consistent à étudier l'hypersurface formée par la fonction de coût pour en chercher toutes les vallées.

7.2.4 *Évolution des solutions en fonction des hyperparamètres*

Les trois pénalités que nous proposons dépendent d'un hyperparamètre γ qui a une forte influence sur les résultats. Ainsi, certaines « structures » au sein des données ne sont observables que pour des plages de valeur relativement étroites. C'est le cas par exemple des données artificielles de l'expérience du chapitre 4 : pour des valeurs de γ trop petites ou trop grandes, les algorithmes de subspace clustering flou que nous considérons ne retrouvent pas la dimensionnalité des hyperplans générés, parce qu'ils la surestiment pour des valeurs de γ faibles, ou parce qu'ils la sous-estiment sinon. De même dans le cas du chapitre 6, dans lequel certains clusters ne sont formés par l'algorithme que pour des plages de valeurs de γ.

Ici encore, on s'intéresse donc à la possibilité de relancer un algorithme de subspace clustering flou pour qu'il extraie des informations supplémentaires des données. En faisant varier γ incrémentalement, on pourrait alors idéalement produire une liste des clusters ou formes particulières qui apparaissent et disparaissent au fur et à mesure. Une telle liste pourrait avoir plusieurs applications : aider à la visualisation des données en construisant des projections qui ont du sens du point de vue des données, choisir des représentants archétypiques (Cutler et Breiman 1994) des clusters, c'est-à-dire qui représentent tous le groupe mais diffèrent fortement par d'autres aspects, ou encore montrer que certains clusters obéissent à des organisations internes complexes. Une telle recherche rappelle les problématiques d'analyse de données topologique (Munch 2017), qui fait varier un paramètre pour retrouver des structures topologiques caractéristiques dans les données et comprendre la structure de ces dernières.

La nature non supervisée du clustering fait qu'il est difficile d'évaluer la pertinence de ces clusters particuliers quand ils apparaissent. Un algorithme qui ne les identifie pas n'est pas particulièrement fautif ; d'autre part, il n'est généralement pas possible de déterminer que ces structures n'ont pas été détectées au sein des données alors qu'elles auraient dû l'être. Par ailleurs, selon les algorithmes utilisés, ces structures

connaissent des évolutions complètement différentes lorsque γ augmente : dans le cas d'une pénalité inductrice de parcimonie, les structures identifiées par l'algorithme perdent en dimensionnalité, alors que dans le cas possibiliste, il existe un γ maximum au-delà duquel tous les points sont totalement affectés à tous les clusters.

Rodríguez et de Carvalho (2017) proposent une stratégie non supervisée intéressante pour le choix du ou des hyperparamètres : pour des données fixées, ils relancent l'algorithme en faisant varier γ et prennent une valeur pour laquelle la matrice C produite (ou plus généralement toutes les variables) sont stables, c'est-à-dire qu'ils ne changent pas de valeur pour des petites variations de γ. Si cette approche remplit un objectif différent de celui qui nous intéresse ici, elle pourrait néanmoins constituer un point de départ.

7.2.5 *Visualisation et étude des algorithmes en haute dimension*

Bien que le subspace clustering ait été initialement formulé dans le contexte de la fouille de données en haute dimension, notre travail n'est pas particulièrement orienté vers ces problématiques. En effet, l'intuition que nous développons de nos algorithmes se fonde sur des exemples en deux ou trois dimensions, qui sont choisis pour leur capacité à être visualisés ; des expériences les généralisent dans des dimensions plus élevées mais ne prennent pas en compte les phénomènes géométriques et statistiques qui apparaissent lorsque la dimension augmente.

La haute dimension complique également la visualisation des résultats des algorithmes de subspace clustering flou. À notre connaissance, la littérature est étonnamment pauvre sur le sujet ; principale référence dans le domaine, Achtert et al. (2007) proposent de construire un graphe hiérarchisé, représentant les clusters trouvés en fonction de leur dimensionnalité ainsi que les inclusions possibles ; par exemple, deux clusters plans sécants forment un nouveau cluster de dimensionnalité 1. Leur méthode de visualisation ne produit cependant pas d'autre information sur les clusters obtenus, les points ou les centres des clusters ne faisant même pas partie du résultat comme c'était par exemple le cas dans l'expérience du chapitre 4, où nous visualisions le résultat des algorithmes à l'aide de l'algorithme MDS.

Hormis ces travaux, l'approche classique telle que proposée par exemple par Borgelt (2010) consiste à visualiser les clusters après projection dans les sous-espaces formés des dimensions les plus pertinentes. Cette approche n'étant bien sûr pas capable de représenter des données un peu complexes, nous pensons que des travaux supplémentaires doivent être conduits, qui adapteraient des méthodes modernes de réduction de la dimensionnalité et de visualisation comme T-SNE (Maaten et Hinton 2008) ou UMAP (McInnes et Healy 2018).

Enfin, lorsqu'une meilleure méthode d'évaluation et de visualisation des résultats aura été proposée, il conviendra également de généraliser notre travail en remplaçant non plus les termes de pénalité, mais le modèle de base des c-moyennes floues pondérées. En effet, celui-ci est peu adapté aux problèmes en haute dimension : Klawonn (2013) montre ainsi que l'algorithme des c-moyennes floues tend à proposer une solution triviale, qui place tous les centres au milieu des données et affecte les points également à chaque cluster. Ceci est dû à un minimum local de la fonction de coût des c-moyennes floues qui se forme en haute dimension : dans ces conditions, les centres

tendent à converger. Bien que l'utilisation de vecteurs de poids permette aux algorithmes de subspace clustering d'éliminer les dimensions superflues, la sensibilité des c-moyennes floues à ces phénomènes complique le travail des algorithmes de subspace clustering qui en dérivent.

7.2.6 *Comparaison généralisée des algorithmes à base de descente de gradient*

L'une des particularités des algorithmes que nous proposons dans les chapitres 4 et 5 est qu'ils reposent sur une procédure de descente *approximative* dans le sens où elle n'atteint pas le minimum de la fonction $F(V)$ qu'elle minimise, contrairement aux équations de mise à jour utilisées par les autres algorithmes. Nous avons observé dans le chapitre 5 que l'algorithme PFSCM, qui découle d'une fonction de coût égale à celle de l'algorithme AWFCM, n'identifie pas pour autant les mêmes minima : bien que les deux approches, respectivement basées sur la descente de gradient et sur des équations de mise à jour dérivées du lagrangien, soient toutes les deux basées sur les dérivées partielles de la fonction de coût, elles produisent des résultats différents.

D'un côté, les algorithmes basés sur des équations de mise à jour convergent plus rapidement et permettent de s'affranchir de certaines questions, telles que le choix du pas de descente. De l'autre, la technique des multiplicateurs de Lagrange est moins expressive, car elle ne permet pas systématiquement d'isoler la variable que l'on veut mettre à jour. Dans le chapitre 6 par exemple, l'équation de mise à jour ne peut pas, à notre connaissance, être formée sans avoir pris $m = 2$ dans la fonction de coût initiale. Wu et al. (2005) font une remarque similaire au sujet de la fonction de coût qu'ils proposent. Les approches basées sur la descente du gradient n'ont pas ce problème.

Les méthodes à base de gradient font l'objet d'une vaste littérature, plus riche que celle des algorithmes à base d'équations de mise à jour. Nous pensons qu'il serait intéressant de comparer les versions des algorithmes de clustering flou basées sur des équations de mise à jour à celles qui se basent sur des algorithmes de descente du gradient, en maintenant par exemple les contraintes à l'aide de la technique de projection utilisée dans cette thèse : si ces dernières se révèlent suffisamment efficaces, elles permettraient de généraliser les fonctions de coût et algorithmes utilisés en clustering flou.

PUBLICATIONS

Nos travaux de thèse ont conduit aux publications suivantes :

Guillon, A., Lesot, M.-J. et Marsala, C. (2016a). Optimisation proximale pour le subspace clustering flou. In : *25^{e} Rencontres francophones sur la Logique Floue et ses Applications*. Prix du meilleur article étudiant.

– (2017). Laplacian regularization for fuzzy subspace clustering. In : *Proceedings of the IEEE International Conference on Fuzzy Systems*. IEEE, p. 1-6.

– (2018a). A proximal framework for fuzzy subspace clustering. In : *Fuzzy Sets and Systems*. Elsevier, In Press.

– (2018b). Régularisation laplacienne pour le subspace clustering flou. In : *27^{e} Rencontres francophones sur la Logique Floue et ses Applications*.

Guillon, A., Lesot, M.-J., Marsala, C. et Pal, N. R. (2016b). Proximal optimization for fuzzy subspace clustering. In : *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, p. 675-686.

BIBLIOGRAPHIE

Achtert, E., Böhm, C., Kriegel, H.-P., Kröger, P., Müller-Gorman, I. et Zimek, A. (2007). Detection and visualization of subspace cluster hierarchies. In : *International Conference on Database Systems for Advanced Applications*. Springer, p. 152-163 (cf. p. 108).

Aggarwal, C. C., Wolf, J. L., Yu, P. S., Procopiuc, C. et Park, J. S. (1999). Fast algorithms for projected clustering. In : *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA : ACM, p. 61-72 (cf. p. vi, 18).

Aggarwal, C. C. et Yu, P. S. (2000). Finding generalized projected clusters in high dimensional spaces. In : *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA : ACM, p. 70-81 (cf. p. 19).

Agrawal, R., Gehrke, J., Gunopulos, D. et Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. In : *Proc. of the 1998 ACM SIGMOD Int. Conf. on Management of Data*. ACM, p. 94-105 (cf. p. vi, 3, 4, 16).

Agrawal, R. et Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In : *Proceedings of the 20th International Conference on Very Large Data Bases*. VLDB. Morgan Kaufmann Publishers Inc., p. 487-499 (cf. p. 17).

Allab, K., Labiod, L. et Nadif, M. (2017). A semi-NMF-PCA unified framework for data clustering. In : *IEEE Transactions on Knowledge and Data Engineering*. IEEE, 29.1, p. 2-16 (cf. p. 97).

Arabie, P. et Hubert, L. (1996). Advances in cluster analysis relevant to marketing research. In : *From Data to Knowledge*. Springer, p. 3-19 (cf. p. 4).

Arthur, D. et Vassilvitskii, S. (2007). k-means++ : The advantages of careful seeding. In : *Proceedings of the 18th annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial et Applied Mathematics, p. 1027-1035 (cf. p. 12).

Bach, F., Jenatton, R., Mairal, J. et Obozinski, G. (2012). Optimization with sparsity-inducing penalties. In : *Foundations and Trends in Machine Learning*. Now Publishers, Inc., 4.1, p. 1-106 (cf. p. 35, 41).

Ballard, A. J., Das, R., Martiniani, S., Mehta, D., Sagun, L., Stevenson, J. D. et Wales, D. J. (2017). Energy landscapes for machine learning. In : *Physical Chemistry Chemical Physics*. 19, p. 12585-12603 (cf. p. 107).

Barni, M., Cappellini, V. et Mecocci, A. (1996). Comments on "A possibilistic approach to clustering". In : *IEEE Transactions on Fuzzy Systems*. IEEE, 4.3, p. 393-396 (cf. p. 14).

Beck, A. et Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. In : *SIAM Journal on imaging sciences*. SIAM, 2.1, p. 183-202 (cf. p. 36).

– (2010). Gradient-based algorithms with applications to signal recovery. In : *Convex optimization in signal processing and communications*. Cambridge University Press, p. 42-88 (cf. p. 36, 44).

Belkin, M. et Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. In : *Neural computation*. MIT Press, 15.6, p. 1373-1396 (cf. p. 92).

Bellman, R. E. (1961). *Adaptive control processes : a guided tour*. Princeton university press (cf. p. 4).

Bezdek, J. C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA : Kluwer Academic Publishers (cf. p. vi, 13).

Bezdek, J. C., Hathaway, R. J., Sabin, M. J. et Tucker, W. T. (1987). Convergence theory for fuzzy c-means : counterexamples and repairs. In : *IEEE Transactions on Systems, Man, and Cybernetics*. IEEE, 17.5, p. 873-877 (cf. p. 105).

Bloomfield, P. et Steiger, W. L. (1984). *Least absolute deviations : Theory, applications and algorithms*. Springer (cf. p. 23).

Blum, A., Hopcroft, J. et Kannan, R. (2018). *Foundations of data science*, non publié. (Cf. p. 4).

Borgelt, C. (2010). Fuzzy subspace clustering. In : *Advances in Data Analysis, Data Handling and Business Intelligence*. Studies in Classification, Data Analysis, and Knowledge Organization. Springer, p. 93-103 (cf. p. vi, 25, 27, 65, 108).

Borgelt, C. et Kruse, R. (2004). Shape and size regularization in expectation maximization and fuzzy clustering. In : *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, p. 52-62 (cf. p. 25).

Burdick, D., Calimlim, M. et Gehrke, J. (2001). Mafia : A maximal frequent itemset algorithm for transactional databases. In : *Proceedings of the 17th International Conference on Data Engineering*. IEEE, p. 443-452 (cf. p. 17).

Carlsson, G. et Mémoli, F. (2010). Characterization, stability and convergence of hierarchical clustering methods. In : *Journal of Machine Learning Research*. 11, p. 1425-1470 (cf. p. 10).

Chang, W.-C. (1983). On using principal components before separating a mixture of two multivariate normal distributions. In : *Applied Statistics*. JSTOR, p. 267-275 (cf. p. 4).

Chen, S. S., Donoho, D. L. et Saunders, M. A. (2001). Atomic decomposition by basis pursuit. In : *SIAM review*. SIAM, 43.1, p. 129-159 (cf. p. 34).

Chen, X., Liao, M. et Ye, X. (2017). Projection subspace clustering. In : *Journal of Algorithms & Computational Technology*. SAGE Publications, 11.3, p. 224-233 (cf. p. 33).

Cheng, C.-H., Fu, A. W. et Zhang, Y. (1999). Entropy-based subspace clustering for mining numerical data. In : *Proceedings of the 5th ACM International Conference on Knowledge Discovery and Data mining*. ACM, p. 84-93 (cf. p. 17).

Combettes, P. L. et Pesquet, J.-C. (2011). Proximal splitting methods in signal processing. In : *Fixed-point algorithms for inverse problems in science and engineering*. Springer, p. 185-212 (cf. p. 36, 72).

Cox, T. F. et Cox, M. A. (2000). *Multidimensional scaling*. Chapman et hall/CRC (cf. p. 65).

Cutler, A. et Breiman, L. (1994). Archetypal analysis. In : *Technometrics*. Taylor & Francis, 36.4, p. 338-347 (cf. p. 107).

de Carvalho, F. A. et Pimentel, J. T. (2012). A fuzzy clustering algorithm based on adaptive city-block distances. In : *IEEE International Conference on Fuzzy Systems*. IEEE, p. 1-8 (cf. p. 23).

De Maesschalck, R., Jouan-Rimbaud, D. et Massart, D. L. (2000). The Mahalanobis distance. In : *Chemometrics and Intelligent Laboratory Systems*. Elsevier, 50.1, p. 1-18 (cf. p. 24).

Delfour, M. (2012). *Introduction à l'optimisation et au calcul semi-différentiel*. Dunod (cf. p. 5).

Deng, Z., Choi, K.-S., Chung, F.-L. et Wang, S. (2010). Enhanced soft subspace clustering integrating within-cluster and between-cluster information. In : *Pattern Recognition*. Elsevier, 43.3, p. 767-781 (cf. p. 30).

Deng, Z., Choi, K.-S., Jiang, Y., Wang, J. et Wang, S. (2016). A survey on soft subspace clustering. In : *Information Sciences*. Elsevier, 348, p. 84-106 (cf. p. 20).

Domingos, P. (2012). A few useful things to know about machine learning. In : *Communications of the ACM*. ACM, 55.10, p. 78-87 (cf. p. 4).

Döring, C., Lesot, M.-J. et Kruse, R. (2006). Data analysis with fuzzy clustering methods. In : *Computational Statistics & Data Analysis*. Elsevier, 51.1, p. 192-214 (cf. p. 11).

Elhamifar, E. et Vidal, R. (2009). Sparse subspace clustering. In : *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, p. 2790-2797 (cf. p. vi, 31, 32).

Ester, M., Kriegel, H.-P., Sander, J., Xu, X. et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In : *Proceedings of the 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. T. 96. 34, p. 226-231 (cf. p. 10).

Evers, F. T., Höppner, F., Klawonn, F., Kruse, R. et Runkler, T. (1999). *Fuzzy cluster analysis : methods for classification, data analysis and image recognition*. John Wiley & Sons (cf. p. 11).

Fuchs, J.-J. (2010). Des moindres carrés aux moindres déviations. In : *Traitement du Signal*. 27.1, p. 109-119 (cf. p. 23).

Gan, G. et Wu, J. (2008). A convergence theorem for the fuzzy subspace clustering (FSC) algorithm. In : *Pattern Recognition*. Elsevier, 41.6, p. 1939-1947 (cf. p. 22).

Gath, I. et Geva, A. B. (1989). Unsupervised optimal fuzzy clustering. In : *IEEE Transactions on pattern analysis and machine intelligence*. IEEE, 11.7, p. 773-780 (cf. p. 25).

Gorban, A. N., Tyukin, I. Y. et Romanenko, I. (2016). The blessing of dimensionality : separation theorems in the thermodynamic limit. In : *Journal of the International Federation of Automatic Control*. Elsevier, 49.24, p. 64-69 (cf. p. 4).

Gorski, J., Pfeuffer, F. et Klamroth, K. (2007). Biconvex sets and optimization with biconvex functions : a survey and extensions. In : *Mathematical methods of operations research*. Springer, 66.3, p. 373-407 (cf. p. 48, 106).

Gray, R. M. (2011). *Entropy and information theory*. Springer Science & Business Media (cf. p. 28).

Gu, J., Jiao, L., Yang, S. et Liu, F. (2018). Fuzzy double C-Means clustering based on sparse Self-Representation. In : *IEEE Transactions on Fuzzy Systems*. IEEE, 26.2, p. 612-626 (cf. p. 31).

Gu, S. et Zhou, X. (2018). Simplified gentlest ascent dynamics for saddle points in non-gradient systems. In : *arXiv preprint arXiv :1807.00654* (cf. p. 107).

Guillon, A., Lesot, M.-J. et Marsala, C. (2016a). Optimisation proximale pour le subspace clustering flou. In : *25^{e} Rencontres francophones sur la Logique Floue et ses Applications* (cf. p. 39, 70).

– (2017). Laplacian regularization for fuzzy subspace clustering. In : *Proceedings of the IEEE International Conference on Fuzzy Systems*. IEEE, p. 1-6 (cf. p. vi, 90).

Guillon, A., Lesot, M.-J. et Marsala, C. (2018a). A proximal framework for fuzzy subspace clustering. In : *Fuzzy Sets and Systems*. Elsevier (cf. p. 39, 70).

– (2018b). Régularisation laplacienne pour le subspace clustering flou. In : *27^{e} Rencontres francophones sur la Logique Floue et ses Applications* (cf. p. 90).

Guillon, A., Lesot, M.-J., Marsala, C. et Pal, N. R. (2016b). Proximal optimization for fuzzy subspace clustering. In : *International Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, p. 675-686 (cf. p. vi, 39, 70).

Gustafson, D. E. et Kessel, W. C. (1978). Fuzzy clustering with a fuzzy covariance matrix. In : *Proceedings of the IEEE Conference on Decision and Control*. T. 17. IEEE, p. 761-766 (cf. p. vi, 24, 25).

Hanmandlu, M., Verma, O. P., Susan, S. et Madasu, V. K. (2013). Color segmentation by fuzzy co-clustering of chrominance color features. In : *Neurocomputing*. Elsevier, 120, p. 235-249 (cf. p. 29).

Hoerl, A. E. et Kennard, R. W. (1970). Ridge regression : biased estimation for nonorthogonal problems. In : *Technometrics*. Taylor & Francis Group, 12.1, p. 55-67 (cf. p. 33).

Huard, P. (1979). Extensions of Zangwill's theorem. In : *Point-to-Set Maps and Mathematical Programming*. Springer, p. 98-103 (cf. p. 105).

Huo, X., Ni, X. S. et K. Smith, A. (2007). A survey of manifold-based learning methods. In : *Recent advances in data mining of enterprise data*, p. 691-745 (cf. p. 92, 101).

Jajuga, K. (1991). ℓ_1-norm based fuzzy clustering. In : *Fuzzy Sets and Systems*. Elsevier, 39.1, p. 43-50 (cf. p. 23).

Jing, L., Ng, M. K. et Huang, J. Z. (2007). An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. In : *IEEE Transactions on knowledge and data engineering*. IEEE, 19.8 (cf. p. vi, 5, 27, 28).

Kaufman, L. et Rousseeuw, P. (1987). *Clustering by means of medoids*. North-Holland (cf. p. 18).

Kaufman, L. et Rousseeuw, P. J. (1990). *Finding groups in data : an introduction to cluster analysis*. T. 344. John Wiley & Sons (cf. p. 10).

Keller, A. et Klawonn, F. (2000). Fuzzy clustering with weighting of data variables. In : *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*. World Scientific, 8.06, p. 735-746 (cf. p. vi, 20, 21, 39, 40).

Klawonn, F. (2013). What can fuzzy cluster analysis contribute to clustering of high-dimensional data ? In : *International Workshop on Fuzzy Logic and Applications*. Springer, p. 1-14 (cf. p. 108).

Klawonn, F. et Höppner, F. (2003). What is fuzzy about fuzzy clustering ? Understanding and improving the concept of the fuzzifier. In : *5th International Symposium on Intelligent Data Analysis*. Springer, p. 254-264 (cf. p. 27).

Kriegel, H.-P., Kröger, P., Sander, J. et Zimek, A. (2011). Density-based clustering. In : *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*. Wiley Online Library, 1.3, p. 231-240 (cf. p. 10).

Krishnapuram, R. (1994). Generation of membership functions via possibilistic clustering. In : *IEEE International Conference on Fuzzy Systems*. IEEE, p. 902-908 (cf. p. 87, 88).

Krishnapuram, R. et Keller, J. M. (1993). A possibilistic approach to clustering. In : *IEEE Transactions on Fuzzy Systems*. IEEE, 1.2, p. 98-110 (cf. p. 14).

Lichman, M. (2013). *UCI Machine Learning Repository*. URL : http://archive.ics.uci.edu/ml (cf. p. 63).

Lindsten, F., Ohlsson, H. et Ljung, L. (2011). Just relax and come clustering !: A convexification of k-means clustering. In : Linköping University Electronic Press (cf. p. 12).

Liu, G., Lin, Z. et Yu, Y. (2010). Robust subspace segmentation by low-rank representation. In : *Proceedings of the 27th International Conference on Machine Learning*. ICML, p. 663-670 (cf. p. 33).

Lloyd, S. (1982). Least squares quantization in PCM. In : *IEEE transactions on information theory*. IEEE, 28.2, p. 129-137 (cf. p. vi, 11).

Lu, C.-Y., Min, H., Zhao, Z.-Q., Zhu, L., Huang, D.-S. et Yan, S. (2012). Robust and efficient subspace segmentation via least squares regression. In : *European Conference on Computer Vision*. Springer, p. 347-360 (cf. p. 33).

Maaten, L. v. d. et Hinton, G. (2008). Visualizing data using T-SNE. In : *Journal of Machine Learning Research*. 9.Nov, p. 2579-2605 (cf. p. 108).

MacIver, D. R. (2018). *Python testing library, Hypothesis*. URL : https://github.com/HypothesisWorks/hypothesis (cf. p. 106).

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In : *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. T. 1. 14, p. 281-297 (cf. p. 11).

McInnes, L. et Healy, J. (2018). UMAP : Uniform manifold approximation and projection for dimension reduction. In : *arXiv e-prints* (cf. p. 108).

Mirkin, B. (1996). *Mathematical classification and clustering*. Nonconvex Optimization and its Applications. Springer (cf. p. 15).

Moreau, J.-J. (1962). Fonctions convexes duales et points proximaux dans un espace hilbertien. In : *Compte-rendu de l'Académie des Sciences de Paris Série A Math*. Société mathématique de France, 255, p. 2897-2899 (cf. p. 35).

Munch, E. (2017). A user's guide to topological data analysis. In : *Journal of Learning Analytics*. Education Resources Information Center, 4.2, p. 47-61 (cf. p. 107).

Murphy, C., Gray, P. et Stewart, G. (2017). Verified perceptron convergence theorem. In : *Proceedings of the 1st ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, p. 43-50 (cf. p. 105).

Murtagh, F. (2009). The remarkable simplicity of very high dimensional data : application of model-based clustering. In : *Journal of Classification*. Springer, 26.3, p. 249-277 (cf. p. 4).

Ng, A. Y., Jordan, M. I. et Weiss, Y. (2002). On spectral clustering : Analysis and an algorithm. In : *Advances in neural information processing systems*, p. 849-856 (cf. p. 10).

Oktar, Y. et Turkan, M. (2018). A review of sparsity-based clustering methods. In : *Signal Processing*. Elsevier, 148, p. 20-30 (cf. p. 5).

Pal, N. R. et Bezdek, J. C. (1995). On cluster validity for the fuzzy c-means model. In : *IEEE Transactions on Fuzzy Systems*. IEEE, 3.3, p. 370-379 (cf. p. 13).

Parikh, N. et Boyd, S. (2013). Proximal algorithms. In : *Foundations and Trends in Optimization*. 1.3, p. 123-231 (cf. p. 35, 37).

Parsons, L., Haque, E. et Liu, H. (2004). Subspace Clustering for High Dimensional Data : A Review. In : *SIGKDD Exploration Newsletter*. ACM, 6.1, p. 90-105 (cf. p. 17, 19).

Qiu, X., Qiu, Y., Feng, G. et Li, P. (2015). A sparse fuzzy c-means algorithm based on sparse clustering framework. In : *Neurocomputing*. Elsevier, 157, p. 290-295 (cf. p. 34).

Rockafellar, R. T. (1976). Monotone operators and the proximal point algorithm. In : *SIAM Journal on control and optimization*. SIAM, 14.5, p. 877-898 (cf. p. 35).

Rodríguez, S. I. et de Carvalho, F. d. A. (2017). Fuzzy clustering algorithm with automatic variable selection and entropy regularization. In : *Proceedings of the IEEE International Conference on Fuzzy Systems*. IEEE, p. 1-6 (cf. p. 29, 108).

Ruder, S. (2016). An overview of gradient descent optimization algorithms. In : *arXiv e-prints* (cf. p. 35, 44).

Sadaaki, M. et Masao, M. (1997). Fuzzy c-means as a regularization and maximum entropy approach. In : *Proceedings of the 7th International Fuzzy Systems Association World Congress*, p. 86-92 (cf. p. 29).

Selsam, D., Liang, P. et Dill, D. L. (2017). Developing Bug-Free Machine Learning Systems With Formal Mathematics. In : *International Conference on Machine Learning*, p. 3047-3056 (cf. p. 106).

Shi, J. et Malik, J. (2000). Normalized cuts and image segmentation. In : *IEEE Transactions on pattern analysis and machine intelligence*. IEEE, 22.8, p. 888-905 (cf. p. 32).

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. In : *Journal of the Royal Statistical Society. Series B (Methodological)*. JSTOR, 58.1, p. 267-288 (cf. p. 34, 41).

Udell, M., Horn, C., Zadeh, R., Boyd, S. et al. (2016). Generalized low rank models. In : *Foundations and Trends in Machine Learning*. Now Publishers, Inc., 9.1, p. 1-118 (cf. p. 33).

Vidal, R. (2010). A tutorial on subspace clustering. In : *IEEE Signal Processing Magazine*. IEEE, 28.2, p. 52-68 (cf. p. 33).

Von Luxburg, U. (2007). A tutorial on spectral clustering. In : *Statistics and computing*. Springer, 17.4, p. 395-416 (cf. p. 10).

Walt, S. v. d., Colbert, S. C. et Varoquaux, G. (2011). The NumPy Array : A Structure for Efficient Numerical Computation. In : *Computing in Science and Engg*. IEEE Educational Activities Department, 13.2, p. 22-30 (cf. p. 104).

Wang, W. et Carreira-Perpinán, M. A. (2013). Projection onto the probability simplex : An efficient algorithm with a simple proof, and an application. In : *arXiv preprint arXiv :1309.1541* (cf. p. 27, 53).

Weinan, E. et Zhou, X. (2011). The gentlest ascent dynamics. In : *Nonlinearity*. IOP Publishing, 24.6, p. 1831 (cf. p. 107).

Witten, D. M. et Tibshirani, R. (2010). A framework for feature selection in clustering. In : *Journal of the American Statistical Association*. 105, p. 713-726 (cf. p. 33).

Wu, K.-L., Yu, J. et Yang, M.-S. (2005). A novel fuzzy clustering algorithm based on a fuzzy scatter matrix with optimality tests. In : *Pattern Recognition Letters*. Elsevier, 26.5, p. 639-652 (cf. p. 29, 30, 109).

Xu, D. et Tian, Y. (2015). A comprehensive survey of clustering algorithms. In : *Annals of Data Science*. Springer, 2.2, p. 165-193 (cf. p. 10).

Zangwill, W. I. (1969). *Nonlinear programming : a unified approach*. Prentice-Hall (cf. p. 22, 105).

Zelnik-Manor, L. et Perona, P. (2004). Self-tuning spectral clustering. In : *Advances in Neural Information Processing Systems*. T. 17. 1601-1608, p. 16 (cf. p. 93).