Une approche stochastique à base d'arbres aléatoires pour le calcul de dissimilarités : application au clustering pour diverses structures de données

Kevin Dalleau

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Une approche stochastique à base d'arbres aléatoires pour le calcul de dissimilarités : application au clustering pour diverses structures de données

THÈSE

présentée et soutenue publiquement le 23 novembre 2021

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Kevin Dalleau

Composition du jury

Président :	Pr. Isabelle CHRISMENT, Université de Lorraine
Rapporteurs :	Pr. Pascale KUNTZ, Université de Nantes Pr. Jean-Michel POGGI, Université Paris Sud
Examinateur :	Pr. Antoine CORNUEJOLS, AgroParisTech
Encadrants :	Miguel COUCEIRO, Pr, Université de Lorraine Malika SMAIL-TABBONE, MC HDR, Université de Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

Je tiens tout d'abord à remercier mes directeurs de thèse, Malika Smaïl-Tabbone et Miguel Couceiro, pour tout le temps que vous avez passé à m'encadrer, mais aussi à me recadrer tant je pouvais parfois partir dans tous les sens. Merci de votre patience et de votre confiance – il en aura fallu au cours de ces années ! Merci enfin pour votre soutien, dans les bons moments, comme dans les mauvais.

Je souhaite également remercier les membres du jury, Antoine Cornuéjols, Isabelle Christment, Jean-Michel Poggi et Pascale Kuntz-Kosperec pour l'intérêt qu'ils ont porté à mon travail. Merci pour le temps que vous avez consacré à la lecture et l'évaluation de ma thèse, ainsi que pour vos remarques. Je vous remercie par ailleurs d'avoir accepté de faire partie de ce jury malgré les circonstances particulières.

Merci à toi Adrien, sans qui je n'aurais probablement pas suivi ce chemin. Ton soutien au cours de ces années, au delà de cette thèse, est très important à mes yeux.

Merci aux équipes CAPSID et ORPAILLEUR, qui m'ont accueillies pour cette aventure ! Merci à toi Marie-Dominique, pour tes remarques et tout ce que tu as pu faire pour moi comme pour l'équipe.

Aux amis du LORIA qui ont rendu ces années de thèse si agréables et mémorables. Ceux de l'équipe 11h30 : Antoine, Justine (que d'années depuis nos stages respectifs dans ce laboratoire !), Line... ce serait compliqué de lister tout le monde qui a pu participer à cette fameuse équipe informelle, mais vous vous reconnaîtrez. Nos débats et discussions plus légères, resterons gravés dans ma mémoire. Au bureau B142 et aux personnes qui on pu y passer, plus ou moins longtemps : Joël, Athénaïs, Antoine, Anna, Diego, Bishnu... De crypto aux pralulines, des jeux de société aux bières, tant de passions partagées !

À vous, Jean-Philippe, Quentin et Sylvain. Notre petit bout de chemin a duré un bout de temps, finalement, depuis nos années à Télécom Nancy ! Merci pour ces pauses café, ces créations de startups fictives (qui finalement ont vu le jour, par d'autres), ces débats politiques. Vous avez fait de ces années de thèse ce qu'elles ont été. Longue vie à la Vaiana team !

À mes pharmaciens, Floflo et Pierre. Flo, nos restos mexicains du midi me manquent. Pierre, toi qui me demandait de temps en temps : "Alors, tu en es où dans ta thèse ?". Ça y est, enfin, on y est !

Merci à mes parents et grands-parents. Vous m'avez toujours soutenu, je n'aurais pas été là sans vous (pour d'autres raisons que des raisons biologiques évidentes, bien sûr) ;) Merci d'être là pour moi, malgré la distance. Une pensée à mon parrain, merci à toi pour tout !

À toute la famille du NPDC. Merci d'être là, là aussi malgré la distance (toute relative malgré tout). Pour citer les supporters Lensois : "On est là, on est là, Dans le malheur ou la gloire, nous on est là, Pour l'amour du maillot que vous portez sur le dos [..]" !

Merci à toi Yuna, pour m'avoir diverti pendant une partie de la rédaction et de la préparation de la soutenance. Si tu passes par là dans quelques décennies : ton papa t'aime très fort.

Sommaire

Chapitre 3
Analyse de graphes

Chapitre 4
Forêts d'arbres extrêmement aléatoires pour le calcul de similarités dans les données hétérogènes

Table des figures

Liste des symboles

$\delta(G)$ Le degré minimal d'un graphe G.

$\Gamma(v)$ L'ensemble des noeuds voisins du noeud v.

$\mathbf{x}$ Un objet décrit par un ensemble de variables. On parle aussi de *feature vector*.

$\mathbf{x}_j$ La j-ième variable de $\mathbf{x}$

$\mathcal{C}$ Un ensemble de n clusterings $(C_1, C_2, \ldots, C_n)$

$\mathcal{H}(\mathcal{L})$ Un partitionnement hiérarchique du jeu de données $\mathcal{L}$ en partitions .

$\mathcal{L}$ Un ensemble d'apprentissage.

$\mathcal{N}(\mu, \sigma)$ Une distribution normale de moyenne μ et d'écart type σ.

$\mathcal{P}$ Une partition d'un graphe.

$\mathcal{T}$ Un ensemble d'arbres $\{t_1, \ldots, t_M\}$.

$\mathcal{U}(a, b)$ Une distribution uniforme sur l'intervalle $[a, b]$.

A Matrice d'adjacence d'un graphe.

$b(v)$ La centralité intermédiaire du noeud v.

c_k La k-ième étiquette d'un jeu de données.

cs Coefficient de silhouette.

D Une matrice de distance

d Une mesure de distance.

$d_{inter}(i)$ Distance moyenne entre l'instance considérée i et toutes les autres instances des autres clusters

$d_{intra}(i)$ Distance moyenne entre l'instance considérée i et toutes les autres instances de son cluster d'appartenance

$deg(u)$ le degré d'un noeud u.

DS_Σ La somme des dissimilarités d'une matrice de dissimilarité.

E Ensemble d'arêtes d'un graphe.

G Un graphe défini par ses noeuds V et ses arêtes E.

GI Gain d'information

$H(X)$ Entropie de Shannon de X

K	Nombre d'attributs aléatoirement sélectionnés.
k	Une fonction noyau.
M	Le nombre de modèles dans un ensemble.
m	Nombre d'arêtes dans un graphe.
m_e	Dissimilarité à base d'estimation de masse.
N	Le nombre d'objets d'un jeu de données.
n	Nombre de noeuds d'un graphe.
n_{min}	Nombre d'objets minimal afin de pouvoir effectuer une partition sur un noeud t.
Q	La modularité de Newman d'une partition d'un graphe.
S	Une matrice de similarité
s	Une mesure de similarité.
t	Un arbre de décision.
V	Ensemble des noeuds d'un graphe.
X	Jeu de données sous forme de vecteurs d'attributs, X étant la matrice dont les lignes correspondent à des vecteurs x.
D(G)	Densité d'un graphe G.

1

Introduction

Au cours de ces dernières décennies, la quantité de données générée et stockée a connu une croissance exponentielle. Cet accroissement est lié à de nombreux développements technologiques tels que l'internet des objets, la généralisation de la diffusion en continu de contenu, ou encore la génération massive de données liées aux examens médicaux (données d'imagerie, séquençage de génomes), etc. Le terme *big data* (*données massives*), introduit en 2005 par Roger Magoulas [CS15], est souvent utilisé pour décrire les données générées par ces processus. Ces données ont ouvert la voie à de nombreuses nouvelles applications, mais au coût de nouvelles difficultés de traitement, notamment de par leur volume – pouvant dépasser le seuil du pétaoctet [1] – et leur variété. En effet, les données massives peuvent provenir de plusieurs sources de données distinctes et se présenter sous diverses formes.

Ces données brutes constituent des gisements pouvant être exploités afin d'en extraire des informations, et, *in fine*, des connaissances. Il convient en effet de distinguer les notions de données et de connaissances : les données représentent une suite de symboles respectant une syntaxe et ce n'est qu'en leur donnant un sens permettant d'effectuer des raisonnements que l'on parle de connaissances [Wil02].

Le volume et la variété des données disponibles complexifie ce processus de valorisation. Les objets peuvent y être décrits par un ensemble d'attributs et se présenter sous forme tabulaire, ou comporter une composante relationnelle, ou encore comporter comme principale information les relations entre eux. C'est le cas par exemple des graphes. De plus, nombreuses sont les sources nécessitant un prétraitement conséquent, suite à un recueil manuel des données sans objectif précis.

L'extraction de connaissances à partir de données (ECD, ou *KDD*, *Knowledge Discovery in Databases*) consiste à extraire des connaissances – nouvelles et potentiellement utiles – à partir des données. Au cœur du processus complexe d'extraction de connaissances, on retrouve les algorithmes de fouille. Ces derniers peuvent se classer en deux grandes catégories : les méthodes **supervisées**, et les méthodes **non supervisées**. Là où les premières ont pour but d'entraîner un modèle à partir d'objets étiquetés (*i.e.*, dont on connaît la classe) [2], en vue d'attribuer une

1. À titre d'exemple, en 2017, le CERN a dépassé le seuil des 200 pétaoctets de données stockées. Cela correspond à la quantité stockable sur l'intégralité des bandes magnétiques vendues dans le monde en 1995 [Les97].

2. Que l'on appelle aussi *instances*.

étiquette à des objets futurs, les secondes ont pour objectif de mettre en évidence des groupes d'objets homogènes dans les données, sans connaissance a priori de leur classe d'appartenance.

Bien sûr, comme de nombreuses catégorisations, la dichotomie méthodes supervisées/méthodes non supervisées est imparfaite. En effet, on peut retrouver par exemple des approches dites semi-supervisées, où les données d'apprentissage comportent des objets étiquetés et d'autres non étiquetés ; ou encore les méthodes d'apprentissage par renforcement, où les sorties d'un modèle en réponse à des entrées sont modifiées peu à peu en fonction des conséquences de ces sorties, par un mécanisme de récompense. Par ailleurs, la fouille de données comporte aussi d'autres types d'approches répondant à des objectifs autres que la classification, telles que la recherche de motifs fréquents (*pattern mining*) ou encore la détection d'*outliers*. Nous avons fait le choix de nous limiter à ces deux grandes catégories dans ce document, dans la mesure où il s'agit des approches utilisées au sein de nos contributions. Cependant, le lecteur intéressé par les autres approches mentionnées peut se référer à cette revue de la littérature de Xiojin Zhu [Zhu05] concernant l'apprentissage semi-supervisé, et l'ouvrage de Marco Wiering et Martijn van Otterlo sur l'apprentissage par renforcement [WVO12].

Il est à noter que les termes de fouille de données (*data mining*) et apprentissage machine (*machine learning*) sont souvent utilisés de façon interchangeable. La frontière entre la signification des deux termes peut-être floue, si bien que l'un et l'autre peuvent être utilisés pour signifier la même chose. Cependant, l'analyse des deux termes indique que la fouille de données a pour objectif d'extraire des connaissances des données, alors que l'apprentissage machine vise avant tout à apprendre des modèles permettant de mieux comprendre les processus. La dichotomie entre fouille de données et statistiques est plus couramment retrouvée, notamment sur le type de problèmes et de jeux de données d'entrée [BAK18]. Il est cependant intéressant de considérer ces deux approches comme complémentaires plutôt qu'antagonistes [Fri98].

La terminologie employée pour caractériser les données et leurs représentations dépend quant à elle fortement du contexte d'origine, ainsi que de des affinités et culture des auteurs. Les objets à classer représentent les instances d'un jeu de données. Dans ce document, nous utiliserons de manière interchangeable les termes *objet* et *vecteur*. Un objet est décrit par un ensemble d'attributs. Le terme *variable* est aussi utilisé pour nommer un attribut, notamment dans le vocabulaire statistique (et le domaine biomédical). Lorsque les attributs considérés sont numériques, les objets sont représentés sous forme de vecteurs d'attributs (*feature vectors*). Un ensemble de couples attribut-valeur peut être nommé *tuple* dans le contexte des bases de données, l'ensemble des tuples formant une *table*. Là où les vecteurs d'attributs représentent des objets décrits par des attributs numériques continus, on parlera d'ensemble de tuples dans le cas où un ou plusieurs attributs sont qualitatifs. Lorsque tous les attributs d'un jeu de données sont du même type, on parle de données homogènes. Dans le cas contraire, il s'agit de données hétérogènes.

Au cours de cette thèse, nous nous sommes placés dans le contexte des approches non supervisées, et plus particulièrement des approches de *clustering*. Ces dernières sont intéressantes pour plusieurs raisons. D'une part, les approches supervisées nécessitent des données étiquetées. L'association d'une classe à chaque objet est une tâche chronophage et coûteuse, nécessitant un travail humain pouvant être considérable. D'autre part, l'étiquetage réalisé par des humains peut refléter certains biais, et est limité aux connaissances actuelles du domaine considéré (état de l'art). Enfin, effectuer un apprentissage non supervisé peut permettre de mettre en évidence des

groupes encore inconnus. Cela est particulièrement intéressant dans le domaine biomédical, où les approches non supervisées permettent la découverte de connaissances telles que la mise en évidence de sous-groupes de patients jusqu'alors inconnus, pouvant mener à de nouvelles classifications d'une maladie donnée. C'est précisément l'objet de la *nosographie*. Un point de vue expert reste néanmoins nécessaire afin d'évaluer les résultats d'un clustering et d'en tirer des conclusions utiles.

Afin d'effectuer un *clustering* d'objets sans connaissance a priori, une mesure de similarité est nécessaire. Alternativement, une mesure de distance (ou norme) ou de dissimilarité, quantifiant un niveau de proximité entre objets, mais ayant une relation inverse à la similarité, peut être utilisée. De nombreuses mesures permettant d'évaluer une (dis)similarité ont été proposées dans la littérature. Ces dernières, que nous décrirons dans la suite de ce document (chapitre 2), ne sont pas utilisables dans tous les cas de figure et peuvent être spécifiques à certains types de variables ou certains modèles de représentation des données (données quantitatives, données qualitatives, données hétérogènes, graphes...). En outre, la plupart des mesures sont sensibles à certains paramètres des données tels que la présence d'*outliers*, la présence de corrélations, des différences d'échelle, ou encore le taux de données manquantes. Une conséquence de cette spécificité et sensibilité est la nécessité d'un ou plusieurs pré-traitements des données en amont du calcul des (dis)similarités. Les plus répandus sont la normalisation, l'imputation de valeurs manquantes, la sélection d'attributs ou la dé-corrélation, la discrétisation de variables quantitatives, l'encodage numérique de variables qualitatives...

Une autre difficulté qu'un(e) analyste peut rencontrer pour réaliser un clustering provient du couplage fort qui existe entre certains algorithmes (ou certaines de leurs implémentations) et le calcul des (dis)similarités entre paires d'objets. Ce couplage prive l'analyste du choix la meilleure mesure, sans possibilité de la modifier et rendant leur utilisation moins flexible. Bien que ces choix soient parfois justifiés ou inhérents à la nature de l'algorithme, il est intéressant de découpler ces deux éléments, calcul de (dis)similarité et recherche de *clusters*.

Dans ce travail, nous nous concentrerons sur ce découplage, en proposant des méthodes de calcul de mesures de (dis)similarité permettant de s'affranchir de plusieurs des contraintes évoquées précédemment. Ces mesures, associées à divers algorithmes de clustering, sont utilisables dans deux modèles de représentation de données distincts, (i) des données structurées et hétérogènes, où les objets sont décrits par des attributs ; et (ii) des graphes ou des graphes attribués. Ces deux représentations n'ont pas été choisies au hasard. En effet, nos motivations proviennent de cas d'études concrets, dans le cadre d'un projet de recherche hospitalo-universitaire (RHU), le projet Fight-HF (*Fight Heart Failure*).

Ce projet a comme objectif une meilleure connaissance de l'insuffisance cardiaque, pathologie aux origines complexes et aux formes diverses, en exploitant (i) les données de patients issus de diverses cohortes et (ii) les graphes regroupant les connaissances sur les bio-molécules et leurs relations. Ces données proviennent de différentes sources, privées (données de suivi de cohortes par exemple) ou publiques (bases OMIM et UniProt par exemple). Exploiter les données provenant de sources aussi variées est un défi, que les partenaires du projet ont su surmonter par le développement de plateformes dédiées, sur lesquelles nous aurons l'occasion de revenir dans ce document.

La réflexion menée au cours de cette thèse a donc débuté par des constatations pratiques dans les phases d'exploration de données biomédicales. Ces données sont complexes à explorer, notamment du fait de leur hétérogénéité. D'une part, les objets (les patients typiquement) sont souvent décrits par des attributs qualitatifs tels que le sexe ou le statut tabagique, mais aussi par des attributs numériques, tels que des mesures biologiques. D'autre part, les données de biologie moléculaire (relatives aux entités biologiques telles que protéines, métabolites, médicaments, phénotypes...) sont fortement relationnelles, et il n'est pas rare de retrouver des bases de données publiques les mettant à disposition sous forme de graphes. Ces données sont précieuses dans la mesure où elles permettent d'interpréter ou donner du sens aux résultats d'analyses d'observations cliniques, mais aussi d'explorer les données patients à la lumière de ces connaissances de biologie moléculaire. Enfin les tâches de prétraitement énumérées précédemment peuvent être très chronophages dans ce contexte bio-médical et le découplage entre calcul de (dis)similarités et algorithme de clustering apporterait plus généralement un degré de flexibilité bienvenu pour traiter des données du monde réel.

Certaines approches telles que les arbres aléatoires sont connues pour leur robustesse et leur versatilité vis-à-vis des données d'entrée. Le fil directeur de cette thèse consiste ainsi à explorer et proposer des approches à base d'arbres aléatoires permettant de calculer des (dis)similarités entre objets selon différents modèles de représentation, dans le but ultime d'effectuer des tâches de classification non supervisée. Nous avons tout d'abord défini l'approche UET (Unsupervised Extra Trees) comme une extension basée sur les arbres extrêmement aléatoires permettant de calculer des (dis)similarités entre objets de manière non supervisée [DCST18]. Nous avons montré que cette approche présente de nombreux avantages, notamment en termes de performances et de non sensibilité aux spécificités des données. Nous avons de plus exploré l'application de cette méthode dans le cas des données hétérogènes, en utilisant les caractéristiques des arbres. Nous avons mené de nombreuses expériences d'évaluation comparatives sur des jeux de données standards mais également sur des données cliniques. Nous avons ensuite étendu la méthode UET aux graphes simples puis aux graphes attribués, en combinant une forêt d'arbres sur les attributs et une forêt d'arbres sur les noeuds[DCST19]. Nous avons également conduit une campagne d'évaluation importante incluant une première expérimentation sur les graphes de données du projet Fight-HF.

Cette thèse est organisée de la manière suivante. Dans le chapitre 2, nous présenterons les grandes familles de méthodes de classification supervisée et non supervisée, afin que toutes les informations nécessaires à la compréhension des contributions et de leur cadre soient présentes au sein de ce document. La dernière section de ce chapitre se focalisera tout particulièrement sur les méthodes d'évaluation des résultats des méthodes non supervisées.
Ensuite, dans le chapitre 3 nous nous concentrerons sur l'analyse de graphes, et plus particulièrement de leur clustering. Ce terme pouvant être ambigu dans le cadre des graphes, nous débuterons par une présentation du contexte et de quelques définitions, avant de présenter plusieurs grandes approches de clustering.
Nous décrirons ensuite dans les chapitres 4 et 5 l'approche que nous avons proposé, UET, ainsi que leur application d'une part aux données tabulaires, et d'autre part sur les données sous forme de graphes. Nous terminerons le document par une conclusion et quelques perspectives, chapitre 6.

2

Classification supervisée et non supervisée : deux approches différentes, mais complémentaires

Sommaire

L'exploration de données à la recherche de connaissances est un domaine qui a connu au cours de ces dernières décennies un engouement majeur. Ainsi, de nombreux travaux relatifs aux algorithmes et aux méthodologies sont menés et leurs fruits viennent enrichir une littérature déjà très dense, si bien qu'il est parfois difficile de s'y retrouver dans ce vaste éventail de méthodes. L'objectif de ce chapitre est d'éclaircir quelque peu certaines notions, et en quelque sorte de tracer un sentier dans la jungle des algorithmes disponibles, afin de comprendre les forces et les faiblesses de certaines approches majeures pour les problèmes de classification.

Nous présenterons les processus d'extraction de connaissances à partir de données dans la section 2.1. Nous développerons dans un premier temps les approches de classification dites supervisées au sein de la section 2.2. Nous présenterons ensuite des méthodes non supervisées

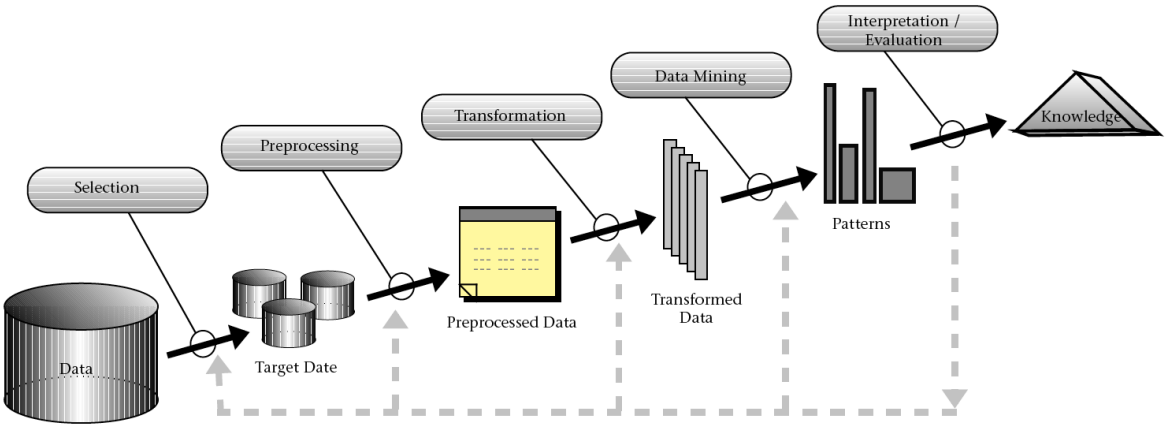


FIGURE 2.1 – Processus d'extraction de connaissances à partir de bases de données, schématisé par Fayyad *et al.* [FPSS96].

dans la section 2.3, au coeur de cette thèse. Nous discuterons enfin (section 2.4) les points de convergence et de complémentarité existant entre les deux familles de méthodes.

2.1 L'extraction de connaissances à partir de données : un processus complexe

L'extraction de connaissances à partir des données (ECD) peut se faire dans deux cadres principaux, en fonction du problème de classification posé et de la présence ou non de données étiquetées : le cadre supervisé, et le cadre non supervisé. Quelle que soit l'approche, le processus d'ECD n'est pas trivial et comporte plusieurs étapes. Une façon commune de subdiviser ce processus est celle décrite par Fayyad *et al.* dans [FPSS96], et présentée figure 2.1.

Les étapes proposées par les auteurs sont les suivantes :

La sélection des données. Cette étape a pour but de déterminer la ou les sources de données d'intérêt.

Le prétraitement des données. Une fois les données sélectionnées, il est nécessaire de les préparer avant leur exploitation. À cette étape, les données peuvent être :

- Nettoyées et corrigées. Cela peut par exemple consister en l'élimination de variables non intéressantes pour le problème, ou encore très peu remplies (on parle dans ce cas de données manquantes). Il s'agit de la tâche de *feature selection* (sélection d'attributs), qui a une place très importante dans les processus d'ECD. La tâche de correction quant à elle consiste par exemple à modifier des valeurs notablement erronées pour certaines variables.
- Homogénéisées, dans le cas où les données proviennent de diverses sources, dont l'encodage diffère.

La transformation des données. Ici, il s'agit d'adapter les données sources prétraitées à la méthode de fouille choisie. En effet, chaque méthode est adaptée à un type de données d'entrée qui lui est propre. Il peut donc être nécessaire, en fonction de la source, de discrétiser certaines variables, de normaliser les données, ou encore de changer leur formatage. Par ailleurs, au-delà des considérations liées aux attributs, cette étape de transformation peut consister en d'autres tâches, telles que le calcul de dissimilarités entre objets dans l'objectif d'appliquer une méthode de fouille non supervisée par exemple.

Comme nous l'avons déjà signalé, ces étapes de prétraitement et de transformation peuvent être très chronophages. De nombreux problèmes peuvent en effet être rencontrés et les solutions ne sont pas triviales. Tout d'abord, les données peuvent ne pas correspondre à ce qui est attendu par la méthode de fouille. Il s'agit par exemple du cas où les objets du jeu de donnée sont décrits par des variables de type différent, ce qui peut être problématique pour des méthodes destinées à des données homogènes. Certaines transformations permettent de transposer ce problème de données hétérogènes et problème de données homogènes, par exemple par discrétisation ou encore par utilisation du *One-hot encoding*. Ce sujet constitue l'une des problématiques importantes traitées au cours de cette thèse, et nous y reviendrons donc dans la suite de ce document au sein du chapitre 4.

D'autres points sont aussi à considérer, tels que les différences d'échelles dans les données, la corrélation entre variables, ou encore la présence de données manquantes. La littérature consacrée à ce dernier point foisonne de contributions, signe qu'il s'agit d'un problème actif et fréquent. Parmi les approches que l'on peut retrouver pour pallier ce problème, la plus simple reste celle consistant à remplacer les valeurs manquantes par la moyenne ou la médiane de l'attribut correspondant. Bien que simple et rapide, cette solution ne prend pas en compte les relations complexes entre attributs et peut ajouter du bruit dans les données d'entrée. Une approche similaire consiste à remplacer les valeurs manquantes par les valeurs les plus fréquentes, ce qui pose là aussi un souci concernant l'introduction d'un certain biais dans les données.

Des approches plus *intelligentes* considérant les données observées ont été proposées afin d'imputer les valeurs manquantes. Parmi elles, on retrouve l'imputation utilisant l'algorithme des k plus proches voisins [PYC+15] ou la construction d'un classifieur dédié tel qu'un perceptron multicouches [SRPMLCCdlV11] ou encore un arbre de décision [Twa09].

Enfin, le travail de sélection des attributs peut s'avérer long et complexe, en fonction des données d'entrée et de la tâche à accomplir. La littérature est par également très fournie en méthodes permettant de mener à bien cette présélection. Une revue des méthodes de sélection d'attributs est faite dans [CS14], et est significative de la quantité de méthodes à la disposition de l'expert ou de l'analyste réalisant le processus d'ECD.

Une fois ces tâches effectuées, il reste à effectuer :

La fouille des données à proprement parler, par le choix et l'utilisation de méthodes telles que des arbres de décision ou des fonctions de régression. Le choix d'un certain nombre d'hyperparamètres peut s'avérer bénéfique.

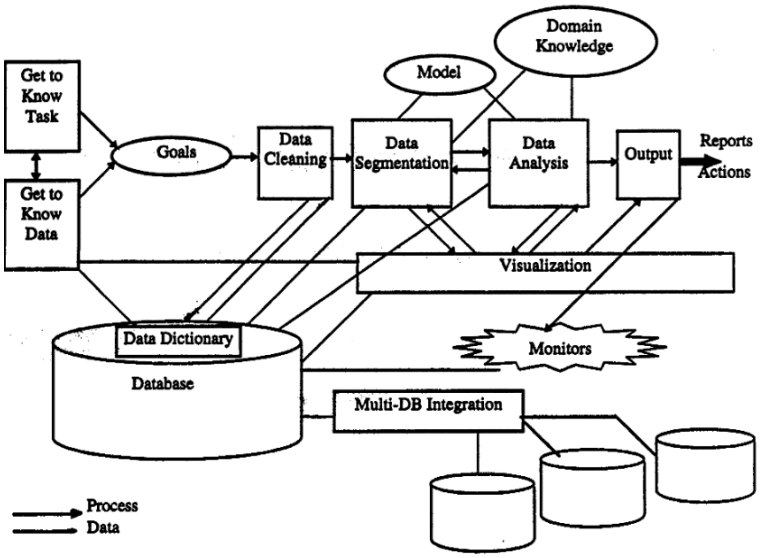


FIGURE 2.2 – Processus d'extraction de connaissances à partir de bases de données, schématisé par Brachman *et al.* [BA96].

L'évaluation et l'interprétation des résultats permettant de s'assurer de la qualité des résultats de la fouille. Dans les scénarios favorables, une interprétation experte des résultats permet de les élever au rang de connaissances potentiellement utiles pour la résolution de problèmes. Ces connaissances extraites à partir des données initiales peuvent par exemple servir de support à des raisonnements ou à des processus décisionnels.

Cette modélisation du processus d'ECD n'a pas valeur de protocole figé et a plutôt le rôle de guide méthodologique en tant que point de départ pour tout projet d'ECD. En effet, dans la pratique, certaines étapes peuvent ne pas être nécessaires. Dans le cas du clustering par exemple, la tâche de calcul de (dis)similarités entre objets peut être découplée de la tâche de fouille en elle-même, ou y être intégrée. De plus, le processus d'ECD est en pratique très souvent interactif et itératif puisque des retours utilisateurs ou experts sont régulièrement requis et certaines étapes peuvent être répétées afin d'obtenir des résultats exploitables. Cette interactivité est notamment prise en compte dans la modélisation de Brachman et Anand [BA96], présentée figure 2.2.

Cette modélisation du processus d'ECD est par ailleurs plus complète et proche de la réalité que la modélisation de Fayyad, qui a pourtant un statut quasi canonique. En effet, cette dernière met en évidence les tâches dites *get to know*, précisant que les premières étapes consistent à connaître à la fois les données et le problème posé. Il est important de le rappeler, dans la mesure où se lancer dans un processus de fouille sans objectif clair est une pratique présentant un risque élevé pour la réussite d'un projet. De plus, le fait de bien connaître les données permet de faire une sélection éclairée en amont, afin d'avoir des données propres et pertinentes pour la tâche à accomplir. En effet, il est compliqué d'obtenir de bons résultats dans le cas contraire – *garbage in, garbage out* –. Enfin, point notable, cette modélisation des processus d'ECD met explicitement en exergue l'existence d'un dictionnaire de données. Il s'agit dans le cas de certaines tâches d'une source d'information essentielle, notamment lorsque les données proviennent de sources diverses. C'est d'ailleurs dans cet esprit que le *Patient Data Model* (PDM) a été développé dans le cadre

http ://schema.org/Gender	http ://ebi.ac.uk/efo/EFO_000246	http ://my-patient-data-model/history-of-diabetes	http ://ebi.ac.uk/efo/EFO_0004348
http ://schema.org/Male	89	False	97
http ://schema.org/Male	60	False	92
http ://schema.org/Male	63	False	127
http ://schema.org/Female	78	True	69

TABLE 2.1 – Extrait de la PDM. Les colonnes sont ici nommées par des termes issus d'ontologies, afin d'homogénéiser leur description.

du projet FIGHT-HF (*Fight Heart Failure*).

Le PDM permet en effet de modéliser les données collectées sur des patients dans différentes études cliniques, aux objectifs et modalités distincts. Le tableau 2.1 présente un extrait issu de l'entrepôt de données patients. On peut remarquer ici une caractéristique des données cliniques que nous traiterons tout au long de ce document, qui est celui des données hétérogènes. En effet, on constate que les colonnes – et donc les variables décrivant chaque patient – ne sont pas du même type. Ainsi, co-existent des variables qualitatives telles que le sexe, des variables numériques continues telles que l'âge (défini par le terme EFO_0000246), ou encore binaires telles que les antécédents de diabète.

Le terme *EFO_ 0000246* est issu d'une ontologie, l'*Experimental Factor Ontology*. En effet, le PDM se base entre autres sur des définitions communes des variables, afin d'homogénéiser les informations contenues dans les différentes sources. Le développement d'un outil dédié dans ce cadre, ainsi que les ressources – humaines et financières – utilisées sont révélatrices de l'importance et de la complexité de ces tâches de prétraitement des données dans la mise en oeuvre d'un processus d'ECD. La problématique imposant ce prétraitement est ici liée à l'intégration de données provenant de diverses sources.

Enfin, une autre modélisation reconnue du processus d'ECD est CRISP-DM [She00]. Acronyme de *Cross-industry standard process for data mining*, ou *processus standard de fouille de données intersectorielle*, elle est clairement orientée vers l'aide à la décision dans un contexte industriel, ce qui n'est pas étonnant dans la mesure où le projet à l'origine du modèle fut financé par cinq grandes entreprises. La figure 2.3 présente de manière schématique cette modélisation. Ce qui est intéressant quand on la compare aux précédentes modélisations est la notion de déploiement du modèle, c'est à dire des connaissances extraites à partir des données. Intervenant à la fin du processus, cette étape consiste à déployer le modèle pour une utilisation future. Il est judicieux d'intégrer cette étape au processus d'ECD car dans la pratique le déploiement d'un modèle extrait des données est une étape importante d'un projet industriel.Le caractère sans fin ou itératif du processus CRISP-DM correspond notamment aux situations ou le modèle déployé et surveillé (on parle de *monitoring* de modèles) perd en efficacité et une nouvelle itération du processus s'avère nécessaire afin d'apprendre un modèle plus à jour. Cette déviation du modèle à la réalité au cours du temps est retrouvée sous le terme de *concept drift* [dSPFR21] [MW17]. Certains travaux, menés par exemple par des équipes de Google, s'intéressent à l'aspect pratique de ce monitoring [BCN+17].

Il est bon de mentionner cette étape, dans la mesure où souvent, un modèle à usage unique n'est pas l'objectif, mais plutôt un modèle pouvant être utile quand de nouvelles données lui sont présentées en continu, dans des processus d'aide à la décision notamment.

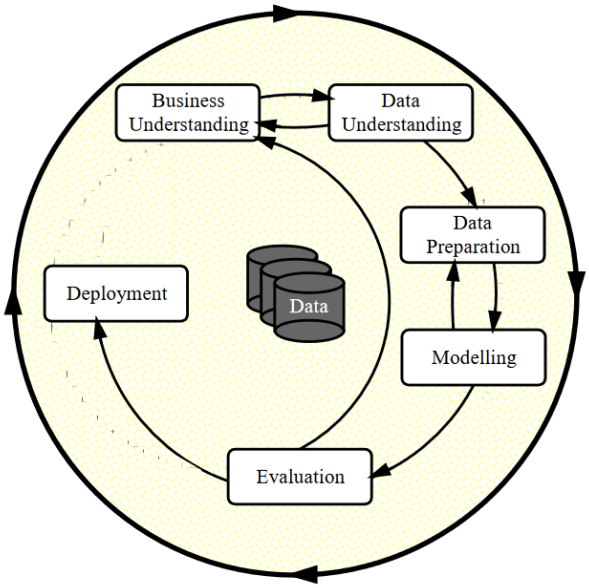


FIGURE 2.3 – Représentation de la modélisation des processus d'ECD par CRISP-DM [WH00].

Quelle que soit la modélisation du processus, ce qu'il est important de retenir ici est qu'il s'agit d'un processus complexe, constitué de différentes étapes pouvant être composées entre elles, les sorties d'un *bloc* constituant les entrées d'un autre *bloc*. Il s'agit par ailleurs de processus pouvant très souvent être itératifs.

Maintenant que nous avons décrit les principales étapes d'un processus d'extraction de connaissances, nous allons rentrer dans le détail de deux grands types d'approches. L'objectif est d'ici d'avoir une vision d'ensemble des approches existantes, sans pour autant en présenter une liste exhaustive, et ce afin de (i) mettre en évidence les forces et faiblesses des différentes approches et (ii) noter les points de convergence entre approches. L'objectif est par ailleurs d'introduire des concepts importants dans le cadre du travail présenté dans cette thèse.

Nous commencerons dans la section 2.2 par présenter quelques approches de classification supervisée. Nous nous concentrerons tout particulièrement sur les méthodes à bases d'arbres de décision et nous intéresserons aux méthodes d'ensembles, concepts clés au sein des approches que nous avons proposées. Nous présenterons ensuite quelques approches non supervisées au sein de la section 2.3. Cette section se termine par une description des approches d'évaluation du résultat d'un clustering, évaluation complexe dans la mesure où la *vérité* terrain n'est pas toujours connue.

2.2 Classification supervisée

La classification supervisée consiste à construire un modèle à partir de données d'entraînement et d'une variable cible, la classe – ou étiquette –. Ce modèle permet par la suite de prédire la valeur de cette classe pour de nouveaux objets pour lesquels elle n'est pas connue. Formellement, l'objectif d'un algorithme de classification supervisée est d'apprendre une fonction $\hat{y} = f(\mathbf{x})$, qui, étant donné un objet $\mathbf{x}$, retourne une ou plusieurs étiquettes parmi un ensemble fini d'étiquettes

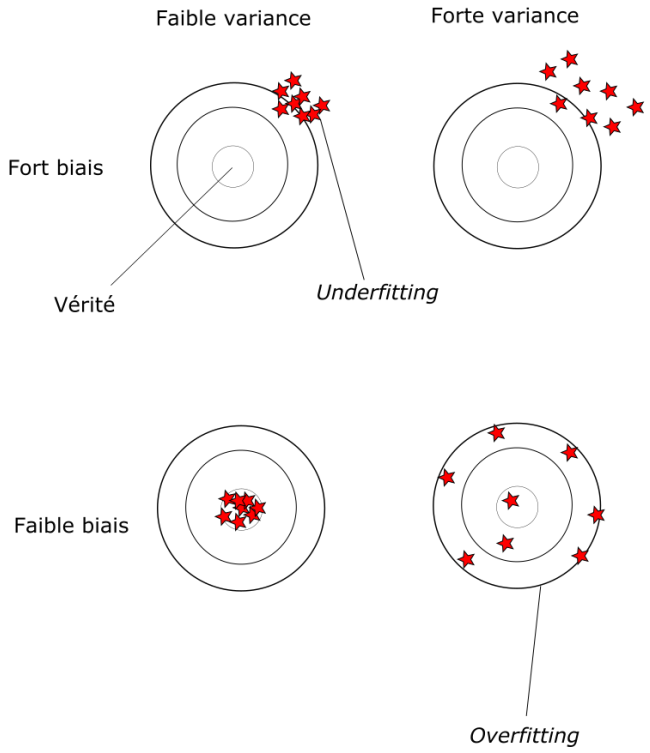


FIGURE 2.4 – Représentation schématique du biais et de la variance sur les résultats d'un modèle. Chaque étoile représente la distance de la prédiction du modèle à la cible, *i.e.* la valeur attendue. Le cas idéal correspond au cas où la méthode présente un biais et une variance faibles. Cette représentation sous forme de cible est inspirée de [Dom12].

$c_1, c_2, \cdots, c_k$ dans le cas d'un problème de classification, ou un nombre réel dans le cas d'un problème de régression.

Une partie de la complexité d'apprendre un *bon* modèle – au-delà de tous les processus préalables – repose dans le fait qu'un compromis entre biais et variance doit être trouvé. Commençons par définir ces deux termes. La variance d'un modèle correspond à l'erreur liée à sa dépendance aux données d'entraînement, plus précisément aux faibles variations dans ces dernières. Un modèle ayant une variance élevée est très sensible aux modifications des données d'entrée. Le biais correspond quant à lui à l'erreur intrinsèquement liée au modèle choisi. Intuitivement, cela correspond à l'erreur que le modèle ferait s'il avait pu être construit sur la base de l'ensemble des données possibles. Formellement, soit une estimation $\hat{y}$ d'une fonction y par un estimateur. On définit le biais comme la valeur $E[\hat{y}] - y$, et la variance par $E[(\hat{y} - y)^2]$. La figure 2.4 présente une représentation schématique de ces deux notions. La notion importante d'*overfitting* (ou surapprentissage) y est présentée. Un modèle qui se trouve dans ce cas modélise parfaitement la fonction permettant d'associer une classe aux données sur lesquelles il a été construit, mais n'est

pas généralisable. Ainsi, il est important de valider un *bon* modèle afin de s'assurer que ce dernier n'est pas valide uniquement sur les données avec lesquelles il a été construit. Ce cas correspond à un modèle ayant un faible biais – le modèle ne ferait pas d'erreurs s'il était construit sur la base de l'ensemble des données possibles – mais une forte variance – le modèle est très sensible aux fluctuations et au bruit présent dans l'ensemble d'apprentissage –. Un modèle capable de donner de bons résultats sur des données nouvelles est dit généralisable. Le compromis entre le biais et la variance d'un modèle est a priori inévitable (il s'agit du *bias-variance tradeoff*) [DSH20], ce qui rend ces notions d'autant plus importantes.

De très nombreux algorithmes d'apprentissage existent, ayant tous leurs points forts et leurs points faibles, en fonction notamment des besoins de la tâche et des données d'entrée. Bien qu'il peut parfois être difficile de s'y retrouver, ces méthodes peuvent être regroupées dans un nombre restreint de catégories. Parmi ces catégories, on trouve les méthodes géométriques, les méthodes probabilistes, les méthodes logiques, ou encore les méthodes d'apprentissage profond [Fla12]. Comme beaucoup de classifications en groupes stricts, cette dernière ne reflète pas parfaitement la variété des algorithmes de classification supervisée, mais est intéressante pour effectuer une présentation de l'état de l'art. Nous présenterons ici de manière succincte certaines de ces méthodes, et nous concentrerons particulièrement sur une méthode, les arbres de décision, appartenant au groupe des méthodes logiques selon [Fla12].

2.2.1 Méthodes géométriques

Les méthodes géométriques se basent sur des concepts tels que les droites et leurs généralisations dans des espaces vectoriels de plus grande dimension (*i.e.*, des hyperplans) afin de modéliser la séparation des objets en classes . Ces hyperplans représentent une séparation entre les classes des objets. La figure 2.5 présente des objets décrits par deux dimensions, $\mathbf{x_1}$ et $\mathbf{x_2}$. Dans cet exemple, les objets peuvent être associés à deux classes, représentées par les cercles pleins et les cercles vides. Trois droites, H_1, H_2 et H_3 représentent trois possibilités – parmi une infinité – de les séparer. On constate que la droite H_3 ne constitue pas un bon séparateur dans le cas de ces données d'exemple. En effet, contrairement à H_1 et H_2, celle-ci ne permet pas de discriminer les objets en fonction de leur classe de manière efficace. L'objectif des algorithmes géométriques est donc de trouver un hyperplan permettant de bien séparer les objets en fonction de leur classe, tout en étant relativement simple et généralisable.

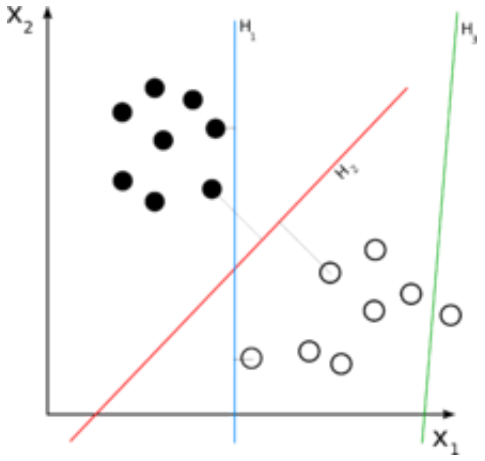


FIGURE 2.5 – Exemple de données pouvant être séparées par un hyperplan.

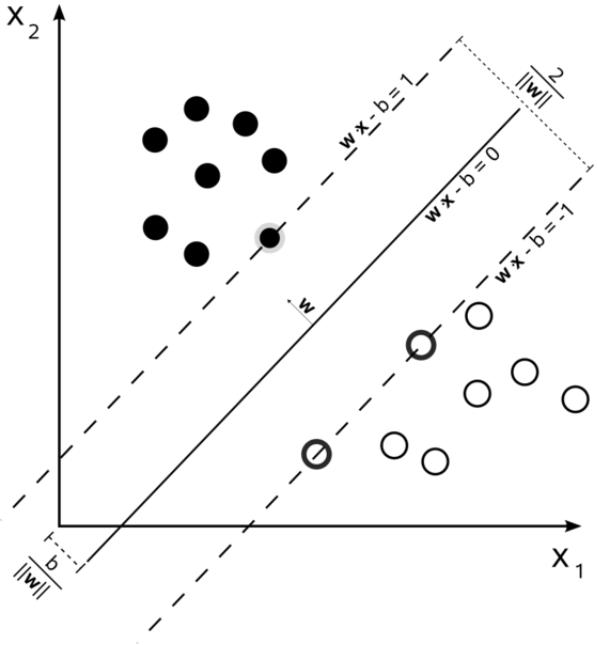


FIGURE 2.6 – Ensemble d'objets séparés par un hyperplan maximisant à la marge, d'équation $y = \mathbf{w} \cdot \mathbf{x} - b$. Les frontières en pointillé correspondent aux vecteurs supports [Cyc08].

Dans la mesure où une infinité d'hyperplans candidats existent se pose la question de l'hyperplan optimal. Intuitivement, une option consiste à choisir celui pour lequel la distance entre les objets les plus proches de la frontière de décision pour chacune des classes –les *vecteurs supports*– et ce dernier est la plus grande. Cette distance est aussi appelée *marge maximale*. La méthode des machines à vecteur de support (*SVM*, *Support Vector Machine*) constitue un représentant des méthodes géométriques visant à maximiser cette marge [BGV92]. Un exemple est présenté figure 2.6.

Soit $\mathbf{x}$ un objet. La phase de construction du modèle pour ce type de méthode consiste à apprendre une frontière de décision définie par $\mathbf{w} \cdot \mathbf{x} - b = 0$, où $\mathbf{w}$ représente un vecteur normal

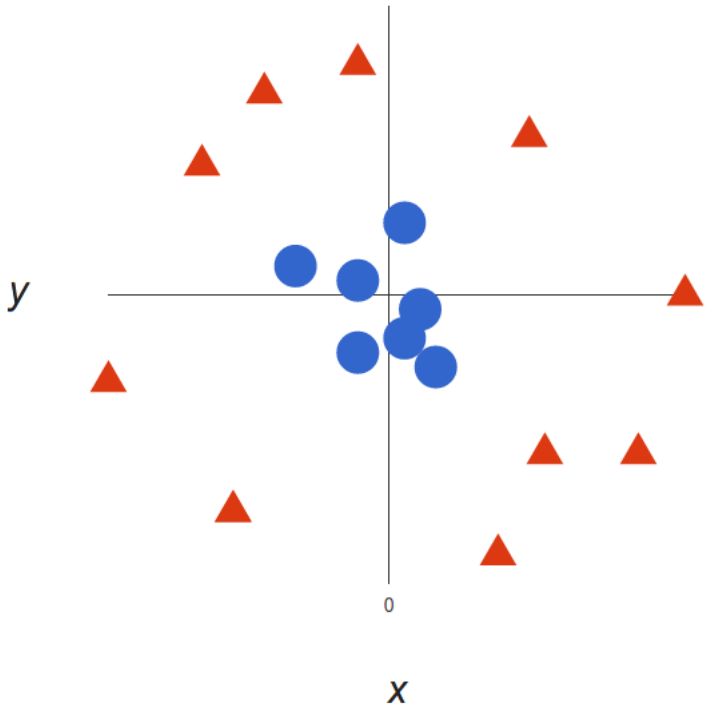


FIGURE 2.7 – Exemple de données ne pouvant pas être séparées par un hyperplan.

à cet hyperplan. L'attribution d'une classe à un nouvel objet inconnu consiste à évaluer le signe de $\mathbf{w} \cdot \mathbf{x} + b$ pour cet objet $\mathbf{x}$. Formellement, cela correspond à l'équation 2.1.

$$\mathbf{w} \cdot \mathbf{x} + b \begin{cases} \geq 0 & \mathbf{x} \text{ est de classe 1} \\ < 0 & \mathbf{x} \text{ est de classe 0} \end{cases} \tag{2.1}$$

Le biais b est souvent noté w_0 dans la littérature. Ce dernier permet de contrôler l'hyperplan séparateur, et de faire en sorte que ce dernier ne passe pas forcément par l'origine. La fonction de décision est donc $f(x) = signe(\mathbf{w} \cdot \mathbf{x} + w_0)$, et on montre que l'objectif est de trouver les $\mathbf{w}$ et w_0 correspondant à un minimum de $\frac{1}{2}||\mathbf{w}||^2$.

Cependant, qu'en est-il des données réelles, souvent bruitées ? En effet, cette approche est très peu tolérante aux variations des marges fixées par apprentissage sur les données d'entrée. La réponse à cette question se trouve dans les SVM à marges poreuses (*soft margin*), ajoutant un paramètre C, constante permettant de contrôler la concession biais/variance. En effet, de grandes valeurs de C sont associées à une faible variance et un biais fort, alors que de faibles valeurs de C mèneront à une forte variance et un faible biais.

Nous avons mentionné précédemment que ces méthodes ne peuvent s'appliquer que dans le cas où les données sont linéairement séparables. Il arrive que les données, bien que ne respectant pas cette condition, puissent être séparées. Prenons l'exemple des données présentées figure 2.7.

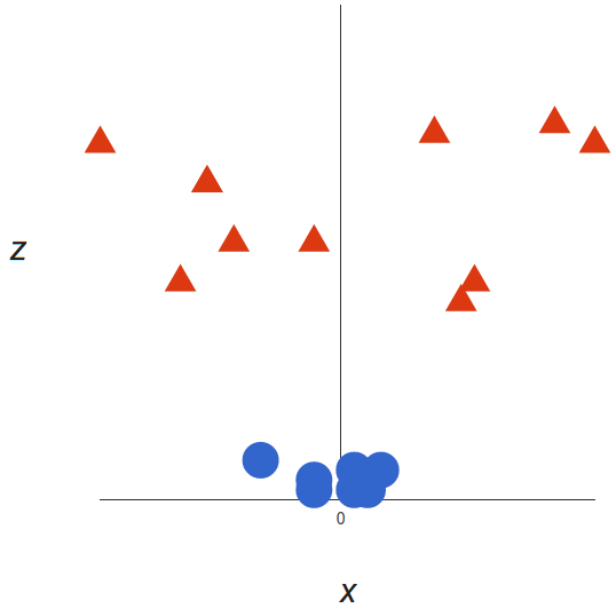


FIGURE 2.8 – Exemple de coupe horizontale de l'espace après transformation.

Bien qu'aucune droite ne puisse séparer les objets bleus des objets rouges, une astuce permet de le faire. Considérons une nouvelle dimension z, où les coordonnées de chaque point du jeu de données sont obtenues par la relation $z = x^2 + y^2$. En effectuant une coupe de cet espace de trois dimensions, on peut retrouver l'exemple de la figure 2.8. Dans ce cas, on constate qu'une droite peut maintenant séparer les objets des deux classes. Dans la mesure où cette droite est d'équation $z = x^2 + y^2$, on tombe sur l'équation d'un cercle, entourant les objets bleus de départ.

Le passage d'un espace $\mathcal{X}$ à un espace $\mathcal{V}$ peut cependant être coûteux en ressources. En effet, dans l'exemple ci-dessus, nous avons considéré des objets décrits dans deux dimensions, *i.e.* décrits par deux attributs, et n'avons ajouté qu'une dimension afin de pouvoir obtenir une séparation linéaire. Cette transformation peut être bien plus complexe dans la pratique, ajoutant de nombreuses dimensions. L'utilisation de l'astuce du noyau (*kernel trick*) permet de passer outre ce problème.

Afin de comprendre cette approche, un peu de terminologie est nécessaire. Soit $\phi : \mathcal{X} \to \mathcal{V}$ une fonction permettant de transformer l'espace d'entrée en espace de dimension supérieure doté d'un produit scalaire, où une séparation linéaire est possible, appelé espace de redescription. Un noyau est une fonction $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, qui prend en entrée deux vecteurs de l'espace d'entrée et retourne un réel. Cette valeur correspond au produit scalaire entre ces vecteurs, et peut-être considérée comme une notion de similarité entre ces derniers. Formellement, cette fonction est donc sous la forme présentée équation 2.2.

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \tag{2.2}$$

L'astuce du noyau consiste à éviter d'effectuer explicitement la transformation des données d'entrée par le biais de la fonction ϕ, en utilisant le noyau k. Bien qu'il existe de nombreuses

propositions de fonctions de noyau, certaines sont plus courantes que d'autres. On retrouve notamment de manière usuelle dans la littérature et les implémentations le noyau polynomial (équation2.3), le noyau gaussien (équation 2.4) ainsi que la fonction de base radiale (*Radial Basis Function*, ou *RBF*, équation 2.5) .

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^d \tag{2.3}$$

$$k(\mathbf{x}, \mathbf{y}) = exp(-\frac{||\mathbf{x} - \mathbf{y}||^2}{2\sigma^2}) \tag{2.4}$$

$$k(\mathbf{x}, \mathbf{y}) = exp(-\gamma||\mathbf{x} - \mathbf{y}||^2) \tag{2.5}$$

Le choix de cette fonction n'est cependant pas trivial, et dépend comme souvent des données d'entrée. Bien que seules quelques fonctions disponibles soient utilisées en pratique, trouver la bonne approche parmi tout l'éventail de noyaux existants peut s'avérer chronophage.

Nous n'avons ici présenté en détail qu'un exemple particulier de méthodes géométriques, les machines à vecteur de support. Il existe cependant de nombreuses autres méthodes, telles que l'algorithme du perceptron [Ros61] – à la base des réseaux de neurones notamment –, ou encore l'algorithme de Ho et Kashyap [HK65]. Les méthodes linéaires, en particulier les machines à vecteur de support, ont pour avantage d'être efficaces dans le cas de données de haute dimension, mais surtout d'être très versatiles de par l'utilisation de diverses fonctions de noyau. En effet, la possibilité de travailler sur des données n'étant pas linéairement séparables grâce à l'astuce du noyau présente un intérêt majeur.

Bien que performantes, ces approches souffrent d'une limitation : leur application à des données qualitatives n'est souvent pas possible sans un passage par une étape de prétraitement, potentiellement délétère concernant l'information contenue dans les données. De plus, elles sont très sensibles au bruit. Enfin, les SVM ne fournissent en sortie qu'une classe d'appartenance, un certain nombre d'opérations supplémentaires sont alors nécessaires pour obtenir des probabilités.

2.2.2 Méthodes probabilistes

Nous avons vu précédemment que les méthodes d'apprentissage supervisé ont pour objectif d'apprendre une fonction $f(\mathbf{x})$ pouvant retourner la classe d'appartenance d'un objet $\mathbf{x}$. Les méthodes probabilistes étendent ce concept de classe d'appartenance à une probabilité d'appartenance. Un modèle probabiliste retourne donc une distribution de probabilités conditionnelles $P(Y|\mathbf{x})$, associant à $\mathbf{x}$ une probabilité d'appartenance à chacune des classes de Y.

La classification naïve bayésienne est un exemple de méthode probabiliste. Rappelons une relation essentielles à la compréhension de cette méthode : la règle de Bayes, présente équation 2.6 et la formule des probabilités composées.

$$P(Y|X) = \frac{P(\mathbf{x}|Y)P(Y)}{P(\mathbf{x})} \tag{2.6}$$

On sait par ailleurs que lorsque les variables sont indépendantes conditionnellement à Y, on a $P(x_1, x_2, \ldots, x_n|Y) = P(x_1|Y)P(x_2|Y)\ldots P(x_n|Y)$. Le terme *naïf* dans l'appellation de la

méthode vient justement du fait que cette hypothèse d'indépendance est faite : on suppose que l'effet d'une variable x_i sur la classe d'appartenance Y ne dépend pas des autres variables. Ainsi, les corrélations entre prédicteurs ne sont pas prises en compte. Il s'agit d'une hypothèse forte, et d'une limitation de l'approche dans la mesure où cela est rarement le cas en pratique.

En utilisant la règle de Bayes, on constate que la sortie du modèle $P(Y|\mathbf{x})$ correspond à la relation présentée équation 2.6. $\mathbf{x}$ étant un vecteur $(x_1, x_2, \ldots, x_n)$, on a :

$$P(Y|\mathbf{x}) = \frac{P(x_1|Y)P(x_2|Y)\ldots P(x_n|Y)P(Y)}{P(x_1)P(x_2)\ldots P(x_n)}. \tag{2.7}$$

On constate que pour un jeu de données, le dénominateur est constant. On peut donc finalement écrire :

$$P(Y|\mathbf{x}) \propto P(x_1|Y)P(x_2|Y)\ldots P(x_n|Y)P(Y) = P(Y)\prod_{i=1}^{n} P(x_i|Y) \tag{2.8}$$

Dans le cas où les variables x_i sont des variables continues, il est possible de calculer les valeurs individuelles des $P(x_i|Y)$ grâce à des lois de probabilités connues, telles que la loi normale. Dans ce cas, l'estimation de l'espérance μ_y et de la variance σ_y^2 pour une classe $y \in Y$ à partir des données nous permet par exemple d'obtenir :

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp^{-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}} \tag{2.9}$$

On suppose ici que les données suivent cette loi, pouvant constituer une limite de l'approche. D'autres lois peuvent cependant bien être utilisées. Dans le cas où l'on sait que distribution des données ne suit pas la loi normale, d'autres approches peuvent être utilisées, telles que l'estimation par noyau (*kernel density estimation*) [JL13].

Afin d'obtenir une classe en sortie, il est possible d'utiliser le maximum a posteriori (MAP) ou encore le maximum de vraisemblance (MV) [3]. Les deux méthodes sont présentées équations 2.10 et 2.11.

$$y_{MAP} = \underset{Y}{\operatorname{argmax}}\, P(\mathbf{x}|Y)P(Y) \tag{2.10}$$

$$y_{MV} = \underset{Y}{\operatorname{argmax}}\, P(\mathbf{x}|Y) \tag{2.11}$$

Dans le cas de l'approche utilisant le maximum de vraisemblance, on suppose que les classes sont distribuées de manière uniforme, ou on choisit de les ignorer.

Il ne s'agit ici que d'une méthode probabiliste, présentée à titre d'exemple. D'autres méthodes, telles que la régression logistique [Ber44] ou l'algorithme de maximisation de l'espérance (*expectation-maximization* ou EM) [DLR77], sont elles aussi probabilistes par nature. La régression logistique est particulièrement appréciée dans le domaine biomédical, dans la mesure où elle permet de mettre en évidence l'influence de chaque variable sur un résultat.

3. ou *maximum likelihood*.

2.2.3 Arbres de décision

Les arbres de décision constituent l'une des approches majeures en apprentissage automatique, de par leur versatilité, leur interprétabilité et leur efficacité. Leur principe général est d'effectuer une succession de partitions de l'espace d'entrée, par le biais de tests effectués sur les variables décrivant les objets de cet espace. Un arbre est construit [4] selon une approche descendante, où la variable séparant le mieux l'ensemble des objets est sélectionnée à chaque étape. Cette approche est commune à tous les algorithmes d'arbres de décision. Ces derniers diffèrent cependant sur certains éléments, tels que les heuristiques concernant leur apprentissage à partir des exemples. Ces heuristiques sont nécessaires dans la mesure où il n'est pas envisageable d'effectuer l'ensemble des tests, l'espace de recherche pouvant être conséquent [5]. De plus, il est nécessaire d'évaluer la pertinence des tests.

Une approche afin de sélectionner la variable et sa valeur de séparation est de considérer l'entropie et le concept de gain d'information. Les algorithmes ID3 [Qui86], et son amélioration proposée 7 ans plus tard par le même auteur, C4.5 [Qui93] utilisent cette approche à base d'entropie.

Soit $H(X)$ l'entropie définie par l'équation 2.12 :

$$H(X) = -\sum p_i log(p_i) \tag{2.12}$$

, où p_i représente les proportions d'objets appartenant à la classe i dans le jeu de données. Cette mesure permet de quantifier l'homogénéité d'un ensemble.

La figure 2.9 présente un exemple de jeu de données avec deux classes. Notons p_1 les probabilités des objets rouges, et p_2 celle des objets bleus. Dans le cas (a), on a $p_1 = \frac{8}{13}$, et $p_2 = \frac{5}{13}$. On a donc :

$$H = -(\frac{8}{13}log(\frac{8}{13}) + \frac{5}{13}log(\frac{5}{13})) = 0.961 \tag{2.13}$$

Dans le cas (b), on constate que la classe 1 des objets rouges est majoritaire. On a ici

$$H = -(\frac{8}{10}log(\frac{8}{10}) + \frac{2}{10}log(\frac{2}{10})) = 0.722. \tag{2.14}$$

Afin d'évaluer la pertinence d'une variable pour le branchement, la différence d'entropie entre la répartition des objets dans données initiales et celle des objets après partition suite à un test sur une variable. Le gain d'information (que l'on notera GI) est obtenu par l'équation 2.15.

$$GI(X, x_j) = H(X) - H(X|x_j) \tag{2.15}$$

où x_j représente un attribut.

4. On parle aussi d'induction d'arbre.
5. À titre d'exemple, 20 noeuds ayant au maximum 2 descendants donnent plus d'un milliard d'arbres possibles.

(a) (b)

FIGURE 2.9 – Illustration de l'entropie, mesurant la *pureté* d'un échantillon. Ici, dans le cas (a), cette pureté est faible, donnant une entropie de 0.961, plus élevée que dans le cas (b) où elle est de 0.722.

Une bonne variable est une variable qui diminue l'entropie, *i.e*, qui *contient beaucoup d'information*. Afin de guider la construction de l'arbre, il est nécessaire de comparer le gain d'information associé à chacune des variables. La variable ayant le GI le plus élevé sera la variable à sélectionner.

Une autre métrique utilisée dans le choix de la meilleure partition est l'indice de Gini, utilisée notamment en tant que critère de division dans l'algorithme CART [BFSO84]. Le calcul de cette valeur est présenté équation 2.16, avec $|c|$ le nombre de classes et p_i la proportion d'instances appartenant à la classe i. On constate que sa valeur est minimisée lorsque la somme est égale à 1, c'est-à-dire que l'ensemble des instances appartiennent à la même classe.

$$Gini = 1 - \sum_{i=1}^{|c|} p_i^2 \tag{2.16}$$

Ces processus de partition peuvent être effectués sur des variables pouvant être de différents types, qualitatif ou continu. Il est intéressant de noter que dans le cas où tous les attributs de l'ensemble de données initial sont continus, on obtient une séparation en régions de l'espace de données, où les régions sont séparées par des hyperplans orthogonaux aux axes. Un exemple est présenté figure 2.10.

Quel que soit l'algorithme choisi, le processus de partition est répété jusqu'à ce qu'une condition d'arrêt soit rencontrée. Tout d'abord, l'une des premières conditions qui ne dépend pas de l'utilisateur, mais de la méthode elle-même, est lorsqu'aucun noeud ne peut subir le processus de partition. Cette condition est rencontrée dans deux cas :

1. Les objets contenus dans le noeud sont constants : ils ont tous la même valeur pour l'attribut sélectionné. Dans ce cas, bien qu'il est possible de découper de tels noeuds, il s'agit en pratique d'un condition d'arrêt. En effet, cela n'aurait pas d'utilité.
2. Le noeud est homogène, c'est-à-dire qu'il ne contient que des objets appartenant à la même classe.

Dans ce cas, on obtient un arbre dit maximal. Le cas où les feuilles sont constituées uniquement de noeuds homogènes est une condition qui n'est pas toujours respectable ni désirée. En effet, en augmentant la profondeur de l'arbre – et *in fine* le nombre de feuilles –, le modèle aura tendance à avoir une très forte variance et un très faible biais : il s'agit d'un modèle qui a tendance à surajuster (*overfitter*) les données d'entrée. L'*overfitting*, ou surajustement, correspond au

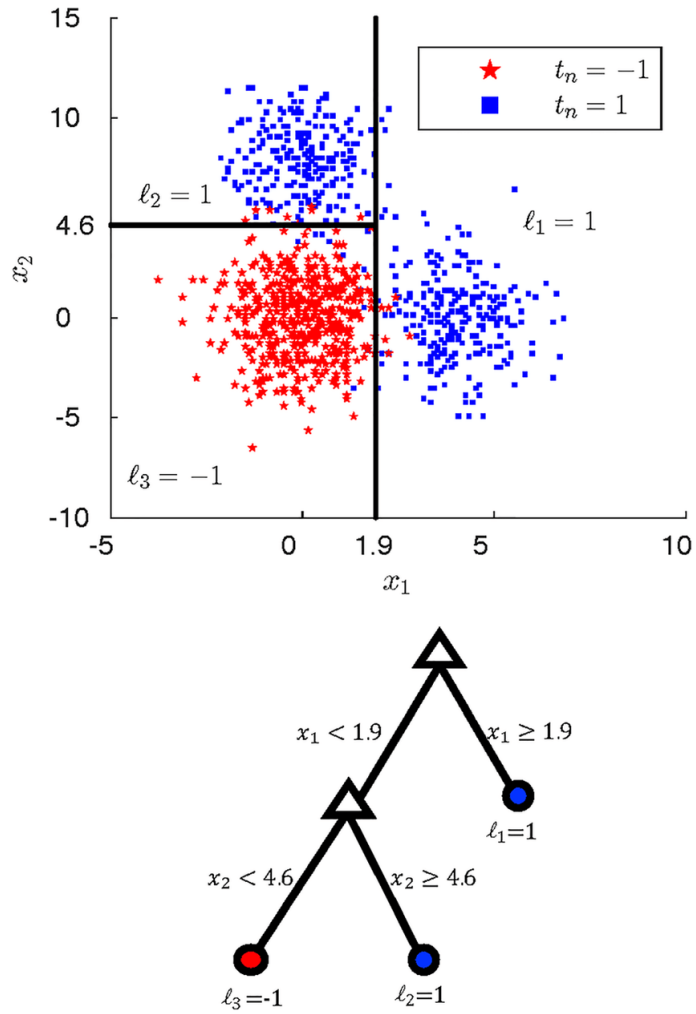


FIGURE 2.10 – Exemple d'espace – ici, de dimension 2 – séparé en 3 régions par un arbre de décision [NRG17].

cas où le modèle est parfaitement adapté aux données sans être généralisable à de nouvelles observations. Ainsi, d'autres conditions d'arrêt peuvent être définies.

Le nombre minimal d'objets pour effectuer une partition Noté n_{min} et parfois retrouvé sous le nom de *smoothing parameter* [GEW06], ce paramètre détermine si un noeud est une feuille ou non. Si le nombre d'objets contenus dans un noeud est inférieur à n_{min}, alors ce noeud est une feuille.

La profondeur maximale de l'arbre Ce paramètre indique qu'un noeud est une feuille si sa profondeur est supérieure ou égale à la valeur de profondeur maximale de l'arbre fixée par l'utilisateur.

La variation du critère de séparation Si la variation du gain d'information ou d'indice de Gini est inférieure à un seuil ϵ fixé par l'utilisateur, le noeud est considéré comme une feuille.

Déterminer les valeurs de ces paramètres est essentiel dans la mise en place d'une stratégie de fouille. En effet, l'ensemble de ces paramètres participe à la détermination de la taille de l'arbre et influe sur ses performances. Un grand arbre aura tendance à donner un modèle peu généralisable, comme indiqué précédemment. Un petit arbre en revanche aura cependant tendance à ne pas voir l'ensemble des données d'entrée, dans la mesure où sa croissance s'arrêtera prématurément. Il s'agit donc d'un compromis à trouver.

En fonction du type de la classe cible, les arbres peuvent être des arbres de classification ou des arbres de régression. De plus, les attributs des objets sur lesquels les tests sont effectués peuvent être de type différent.

Les arbres de décision peuvent être élagués (*pruning*), notamment dans le cas des arbres maximaux. Cette technique consiste à couper les branches de l'arbre ne contenant qu'un nombre réduit d'objets [6]. L'intuition est ici que ces branches ne se concentrant que sur quelques objets bien spécifiques, ces dernières nuisent à la généralisation du modèle. L'une des approches, proposée par Quinlan en 1987 [Qui87], consiste à élaguer une branche contenant peu d'instances, et d'appliquer le nouveau modèle aux données d'entraînement. Si la performance du modèle n'est pas dégradée, alors la suppression de cette branche est validée. Cette approche, nommée *reduced error pruning* constitue une bonne base pour l'évaluation d'autres méthodes, nombreuses dans la littérature [EMSK97].

Les arbres de décision constituent une méthode de choix dans les tâches d'apprentissage supervisé. Parmi leurs avantages, on retrouve :

- Leur capacité à pouvoir être construits sur des données hétérogènes.
- Leur similarité avec les processus décisionnels *humains*.
- L'interprétation aisée des modèles.
- Leur robustesse au bruit et aux *outliers*.
- Le fait qu'il s'agisse de modèles non paramétriques, ne nécessitant pas de supposition *a priori*.

6. Il s'agit plus précisément d'un post-élagage. En effet, le fait d'effectuer une partition sur un noeud lorsque la valeur du critère de partition choisi est inférieure à une certaine valeur correspond à un élagage.

Cependant, il est à noter que les performances et la robustesse des arbres de décision seuls sont de manière générale inférieures à d'autres méthodes [JWHT13]. De plus, ces modèles sont connus pour avoir une forte variance, mais un faible biais.

Il est possible de diminuer la variance des arbres de décision en utilisant des méthodes dites d'ensemble, combinant plusieurs arbres en un seul modèle.

2.2.4 Méthodes d'ensemble pour la classification supervisée

Les méthodes d'ensemble sont basées sur l'intuition que la combinaison de plusieurs modèles de classification produit un modèle plus performant, et ce pour trois raisons [Die00]. Premièrement, il s'agit d'une raison statistique : lorsque l'ensemble d'apprentissages est de taille réduite, agréger les résultats de plusieurs modèles permet de réduire l'erreur de chaque modèle. Deuxièmement, de nombreux algorithmes peuvent se retrouver coincés dans un minimum local, et retourner une prédiction incorrecte. Un ensemble de modèles construits sur différents sous-ensembles de données peuvent limiter ce risque, et retourner une meilleure approximation de la fonction représentée par les données que les modèles individuels. Enfin, le fait de combiner différents modèles permet de modéliser de manière plus fidèle la fonction recherchée.

On retrouve de manière intéressante ces observations dans les comportements humains. Cette théorie de l'intelligence collective où un ensemble d'individus est plus à même de résoudre un problème qu'un individu la composant seul est applicable dans de nombreux domaines [Sur04] [7].

Les méthodes d'ensembles courantes sont le *bagging*, *boosting* et *stacking*, que nous décrirons dans cette section. Nous insisterons plus particulièrement sur le *bagging*, dans la mesure où il s'agit d'une notion importante dans le travail présenté ici.

2.2.4.1 Quelques approches générales de méthodes d'ensemble

Bagging Le bagging, pour *bootstrap aggregating* (agrégation bootstrap) est une méthode d'ensemble fonctionnant en générant un certain nombre de jeux de données d'apprentissage de taille n, par échantillonnage des objets du jeu de données initial. Les échantillons réalisés sont des échantillons (*bootstrap*) [8], donnant le nom à la méthode.

Il s'agit d'une méthode d'échantillonnage où les échantillons sont prélevés de manière uniforme et avec remise. Dans un jeu de données initial $X = ((x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n))$, seuls environ 63% des objets sont présents après échantillonnage. En effet, après N tirages avec remise, la probabilité qu'un objet ne soit jamais sélectionné tend vers 0.368, comme indiqué par l'équation 2.17 [CL14].

$$\lim_{N\to\infty}(1-\frac{1}{N})^N \approx \frac{1}{e} \approx 0.368 \tag{2.17}$$

L'ensemble des modèles construits sur les différents échantillons bootstrap sont par la suite agrégés avec une fonction d'agrégation adaptée (moyenne, majorité, etc.). La figure 2.11 illustre le fonctionnement du bagging.

7. La chaîne Youtube «Fouloscopie» traite d'ailleurs du sujet d'une manière légère et ludique.
8. Le terme *à la Cyrano* peut être utilisé en français, en référence à Cyrano de Bergerac.

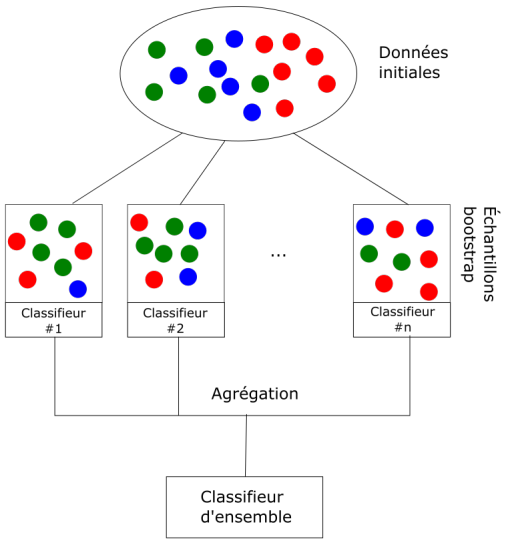


FIGURE 2.11 – Schéma général illustrant l'approche d'agrégation bootstrap.

Cette approche est intéressante dans les cas où les algorithmes d'apprentissages ne sont pas stables, c'est-à-dire quand les modèles subissent de fortes modifications lorsque les données d'entrée varient légèrement. En effet, le fait de générer plusieurs modèles avec différentes données d'entrée proches, mais différentes, et de les agréger permet d'améliorer leur performance. Cependant, cela est conditionné à la taille du jeu de données. En effet, le risque d'écarter –en moyenne– environ un tiers des données d'entrée peut être rédhibitoire dans le cas de petits jeux de données.

Boosting Le *boosting*, bien que similaire au *bagging*, utilise une approche plus complexe que des échantillons bootstrap afin de créer des jeux de données d'apprentissage. L'idée sous-jacente du *boosting* est de concentrer la construction de nouveaux modèles sur les cas complexes, c'est-à-dire les cas pour lesquels les précédents modèles obtiennent de mauvais résultats. Pour ce faire, un poids est associé à chaque instance. Les instances mal classées, ou possédant un écart élevé avec la valeur cible, sont associées à un poids plus élevé. L'algorithme de classification doit bien entendu pouvoir prendre en compte ces poids dans la construction du modèle. La figure 2.12 présente un exemple d'ensemble d'arbres de décision réalisé par *boosting*.

Stacking Enfin, le *stacking* (empilage) de modèle constitue un troisième type de méthode d'ensemble. Contrairement aux deux approches précédentes, ici, un modèle est construit sur la base des résultats d'autres modèles. Ces modèles peuvent être différents conceptuellement les uns des autres – arbres de décision, SVM, etc. –, ou être construits sur la même base conceptuelle avec une variation de leurs paramètres. La figure 2.13 présente une vision globale du stacking. Observons que la sortie du modèle est obtenue par l'application d'un *méta classifieur* sur les résultats des autres modèles. Le rôle de ce dernier classifieur est d'agréger les résultats obtenus,

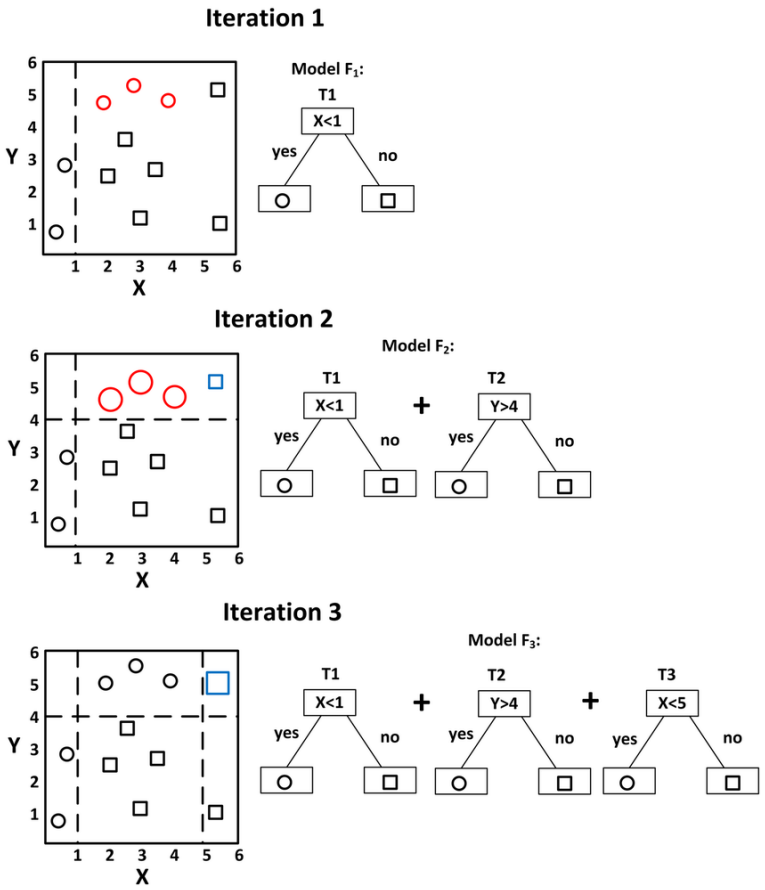


FIGURE 2.12 – Exemple de boosting [ZMD+18]. Ici, trois modèles sont construits. À la première itération, l'arbre de décision se trompe sur trois instances, colorées en rouge. Le poids de ses instances dans la construction modèle suivant est donc incrémenté. L'ajout de deuxième modèle améliore le modèle global, qui ne se trompe plus que sur une instance. La procédure est ainsi répétée une troisième fois, et un *bon* modèle est obtenu.

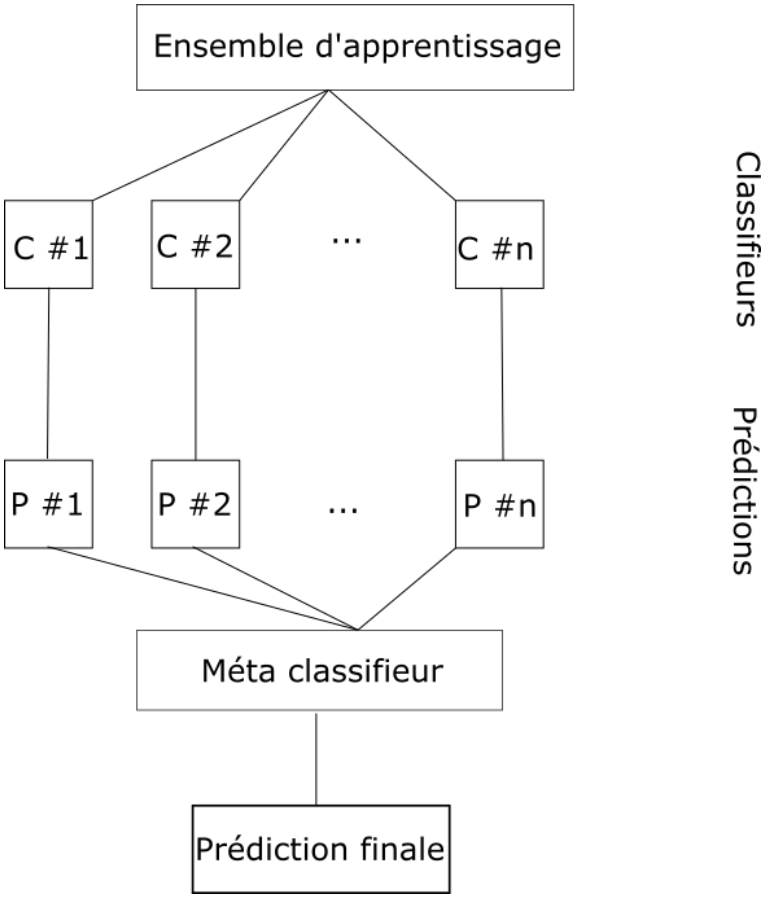


FIGURE 2.13 – Exemple de stacking de modèles. Ici, $C\#1, C\#2, \ldots, C\#n$ représentent n modèles différents. Leurs sorties respectives sont agrégées par un méta modèle. Ce dernier peut, par exemple, être un modèle de régression logistique binaire.

afin d'obtenir une sortie unique. Nous mentionnons le cas de la régression logistique dans notre exemple, dans la mesure où il s'agit d'un classifieur intéressant pour ce type de tâches, prenant en entrée une série de variables binaires ou continues, et retournant une classe parmi 0 et 1. D'autres approches peuvent bien sûr être utilisées en tant que méta modèle, rendant l'approche flexible.

2.2.4.2 Méthodes d'ensemble et arbres de décision

Les forêts d'arbres aléatoires (ou *Random Forests* sont parmi les méthodes d'ensemble les plus populaires dans la littérature, utilisant l'approche de *bagging*. Il s'agit d'une approche présentant de nombreux avantages, parmi lesquels on retrouve, au-delà de la performance, certaines garanties théoriques. L'approche la plus souvent rencontrée est celle de Léo Breiman [Bre01], aussi retrouvée sous le nom de RF-RI, pour Random Forests-Random Inputs (forêts aléatoires à variables d'entrée aléatoires).

De manière concrète, la méthode consiste en la création de plusieurs ensembles d'apprentissage, provenant d'un échantillon aléatoire de l'ensemble de données initial. Ces ensembles sont décrits par un échantillon de leurs attributs, et par leur classe d'appartenance. Un arbre de décision

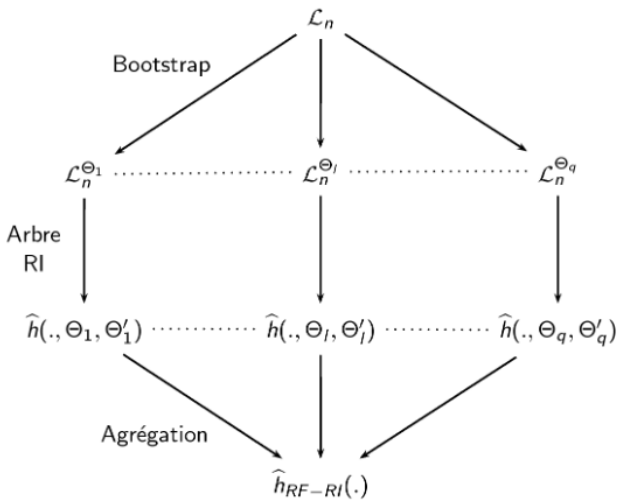


FIGURE 2.14 – Principe général de l'approche des forêts aléatoires [GP17]. Ici Θ' représente le tirage aléatoire sur les attributs.

est par la suite entraîné sur chacun de ces sous-ensembles. Un exemple schématique est présenté figure 2.15. Formellement, le principe général est de construire un ensemble de prédicteurs $\hat{h}$ à partir de plusieurs échantillons *bootstrap* $\mathcal{L}_n^{\Theta_i}$ de l'ensemble d'apprentissage $\mathcal{L}_n$, pour ensuite agréger l'ensemble de leurs prédictions. Par ailleurs, contrairement aux arbres de décision simples que nous avons décrits section 2.2.3, la partition à chaque noeud n'est évaluée que sur un sous-ensemble des attributs, tirés aléatoirement sans remise. Ce principe général est résumé figure 2.14.

Dans le cas de la classification, cette agrégation correspond au vote à majorité. Soit $_i$ le $i$ème modèle, chaque $_i$ étant construit sur un échantillon bootstrap différent, $\hat{y}$ la prédiction de l'ensemble et $C = (c_1, c_2, \cdots, c_n)$ les étiquettes possibles des objets. Le vote majoritaire est décrit formellement équation 2.18.

$$\hat{y}(x) = argmax_{c \in C} \sum_{i=1}^{M} \mathbb{1}(_i(x) = c) \tag{2.18}$$

Cette méthode d'agrégation est valide dans le cas où chaque modèle de l'ensemble renvoie une prédiction de classe d'appartenance. Certains modèles pouvant retourner une probabilité d'appartenance, une autre possibilité est d'effectuer la moyenne arithmétique de ces probabilités et prédire la classe la plus probable. Le terme de *soft voting* est utilisé dans ce cas [Zho12].

La base théorique de cette agrégation est retrouvée dans le théorème du jury de Nicolas de Condorcet. Soit un groupe d'individus – correspondants à nos estimateurs – souhaitant prendre une décision. Si chaque individu a une probabilité $p > \frac{1}{2}$ de faire le «bon» choix, et que cette probabilité est indépendante de celles des autres votants, alors augmenter le nombre d'individus augmente la probabilité que le vote à la majorité absolue soit le bon. De plus, quand le nombre d'individus tend vers l'infini, cette probabilité tend vers 1. Dans le cas des méthodes d'ensemble, il est souhaitable que chaque estimateur ait une probabilité $p > \frac{1}{2}$ de retourner la bonne classe d'appartenance afin que cette approche soit efficace.

L'algorithme des forêts aléatoires, présenté par Leo Breiman, consiste en la création d'une multitude d'arbres décisionnels en fonction d'un ensemble de données d'entraînement qui voteront pour classifier de nouvelles données [Bre01]. L'algorithme général se décompose ainsi [Gen10] [Bre01] :

Algorithme 1 : Algorithme de construction d'une forêt aléatoire.

Entrées : Un nombre d'arbres M, un nombre d'attributs F, un ensemble d'apprentissage $\mathcal{L}$

Sortie : Un ensemble d'arbres $(t_1, t_2, \cdots, t_M)$

```
sortie ← {}
pour i ← 0 à M faire
    Construire un échantillon bootstrap en tirant TailleEchantillon fois, avec remise,
     depuis l'ensemble d'apprentissage.
    pour chaque nœud de l'arbre faire
        Choisir F variables sur lesquelles baser la décision au nœud.
        Calcul de la meilleure partition basée sur ces F variables dans l'échantillon
         bootstrap.
        Chaque arbre est construit complètement.
    fin
    Ajouter l'arbre nouvellement construit à sortie
fin
retourner sortie
```

Cette approche permet d'obtenir meilleures performances et une plus grande généralisation du modèle, tout en contrebalançant l'instabilité des arbres de décision simples, une forêt d'arbre étant moins sensible au bruit. Par ailleurs, nous avons vu dans la section 2.2.3 que les arbres de décision simples peuvent, dans certains cas, être sensibles au surajustement. Les forêts d'arbres aléatoires limitent ce risque. Plus précisément, le fait d'ajouter des arbres à la forêt limite le risque de surajustement [Bre01]. Cela ne signifie pas pour autant que les ensembles d'arbres ne peuvent pas donner des modèles surajustés, comme l'ont mis en avant certains auteurs [GGCM08] [Seg04].

Le *bagging* utilisé dans cette méthode peut être considéré comme faisant partie d'une plus grande famille de méthodes de *randomisation* [GEW06]. En effet, alors que le *bagging* introduit des processus aléatoires par le biais de l'échantillonnage *bootstrap*, d'autres approches sont possibles. Toutes ont en commun la perturbation, d'une manière ou d'une autre, lors de la construction des modèles, notamment au niveau du choix du meilleur branchement. Comme indiqué par Geurts *et al.*, il est intéressant de noter que détériorer l'optimalité des algorithmes déterministes de choix de branchement peut s'avérer être productif et améliorer les performances du modèle.

Ces derniers proposent une approche de construction de forêts d'arbres complètement aléatoires [GEW06], où le *cut-point* (valeur/seuil de branchement) est choisi de manière complètement aléatoire à chaque noeud. Cette approche, nommée Extremely Randomized Trees [9], est au cœur de cette thèse, et nous y reviendrons dans les chapitres 4 et 5. L'algorithme

9. Le terme Extra-Trees est aussi retrouvé, à l'origin de l'acronyme ET.

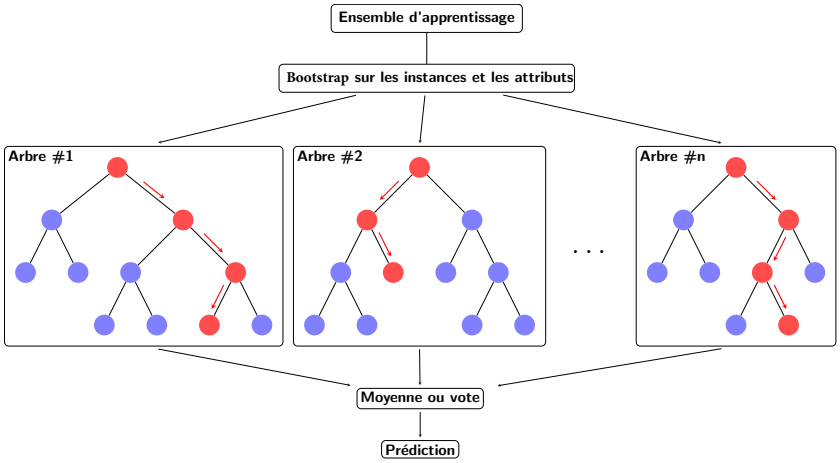


FIGURE 2.15 – Représentation schématique d'une forêt d'arbres aléatoires. Traduit de [Cas19].

2 présente la méthode permettant de construire un arbre extrêmement aléatoire. La fonction $Pick\_a\_random\_split(\mathcal{L}, a_i)$ peut être implémentée de diverses manières, en fonction notamment du type des données considérées pour la variable a_i. L'essentiel est ici que cette fonction retourne deux sous-ensembles disjoints à partir des données sur la base de la variable a_i. L'algorithme 3 présente comment construire une forêt d'arbres extrêmement aléatoires. Il s'appuie sur la définition d'un arbre extrêmement aléatoire simple, dont la construction est décrite dans l'algorithme 2.

Algorithme 2 : Fonction *Build_an_extra_tree*

Entrées : $\mathcal{L}$, n_{min}, K
Sortie : Un arbre t

si $|\mathcal{L}| < n_{min}$ *ou tous les attributs candidats sont constants ou les étiquettes sont constantes* **alors**
  retourner *Une feuille étiquetée par (i) la moyenne dans le cas d'une régression, (ii) les fréquences dans le cas de la classification*
fin
sinon
  Tirer sans remise K attributs $\{a_1, \ldots, a_k\}$;
  Générer K partitions $\{s_1, \ldots, s_K\}$, où $s_i = Pick\_a\_random\_split(\mathcal{L}, a_i), \forall i = 1, \ldots, K$;
  Choisir une partition s_* en fonction du critère de séparation choisi;
  Diviser $\mathcal{L}$ en sous-ensembles $\mathcal{L}_l$ et $\mathcal{L}_r$ à partir du test s_* ;
  Construire $t_l = Build\_an\_extra\_tree(\mathcal{L}_l)$ et $t_r = Build\_an\_extra\_tree(\mathcal{L}_r)$;
  Créer un noeud avec la partition s_*, y attacher t_l et t_r ;
  retourner t;
fin

Bien que les valeurs de branchement soient sélectionnées de manière aléatoire, K attributs sont choisis à chaque noeud. K valeurs aléatoires sont donc sélectionnées, et le branchement choisi est celui maximisant un score, tel que le gain d'information. Ainsi, bien que les arbres

Algorithme 3 : Fonction *Build_an_extra_tree_ensemble*

Entrées : $\mathcal{L}$, M, n_{min}, K
Sortie : Un ensemble d'arbres $\mathcal{T} = \{t_1, \ldots, t_M\}$

pour $i = 1$ *à* M **faire**
  Générer un arbre $t_i = Build\_an\_extra\_tree(\mathcal{L}, n_{min}, K)$ et l'ajouter à $\mathcal{T}$;
fin
retourner $\mathcal{T}$

soient complètement aléatoires, un contrôle est toujours effectué sur les branchements. Bien que la complexité temporelle reste en O(nlog(n)), les auteurs rappellent qu'au vu de la relative simplicité de la procédure de branchement, cette complexité tend à être plus faible que pour d'autres approches à base d'arbres aléatoires.

Pour une revue plus complète tant sur les arbres de décision simples que sur les forêts d'arbres aléatoires, [Lou14] constitue une référence. Cette dernière comprend notamment un chapitre complet sur la notion d'importance de variables, que nous n'avons pas abordé jusqu'ici. En effet, les forêts d'arbres aléatoires ont de nombreuses propriétés très intéressantes, permettant notamment de quantifier l'importance d'une variable dans un processus de décision, et *in fine*, effectuer une sélection des meilleures variables pour une tâche donnée.

Dans cette section, nous avons décrit de manière succincte quelques grandes approches de classification supervisée. Nous avons vu que de nombreux types d'algorithmes coexistent, avec chacun leurs forces et leurs faiblesses. Cela met en exergue le fait que malgré la puissance de ces méthodes d'apprentissage, une expertise humaine reste nécessaire [10], afin de faire le tri, de sélectionner la meilleure approche pour le problème donné, ainsi que de porter un regard critique sur les résultats obtenus. En effet, bien que ces algorithmes sont pour la plupart très efficaces – au moins sur les données correspondant aux cas où ils peuvent l'être –, il n'est pas attendu qu'ils soient *trop* efficaces. Nous avons vu notamment au cours de cette section la notion de surajustement, à détecter absolument avant de mettre en production en modèle ou plus généralement d'exploiter ses résultats. De plus, la supervision humaine reste de mise concernant le choix et l'optimisation des paramètres, parfois d'une importance majeure pour certains types de méthodes. Enfin, bien que cela soit sujet à débat, l'interprétabilité des modèles est capitale dans certains cas de figure, afin de comprendre comment ce dernier arrive à sa prédiction à partir des données d'entrée. Parmi les méthodes décrites ici, les arbres de décisions sont particulièrement intéressants concernant ce point sur l'interprétabilité, et sont particulièrement appréciés dans la communauté biomédicale.

2.3 Classification non supervisée ou clustering

Nous avons vu précédemment les méthodes de classification supervisée, permettant d'obtenir des modèles capables de prédire une classe d'appartenance après une phase d'entraînement sur des données étiquetées. Cependant, de telles données ne sont pas toujours disponibles, et la tâche

10. Des avancées dans le sens de la diminution de l'importance de l'expertise humaine sont cependant faits, dans le cadre de l'*Automated Machine Learning, ou AutoML*.

d'étiquetage est une tâche coûteuse et chronophage. De plus, la question d'une *vérité* absolue peut être questionnée. En effet, cette vérité peut refléter un biais des personnes réalisant l'étiquetage. De plus, il peut y avoir une volonté à forcer les données à respecter une certaine catégorisation pour *rentrer dans des cases* attendues.

Par ailleurs, même dans le cas où des données étiquetées existent, il peut être très intéressant de trouver des groupes *naturels* dans les données. C'est par exemple le cas par exemple dans la nosographie, visant à catégoriser les maladies. Bien que les étiquettes existent, la tâche consiste ici à établir une classification – potentiellement nouvelle – des maladies qui ne soient pas influencée par les connaissances contemporaines des spécialistes, potentiellement biaisées.

Les méthodes d'apprentissage non supervisé, que nous allons décrire, permettent de lever cette nécessité. Les méthodes de clustering ont pour objectif de mettre en évidence des groupes homogènes d'objets au sein de données, nommés *clusters*. Ces méthodes sont très intéressantes, dans la mesure où elles permettent de découvrir des ensembles d'objets partageant des caractéristiques communes qui n'ont jusqu'alors potentiellement pas été découverts, par exemple à cause de la quantité de données et/ou de leur dimension, rendant impossible leur analyse par des humains. En effet, il semblerait que la limite humaine concernant l'interprétation de relations entre objets se situerait autour de descriptions par quatre variables [HBMB05].

Ces approches sont par ailleurs très intéressantes au-delà de la non-nécessité d'étiqueter les données. En effet, il existe des problèmes où ces étiquettes ne sont pas nécessaires. C'est par exemple le cas dans la recherche de groupes de clients dans le commerce en ligne ou encore la recherche de familles de molécules ayant une action similaire, dans le but de recibler certaines molécules.

Ces algorithmes sont donc particulièrement utiles dans les phases d'exploration de données, et ont déjà donné de bons résultats, notamment dans la classification des formes d'une maladie [SKS+15], ou encore dans l'identification de comportements [CNP06].

Bien que de nombreuses méthodes sont décrites dans la littérature, chacune possède ses points forts et ses limites. Dans un processus d'exploration de données, il est donc important de débuter par une phase de description du problème et des données brutes disponibles, afin d'effectuer le choix le plus cohérent possible.

Le choix de l'algorithme dépend donc fortement des contraintes du projet, et l'existence d'un algorithme parfait dans tous les cas de figure relève de l'utopie. En effet, le théorème du *No Free Lunch* (*NFL*) présenté par Wolpert et Macready [WM97], nous dit la chose suivante :

Si un algorithme donne des résultats particulièrement bons en moyenne pour une classe de problèmes, alors ce dernier doit être moins bon que d'autres algorithmes sur les problèmes hors de cette classe. En particulier, si un algorithme donne de meilleurs résultats qu'une recherche aléatoire pour une classe de problèmes, alors il doit donner des résultats moins bons qu'une recherche aléatoire pour les classes de problèmes restantes.

Cependant, la recherche d'un algorithme versatile qui serait utilisable avec un minimum de modifications dans plusieurs classes de problèmes reste possible. Il s'agit ici de notre objectif, sur lequel nous avons travaillé au cours de cette thèse.

Lors du *design* d'une tâche de clustering, il faut donc débuter par une première étape de description des données, et se poser les questions suivantes :

— De quel type de données s'agit-il ?
— Dans le cas de données sous forme de vecteurs, de quels types – *i.e.* continues, qualitatives, ordinales – sont les variables décrivant les objets à étudier ? Toutes les variables sont-elles du même type, auquel cas on parle de données homogènes, ou non ?
— Existe-t-il des données manquantes ? Si oui, dans quelle mesure ?
— Une normalisation des attributs est-elle nécessaire, voire possible ?

Les réponses à ces questions permettent de guider le choix de la méthode.

Ces méthodes appartiennent généralement à quelques grandes catégories, que nous allons présenter dans ce chapitre. Après un point sur les notions de distance et de similarité dans la sous-section 2.3.1, trois grands types d'approches seront exposés, dans les sous-sections 2.3.2, 2.3.3 et 2.3.4. Nous terminerons par une discussion sur les méthodes d'évaluation d'un clustering, sous-section 2.3.7.

2.3.1 Notions de similarité et distance

La notion de distance permet de quantifier la proximité entre objets. Formellement, une distance est une fonction d applicable sur des paires d'éléments d'un ensemble S. Cette fonction est une métrique si pour tout $x, y \in S$ les conditions suivantes sont respectées :

1. $d(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y}$ (identité des indiscernables)
2. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ (symétrie)
3. $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$ (inégalité triangulaire)

De nombreuses mesures de distances sont présentées dans la littérature, si bien que des ouvrages entiers y sont dédiés [DD14]. Parmi les plus courantes, on retrouve tout particulièrement les distances euclidiennes et de Manhattan :

$$\text{La distance euclidienne : } ||\mathbf{x} - \mathbf{y}||_2 = \sqrt{\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{y}_i)^2} \tag{2.19}$$

$$\text{La distance de Manhattan : } ||\mathbf{x} - \mathbf{y}||_1 = \sum_{i=1}^{n} |\mathbf{x}_i - \mathbf{y}_i| \tag{2.20}$$

Ces deux premières distances correspondent en fait à des cas particuliers de la distance de Minkowski :

$$\text{La distance de Minkowski : } ||\mathbf{x} - \mathbf{y}||_i = \sqrt[p]{\sum_{i=1}^{n} |\mathbf{x}_i - \mathbf{y}_i|^i} \tag{2.21}$$

Certaines mesures ne respectent pas une ou plusieurs de ces conditions. S'il s'agit de l'inégalité triangulaire, on parle par exemple de semi-métrique. Dans ce document, nous avons utilisé le terme de dissimilarité, plus général. Il correspond à toute fonction s'appliquant à une paire d'élément, et qui reflète un éloignement entre ces derniers, sans pour autant respecter toutes les conditions pour s'il s'agisse d'une métrique.

Le choix d'une mesure de distance dans un problème donné est complexe et nécessite une compréhension fine à la fois du dit problème et des algorithmes utilisés. Tout d'abord, le type de données, et en particulier le type des variables décrivant les objets, est un paramètre fondamental dans le choix d'une distance. En effet, certaines mesures ne sont pas applicables pour certains types de données. Par exemple, la distance euclidienne vue ci-dessus – et plus généralement les normes L_p – n' est pertinente que si elles sont appliquées sur des objets d'un espace vectoriel orthonormé. Il existe de nombreux cas où les variables descriptives ne rentrent pas dans cette catégorie. Nous ferons l'impasse sur les distances spécifiques aux chaînes de caractères dans ce document (distance d'édition, etc.), et nous concentrerons plutôt les sur cas où les variables descriptives sont numériques.

Une variable numérique peut être de plusieurs types. Tout d'abord, il peut s'agir d'une variable continue. Ce type de variables peut prendre une valeur dans l'ensemble des réels, et donc parmi une infinité de valeurs possibles. Il s'agit de variables telles que la température, ou encore la masse. Ces valeurs étant définies dans $\mathcal{R}$, la notion d'ordre y est présente. Les variables qualitatives [11]. quant à elles sont des variables dont les valeurs peuvent être soit des modalités, soit des niveaux, auquel cas on parle de variable ordinale. Il s'agit bien là de deux cas bien différents, tant conceptuellement que pour la pertinence des algorithmes. De nombreux exemples peuvent être retrouvés dans les données, telles que le sexe ou le statut tabagique – et de manière générale l'ensemble des variables encodant la présence ou l'absence d'un caractère –. Ces variables sont ici encodées en modalités. Des exemples de variables dont les valeurs sont des niveaux se retrouvent notamment dans les cas où une gradation est faisable. C'est le cas notamment des variables encodées sous forme de gradation «Faible»-"Moyenne"-"Élevé». La notion d'ordre y est importante, là où elle n'a pas de sens pour une variable dont les valeurs sont des modalités : habiter à Paris n'est pas exemple pas supérieur ou inférieur au fait d'habiter à Lille.

Des mesures de distance et de similarité sont définies pour les données décrites par ce type de variables. Le *simple matching coefficient* est l'une des approches les plus simples pour cette tâche. Elle correspond au rapport entre le nombre de variables prenant la même valeur pour deux objets différents et le nombre de variables. Plus précisément, on a :

$$SMC(\mathbf{x}_1, \mathbf{x}_2) = \frac{f_{00} + f_{11}}{f_{00} + f_{11} + f_{01} + f_{10}} \tag{2.22}$$

, où f_{00} représente le nombre d'attributs dont les valeurs sont 0 pour $\mathbf{x}_1$ et $\mathbf{x}_2$, f_{11} le nombre d'attributs dont les valeurs sont 1 pour $\mathbf{x}_1$ et $\mathbf{x}_2$, etc. On notera ici que ce coefficient est adapté dans le cas de données binaires, *i.e.* avec uniquement deux modalités. Cette approche est quelque peu simpliste, dans la mesure où elle considère toutes les modalités de la même manière. En pratique, ce n'est pas toujours le cas : la présence d'un attribut peut apporter plus d'information que son absence – et vice-versa. Cependant, cette vision neutre peut être intéressante dans certains contextes. En particulier, certaines variables peuvent bénéficier de cette neutralité. Il s'agit par exemple du cas d'une variable encodant le sexe : le fait que le sexe masculin soit encodé en tant que 1 ou 0 ne devrait pas avoir d'influence sur les similarités obtenues.

11. Le terme de variables *catégorielle* est aussi utilisé.

	Values of character k			
Individual i	+	+	−	−
j	+	−	+	−
s_{ijk}	1	0	0	0
δ_{ijk}	1	1	1	0

FIGURE 2.16 – Calcul de s_{ijk} dans le cas de variables binaires [Gow71].

La distance de Jaccard est très similaire à cette mesure. L'équation 2.23 présente le calcul de cette distance.

$$d_J(\mathbf{x}, \mathbf{y}) = 1 - \frac{|\mathbf{x} \cap \mathbf{y}|}{|\mathbf{x} \cup \mathbf{y}|} \tag{2.23}$$

Fait intéressant, cette distance est une métrique sur les ensembles finis [Kos19]. De nombreuses autres variantes ou mesures proches existent, telles que le coefficient de Sørensen-Dice.

Un cas plus complexe est celui des données mixtes, à savoir réunissant différents types de variables. Dans ce cas, il est nécessaire d'appliquer des mesures de (dis)similarités spécifiques. Là aussi, des mesures ont été proposées, parmi lesquelles la distance de Gower [Gow71], la plus couramment retrouvée dans ce cas. Commençons par décrire la similarité de Gower, dont le calcul est présenté équation 2.24, où i et j représentent deux objets, k l'une des p variables les décrivant, et δ_{ijk} une indication de la possibilité de comparer i et j sur la variable k. δ_{ijk} peut donc prendre deux valeurs, 0 ou 1. Lorsque cette valeur est nulle pour toutes les variables, alors la similarité n'est pas définie.

$$s(i,j) = \frac{\sum_{k=1}^{p} \delta_{ij,k} s_{ij,k}}{\sum_{k=1}^{p} \delta_{ij,k}} \tag{2.24}$$

La similarité $s_{ij,k}$ est définie en fonction du type de la variable k. On a :

- Dans le cas d'une variable qualitative, alors $s_{ij,k} = 1$ si cette variable prend la même modalité pour les objets i et j, et est nulle dans le cas contraire.
- Dans le cas d'une variable quantitative, $s_{ij,k} = 1 - \frac{|\mathbf{x}_i - \mathbf{x}_j|}{R_k}$, où R_k représente l'étendue de la variable k.

Notons le cas particulier d'une variable binaire, qui est une variable qualitative ne prenant que deux valeurs et pouvant être interprétée comme présence ou absence d'un caractère. La similarité définie par Gower peut dans ce cas prendre différentes valeurs selon les combinaisons observées, présentées figure 2.16.

Afin d'obtenir une distance, il est possible d'effectuer la transformation $\sqrt{1 - s_{ij}}$. Cette transformation permet d'obtenir une métrique [Gow71]. Bien que la prise en compte des variables ordinales n'est pas considéré dans la distance originale, une extension a été proposée par la suite pour ce faire [Pod99], que l'on retrouve parfois soir le nom de variante de Podani.

Bien qu'intéressante dans de nombreux cas, la distance de Gower présente quelques limitations. En particulier, il s'agit d'une mesure sensible aux points aberrants dans les données continues, ainsi qu'à la normalité de ces variables. Un prétraitement peut donc être nécessaire afin d'en tirer parti le mieux possible.

Enfin, nous avons décrit jusqu'à présent des distances qui ne dépendent pas des données, en ce sens que tous les objets et leurs variables sont traités de la même façon. Certaines mesures ont été définies qui dépendent des données. L'une des plus connues est la distance de Mahalanobis [Mah36], permettant de calculer la distance entre un objet et une distribution. La distance $d(\mathbf{x})$ d'un objet $\mathbf{x}$ à un ensemble d'objets ayant pour valeurs moyennes $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$ est obtenue par la relation présentée équation 2.25. Dans cette équation, $\sum$ représente la matrice de covariance des données présentes.

$$d(\mathbf{x}) = \sqrt{(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)} \tag{2.25}$$

Enfin, les approches à base d'estimation de masse permettent elles aussi d'obtenir des distances dépendant des données. Nous y reviendrons plus particulièrement au sein des chapitres 4 et 5.

Comme nous avons pu l'entrevoir ici, la littérature est riche d'approches permettant de calculer des distances entre objets, toutes avec leurs points forts et points faibles. Ces distances sont cruciales dans de nombreuses méthodes, notamment dans les méthodes de clustering qui nous intéresseront ici, et leur choix n'est pas toujours aisé.

2.3.2 Clustering hiérarchique

Les méthodes de clustering hiérarchique permettent d' obtenir une hiérarchie de clusters. Cette hiérarchie constitue un arbre binaire, le **dendrogramme**.

Cette hiérarchisation des objets peut se faire selon deux approches :

- L'approche **agglomérative**, où, à l'état initial, l'ensemble des objets forme leur propre cluster, et où les paires de clusters sont fusionnées à chaque itération.
- L'approche **divisive**, où tous les objets font partie du même cluster à l'état initial. À chaque itération, les clusters se divisent, jusqu'à obtenir, éventuellement, des *singletons*.

Quelle que soit l'approche, chaque itération correspond à un nouveau niveau du dendrogramme. Afin de déterminer quels objets séparer ou quels clusters fusionner, deux éléments sont nécessaires : une mesure de la distance ou de la similarité entre deux objets, et une méthode afin de calculer la distance ou la similarité entre groupes d'objets.

Choix de la métrique En pratique, et dans le cadre des méthodes présentées ici, les distances euclidiennes et de Manhattan vues précédemment sont couramment utilisées. On peut aussi retrouver la distance euclidienne au carré $||\mathbf{x}-\mathbf{y}||_2^2 = \sum_{i=1}^{n}(\mathbf{x}_i - \mathbf{y}_i)^2$. Ces mesures sont adaptées à des données numériques continues. Dans le cas de données qualitatives ou hétérogènes, une autre mesure de distance est nécessaire.

Choix du critère de regroupement Le critère de regroupement (*linkage criterion*) définit la manière de calculer la distance entre groupes d'objets. Certains de ces critères sont présentés équations 2.27, 2.26 et 2.28.

$$\text{Complete linkage} : max\{d(\mathbf{x}, \mathbf{y}) : \mathbf{x} \in C_1, \mathbf{y} \in C_2\} \tag{2.26}$$

$$\text{Average linkage} : \frac{1}{|C_1||C_2|} \sum \mathbf{x} \in C_1 \sum y \in C_2 d(\mathbf{x}, \mathbf{y}) s \tag{2.27}$$

$$\text{Single linkage} : min\{d(\mathbf{x}, \mathbf{y}) : \mathbf{x} \in C_1, \mathbf{y} \in C_2\} \tag{2.28}$$

Les avantages de ces méthodes sont :
- Une hiérarchie de clusters est être obtenue, permettant de guider le clustering et son interprétation.
- Bien que la détermination de l'appartenance d'un objet à un groupe nécessite de fixer un nombre de clusters, l'obtention du dendrogramme ne nécessite pas ce paramètre.

Cette approche est cependant sensible au choix de la mesure de distance/similarité et du critère de regroupement.

2.3.3 Clustering à base de barycentres

Les méthodes clustering à base de barycentre (*centroid based* en anglais) consistent à définir k points qui représentant les centres des clusters. Ces points sont ensuite mis à jour progressivement au fil des itérations, jusqu'à ce qu'une condition d'arrêt soit satisfaite, telle qu'une convergence. L'attribution de chaque objet à son cluster se fait par le biais d'une heuristique, telle que la proximité à un centre.

L'une des méthodes de clustering à base de barycentre la plus courante est celle des k-moyennes, développée par Mac Queen et al. [M$^+$67] en 1967.

Ici, $\mathbf{x}_1 ..., \mathbf{x}_n$ représentent des vecteurs de dimension d, et μ_i la moyenne des points dans le cluster C_i. L'algorithme des k-moyennes consiste en fait en une minimisation de fonction, cette fonction étant la somme des carrés des distances intra-clusters (*Within Cluster Sum of Squares*, notée *WCSS*), décrite équation 2.29.

$$WCSS = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} ||\mathbf{x}_j - \mu_i||^2 \tag{2.29}$$

L'algorithme des k-moyennes est présenté dans l'algorithme 4.

Initialisation Cette étape d'initialisation consiste à introduire k points $\mu_1, ..., \mu_k \in \mathbb{R}^d$, qui seront considérés comme les centres des k clusters. Une approche consiste à effectuer l'initialisation des k points aléatoirement. Cette approche présente l'avantage d'être très simple à mettre en œuvre. D'autres approches existent, telles que celle de Kaufman et Rousseeuw [KR09], qui a pour avantage d'effectuer une initialisation des centres de manière déterministe. Une étude comparative de différentes approches d'initialisation est présentée dans [PLL99], où l'initialisation

Algorithme 4 : Algorithme des k-moyennes

Entrées : Un ensemble d'objets X, un nombre de clusters k
Sortie : Un ensemble de k de clusters

tant que *Pas de convergence des* $\mu_1, ..., \mu_k$ **faire**
  Assigner chaque point $\mathbf{x} \in X$ à $argmin_j d(\mathbf{x}, \mu_j)$;
  pour $j = 1$ *à* k **faire**
    $Cluster_j \leftarrow \{\mathbf{x} \in X | \mathbf{x}$ assigné au cluster $j\}$;
    $\mu_j = \frac{1}{|X_j|} \sum_{\mathbf{x} \in X_j} \mathbf{x}$;
  fin
fin

de Kaufman donne de meilleures performances. L'initialisation aléatoire montre cependant de bonnes performances, et reste l'une des approches les plus couramment retrouvées en pratiques.

Assignation Chacun des points x est associé au cluster j dont le centre μ_j est le plus proche. Cette notion de proximité est définie grâce à la mesure de distance/similarité, ce qui montre une fois de plus l'importance du choix de cette mesure. Lorsque chaque observation est associée au centre qui lui est le plus proche, ces derniers sont recalculés.

Cette procédure est répétée jusqu'à convergence, c'est-à-dire lorsque la valeur fonction à minimiser est stabilisée, ou que sa variation est inférieure à un seuil fixé par l'utilisateur.

Bien que cet algorithme très intéressant, notamment de par sa – relative – simplicité, il présente quelques limitations. Tout d'abord, il est nécessaire de fixer un paramètre k, correspondant au nombre de clusters. De plus, une supposition forte est faite concernant la forme des clusters, dans la mesure où cette approche ne permet d'obtenir que des partitions convexes de l'espace [12]. Cette limitation peut cependant être levée, au coût de la performance, en utilisant des noyaux, les *kernel k-means* ayant une complexité bien plus élevée que la version simple.

L'algorithme est par ailleurs dépendant de l'initialisation, et les résultats peuvent donc fortement varier en fonction de la graine (*seed*). Il est possible de rester dans un minimum local, et donc de ne pas trouver le clustering optimal. Afin de limiter ce risque, il est cependant possible de réaliser plusieurs fois un clustering en faisant différentes initialisations.

2.3.4 Clustering à base de densité

Afin de pouvoir identifier des groupes d'objets ne formant pas de partitions convexes de l'espace (telles que les données présentées figure 2.17) , les méthodes de clustering à base de densité peuvent être utilisées. La méthode la plus connue de cette famille est DBSCAN (*Density-Based Spatial Clustering of Applications with Noise* [EKS+96]. L'idée générale de l'algorithme est de parcourir les objets de l'ensemble de données un à un, et de considérer le voisinage de ce point.

12. Dans un espace euclidien, une région convexe est une région où tout segment entre points de la région pris deux à deux est aussi à l'intérieur de la région.

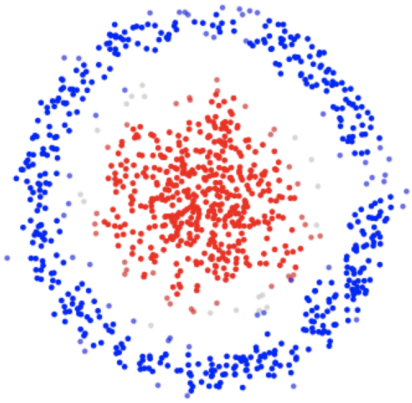


FIGURE 2.17 – Objets appartenant à deux classes, ne formant pas des partitions convexes.

Ce voisinage correspond aux autres objets étant à un distance maximale ϵ fixée par l'utilisateur. Si le nombre d'objets dans ce voisinage est inférieur à une valeur fixée là encore par l'utilisateur, l'objet considéré est traité comme du bruit. Sinon, le cluster est étendu est l'algorithme passe à l'objet suivant.

Deux paramètres sont donc à prendre en compte :
- ϵ, permettant de définir la distance maximale afin de considérer qu'un objet est dans le voisinage d'un autre objet,
- et *minpts*, le nombre minimum d'objets devant se trouver dans l'ϵ-voisinage afin que ceux-ci puissent être considérés comme un cluster.

La distance choisie est couramment la distance euclidienne, bien que d'autres distances peuvent être utilisées. La figure 2.18 présente un exemple de partitionnement avec DBSCAN, avec $minpts = 4$. Dans cet exemple, les cercles autour de chaque objet représentent leur ϵ-voisinage. Les objets rouges se trouvent dans un même cluster. Dans la mesure où ils possèdent au moins *minpts* objets dans leur ϵ-voisinage. Les objets pour lesquels ces deux conditions sont respectées sont nommés *core points*. Les points B et C, bien que dans l'ϵ-voisinage d'au moins un objet du cluster précédent, ne sont pas des *core points*. En effet, ces derniers de possèdent pas *minpts* objets dans ce voisinage. Enfin, le point N ne possède aucun ϵ voisin, et est donc considéré comme du bruit. Les avantages de DBSCAN sont nombreux. Parmi eux, on retrouve :

1. La mise en évidence de clusters de forme et de taille arbitraire est possible.
2. Le nombre de clusters *a priori* n'est pas nécessaire.
3. La possibilité d'identifier le bruit.

Cependant, la détermination de ces paramètres, déterminants dans le résultat, peut-être complexe. De plus, la méthode tend à donner de mauvais résultats dans le cas de clusters de densités variables.

HDBSCAN [CMS13] est une extension de DBSCAN permettant de dépasser ces limitations. De plus, cette méthode a pour avantage d'être une approche retournant une hiérarchie de clusters.

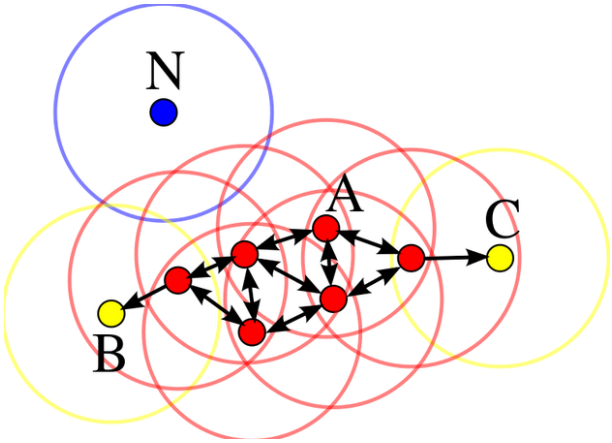


FIGURE 2.18 – Exemple de partitionnement avec DBSCAN. Les points autour de A son les *core points*. Les points B et C ne le sont pas, mais forment un cluster avec le point A, ces derniers étant dans le même voisinage. Le point N est quand à lui un point aberrant, n'étant ni un *core point*, ni un point dans le voisinage d'un autre *core point* [Chi11].

2.3.5 Détection d'anomalies

Nous avons jusqu'à maintenant parlé uniquement du clustering. L'apprentissage non supervisé ne se limite cependant pas à ce type d'approches. L'une des tâches possible dans le cadre non supervisé consiste à détecter des points aberrants, ou anomalies. Il est possible par ailleurs d'étendre cette définition à la détection de nouveautés. On se trouve bien ici dans un cadre où les classes ne sont pas connues, et ne sont, en fait, pas voulues.

Ces approches peuvent être utilisées par exemple dans le système bancaire, afin de détecter de manière précoce des tentatives de fraude ; ou encore dans le domaine médical, afin de détecter des situations anormales.

La détection d'anomalies a ses difficultés propres. Par exemple, la définition même d'anomalie peut être un sujet de discussion à part entière. Nous évoquions précédemment l'exemple de l'application de ces approches dans divers cadres. Dans le cas de l'application médicale, une faible variation de température peut être – doit être – considérée comme anomalie. Dans une application météorologique par exemple cependant, cette même faible variation peut ne pas être significative.

Une bonne revue des méthodes de détection d'anomalies peut être retrouvée dans [CBK09]. Cette ressource commence cependant à être datée, et bien que complète dans la présentation du domaine, ne présente pas des innovations plus récentes, et clés dans le domaine.

Parmi les méthodes non présentées dans cette revue, car plus récentes, on retrouve l'algorithme de forêt d'isolation (*isolation forest*) [LTZ08]. La brique de base de cet algorithme est l'arbre d'isolation. Son fonctionnement est relativement intuitif : il s'agit d'utiliser l'information contenue dans les partitions successives de l'espace, réalisées par des structures d'arbres, afin de quantifier le niveau d'anomalie de chaque objet. Pour ce faire, l'idée est d'utiliser le nombre de partitions nécessaires afin d'isoler un objet – ce qui correspond par ailleurs à la longueur du chemin entre la racine de l'arbre et cet objet, notion que les auteurs ont préféré utiliser dans leur

article –. Cette longueur est nommée $h(\mathbf{x})$. Le score d'anomalie d'un objet $\mathbf{x}$ d'un jeu de données $X = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n\}$, noté $s(\mathbf{x}, n)$, est calculé selon la relation présentée équation 2.30.

$$s(\mathbf{x}, n) = 2^{-\frac{E(h(\mathbf{x}))}{c(n)}} \tag{2.30}$$

Dans cette équation, $E(h(\mathbf{x}))$ correspond à la moyenne des $h(\mathbf{x})$ pour l'ensemble des arbres de la forêt. Cette valeur est normalisée par $c(n)$, la hauteur moyenne d'un arbre binaire pour n objets. On s'attend ici à ce que la longueur moyenne d'un chemin pour atteindre une anomalie soit plus faible que la longueur moyenne de l'arbre. Ainsi, dans le cas d'une anomalie, $E(h(\mathbf{x})) \to 0$, s tend donc vers 1 dans ce cas. Une valeur de score inférieure à 0.5 correspond quant à elle à une absence d'anomalie.

Cette méthode est particulièrement intéressante, dans la mesure où elle combine diverses approches que nous avons précédemment décrites dans le cadre supervisé, afin de les utiliser dans le cadre non supervisé. L'utilisation de structures d'arbres permet ici la détection d'anomalies de manière non , en passant outre une limite de nombreuses autres méthodes, à savoir la difficulté de donner des résultats pertinents dans le cas de données de fortes dimensions. De plus, il s'agit d'un algorithme rapide, ayant une très faible empreinte mémoire.

Il ne s'agit bien sûr pas de la seule méthode récente de détection d'anomalie. Nous avons cependant fait le choix de présenter cette dernière dans la mesure où nous allons la retrouver dans la suite de ce document.

2.3.6 Méthodes d'ensemble

Tout comme pour les approches supervisées, il existe des méthodes d'ensemble dans le cadre non supervisé. L'objectif est toujours de combiner un ensemble de modèles afin d'obtenir un clustering de meilleure qualité qu'un modèle seul. Dans le cadre non supervisé, ces approches d'ensemble sont particulièrement intéressantes dans la mesure où différents algorithmes peuvent donner des résultats drastiquement différents sur un même jeu de données, en imposant une certaine structure aux groupes retrouvés dans les données. Ceci est d'autant plus intéressant qu'ici, les données n'étant pas étiquetées, plusieurs clusterings peuvent coexister, et tous être plus ou moins pertinents en fonction de la tâche. De plus, comme pour le cadre supervisé (voir le théorème du *No Free Lunch*), il n'existe pas d'approche parfaite qui peut donner de *bons* résultats sur toutes les données possibles. Le fait d'agréger les résultats de plusieurs clusterings et donc d'autant plus pertinent [AW19].

Notons que lorsque nous parlons de l'obtention d'un clustering de meilleure qualité par le biais d'une méthode d'ensemble, la définition de cette qualité est plus complexe que dans le cas du cadre supervisé. En effet, ces processus étant par nature exploratoires, différents clusterings des objets peuvent être retrouvés, sans pour autant que certains soit *objectivement* meilleurs que d'autres. Cependant, la mise en œuvre d'une méthode de clustering d'ensemble peut donner un clustering de meilleure qualité en ce sens que ce dernier est plus robuste et plus pertinent pour la tâche d'exploration donnée.

Comme précisé dans [AW19], l'agrégation de modèles n'est ici pas aussi simple que dans le cadre supervisé. L'un des problèmes principaux concerne la méthode d'agrégation. Là où le vote

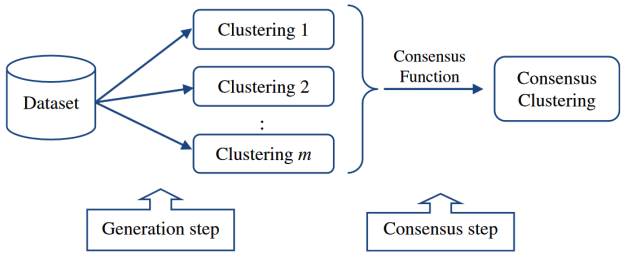


FIGURE 2.19 – Exemple générique de méthode d'ensemble dans le cadre non supervisé. Un ensemble de m clusterings différents sont obtenus avec différentes méthodes, avant d'arriver à un clustering final par l'application d'une méthode de consensus, agrégeant les résultats des clusterings individuels [VPRS11]

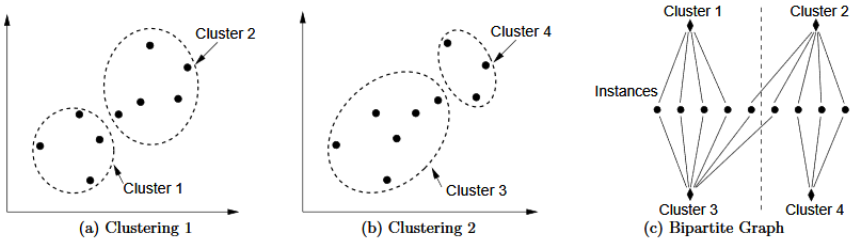


FIGURE 2.20 – Exemple de recherche de consensus entre deux clusterings (a) et (b) via le partitionnement d'un graphe bipartite (c) [FB04]. La ligne en pointillés représente la partition du graphe bipartite correspondant au consensus. Les instances, représentées par les points, sont dont séparées en deux groupes *consensus* par cette ligne.

pour le cas de la classification ou le calcul de la moyenne dans le cas de la régression constituent deux approches relativement simples, des méthodes d'agrégation plus complexes sont nécessaires dans le cas du clustering. En effet, il faut une approche de consensus permettant de prendre en compte les résultats de différents modèles de clustering, pouvant donner des résultats totalement différents, notamment en termes de nombre de clusters.

La figure 2.19 présente l'approche générique des méthodes d'ensemble dans le cadre non supervisé, où m clusterings différents sont agrégés en un seul clustering consensus C*, par le biais d'une méthode de consensus.

Deux approches de consensus principales sont retrouvées dans la littérature [AW19] [VPRS11] : l'approche par co-occurence et l'approche de partition médiane. Dans la première, le consensus est obtenu en analysant les co-occurrences des objets dans les clusters obtenus, plus précisément en comptant nombre de fois où un objet est affecté à un cluster donné, ou le nombre de fois où deux objets appartiennent au même cluster. Parmi les approches de ce type, on retrouve les méthodes de réétiquetage et vote (*relabelling and voting*) [AK10], ou encore les approches fondées sur l'analyse de graphes [FB04]. L'approche proposée dans [FB04] est très intéressante, et consiste à ramener le problème de recherche de consensus à un problème de partitionnement de graphe bipartite. La figure 2.20 présente le fonctionnement de cette approche.

L'approche de recherche de partition médiane quant à elle considère le problème de recherche de consensus comme un problème d'optimisation. Formellement, l'équation 2.31 présente la méthode générale afin d'obtenir le consensus $C*$. Elle consiste à chercher le clustering maximisant la similarité avec l'ensemble des partitions obtenues. Cette similarité correspond à la fonction Γ de l'équation 2.31. De nombreuses propositions ont été faites pour le calcul de cette similarité, par emprunt à de nombreux autres domaines de recherche : mesure de la NMI (que nous verrons dans la section suivante, 2.3.7), calcul du coefficient de Jaccard, fonction noyaux, etc.

$$C^* = argmax_{C \in \mathcal{C}} \sum_{j=1}^{m} \Gamma(C, C_j) \tag{2.31}$$

Nous voyons ici quelque chose d'intéressant, que nous avons déjà entrevu lors de la présentation des méthodes d'ensemble dans le cadre supervisé : de nombreux algorithmes et mesures que nous avons décrits jusqu'à présent peuvent être considérés comme des briques élémentaires d'algorithmes plus complexes, que ce soit dans le cadre supervisé ou non supervisé. Certains concepts applicables dans le cadre supervisé peuvent être transposés dans le cadre non supervisé, décloisonnant une limite stricte entre algorithmes. De la même manière, certaines méthodes peuvent être utilisées à des fins bien différentes de leur utilisation initiale, thème que nous allons revoir dans la suite de ce document.

2.3.7 Évaluation d'un clustering

Comme dans toute tâche d'apprentissage, il est nécessaire d'effectuer une évaluation des performances de la méthode de clustering utilisée. Cette évaluation est essentielle, dans la mesure où elle permet d'avoir une meilleure compréhension de la méthode, ainsi que d'avoir confiance en ses résultats avant déploiement. Elle permet par ailleurs de comparer les résultats de plusieurs algorithmes de clustering.

Dans le cas du clustering, on retrouve deux catégories de métriques d'évaluation : (i) les métriques d'évaluation interne, se basant sur des caractéristiques intrinsèques des données, et (ii) les métriques d'évaluation externes, se basant sur les classes d'appartenance de chaque instance. Ce dernier type de métrique ne peut donc être utilisé que dans le cas -très- particulier de données dont la vérité terrain est connue. En pratique, ce type de métrique semble plutôt adaptée au développement et à l'évaluation de nouvelles méthodes plutôt qu'à une évaluation de résultats sur une problématique de terrain.

Dans cette section, nous allons présenter quelques-unes de ces métriques.

2.3.7.1 Métriques d'évaluation internes

Les étiquettes des instances n'étant pas connues/utilisées, ces métriques se basent sur les informations disponibles. Très souvent, il s'agit de la structure des clusters et les caractéristiques des données.

Au vu des très nombreuses métriques existantes dans la littérature, cette section n'a pas pour objectif d'être exhaustive. Il s'agit plutôt ici de présenter succinctement quelques métriques couramment rencontrées, et les idées sur lesquelles elles se basent.

Dans la description des métriques suivantes, on notera $i \in C_i$ l'instance i, appartenant au cluster i, et $d()$ une fonction de distance, telle que la distance euclidienne.

Silhouette [Rou87] La mesure de silhouette peut être interprétée comme une quantification de la similarité d'une instance vis-à-vis de leur propre cluster d'appartenance par rapport à leur similarité vis-à-vis des autres clusters.

Son calcul dépend de celui de deux éléments :

1. La distance moyenne entre l'instance considérée i et toutes les autres instances de son cluster d'appartenance, que l'on notera $d_{intra}(i)$.
2. La plus petite distance moyenne entre i et toutes les instances d'un autre cluster, que l'on notera $d_{inter}(i)$.

Les équations 2.32 et 2.33 décrivent formellement comment obtenir ces deux valeurs.

$$d_{intra}(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j) \tag{2.32}$$

$$d_{inter}(i) = min_{k \neq i} \frac{1}{C_k} \sum_{j \in C_k} d(i, j) \tag{2.33}$$

À partir de ces deux éléments, il est possible d'obtenir la mesure de silhouette $silh(i)$ d'une instance i par la relation :

$$silh(i) = \frac{d_{inter} - d_{intra}}{max(d_{inter}, d_{intra})} \tag{2.34}$$

Si $|C_i| = 1$, on définit $silh(i) = 0$. La silhouette a donc toujours une valeur entre -1 et 1. Une valeur de 1 correspond à un bon clustering. En effet, si $d_{intra} << d_{inter}$, cela signifie que l'instance considérée est bien plus proche des instances de son cluster d'appartenance que des instances des autres clusters. À l'inverse, une valeur proche de -1 correspond à une attribution erronée.

La mesure plus fréquemment retrouvée dans la littérature est cependant une variante, le *coefficient de silhouette* [KR09]. Soient k le nombre de clusters obtenus par la méthode de clustering, et $\bar{silh}(k)$ la valeur de silhouette moyenne pour toutes les instances du jeu de données. Le coefficient silhouette est obtenu par la relation présentée équation 2.35.

$$cs = max_k \bar{silh}(k) \tag{2.35}$$

Cette approche peut servir d'heuristique dans la détermination du nombre de clusters optimal. En effet, on peut supposer que lorsque la valeur moyenne de silhouette est maximale, le clustering est meilleur. Elle est souvent retrouvée dans la littérature [HDMB21, HKK05].

Indice de Davies-Bouldin [DB79] Comme le coefficient de silhouette, l'indice de Davies-Bouldin peut être utilisé afin d'évaluer la séparation entre clusters. Soit $d_{intra}(i)$ la distance moyenne entre une instance i et toutes les autres instances du même cluster vue précédemment. L'indice de Davies-Bouldin, noté DB est calculé selon la relation présentée équation 2.36, où $R_{ij} = \frac{\bar{d_{intra}}(i) + \bar{d_{intra}}(j)}{d(i,j)}$.

$$DB = \frac{1}{k} \sum_{i=1}^{k} max_{i \neq j} R_{ij} \tag{2.36}$$

L'un des inconvénients de cette mesure est que sa valeur dépend fortement du type de clusters. En effet, dans le cadre de clusters convexes tels que ceux que l'on peut retrouver avec la méthode de k-moyennes, l'indice de Davies-Bouldin a tendance à donner des valeurs plus élevées que dans d'autres cadres.

2.3.7.2 Métriques d'évaluation externes

Les mesures d'évaluation externes permettent quant à elles d'évaluer un clustering à partir de leurs véritables groupes d'appartenance. Leur utilisation présuppose donc la connaissance de la *vérité terrain*, et est donc limitée dans la pratique.

Deux types de méthodes sont couramment retrouvées dans la littérature, notamment dans le cadre de l'évaluation de méthodes de clustering : la mesure de l'information mutuelle (*mutual information*, MI) et la mesure de l'indice de Rand (*Rand index*). On retrouve souvent leurs versions normalisées (*Normalized Mutual Information*, NMI) ou ajustées (*Adjusted Rand Index*, ARI).

Normalized Mutual Information (NMI) Soit $C = \{c_1, c_2, \ldots, c_n\}$ l'ensemble des clusters assignés à n objets par une méthode de clustering, et $R = \{r_1, r_2, \ldots, r_n\}$ l'ensemble des véritables classes de ces objets.

Nous avons défini précédemment la notion d'entropie (équation 2.12), notée H. L'information mutuelle entre C et R, notée $MI(C, R)$, est définie par l'équation 2.37.

$$MI(C, R) = H(C) - H(C|R). \tag{2.37}$$

On notera ici que cette relation est très proche de celle du gain d'information, que l'on peut retrouver dans l'équation 2.15. De nombreuses normalisations de l'information mutuelle existent dans la littérature. Une forme couramment retrouvée est la normalisation par la moyenne des entropies sur les classes d'appartenance des objets et sur les classes attribuées par le clustering.

L'équation (2.38) présente le calcul de cette version normalisée. Il s'agit de celle utilisée notamment dans *Scikit-learn*.

$$NMI(C, R) = \frac{2 \times MI(C, R)}{H(C) + H(R)} \tag{2.38}$$

Une revue des méthodes basées sur l'information est réalisée par Vinh *et al.* dans [VEB10]. La table 2.2 présente quelques-unes d'entre elles.

On retrouve parfois l'information mutuelle ajustée (*Adjusted Mutual Information*, AMI), qui est une normalisation ajustée pour prendre en compte le hasard. Pour ces versions ajustées, un clustering aléatoire doit obtenir un score de 0. L'expression d'une mesure ajustée a une forme générale [HA85], $IndiceAjuste = \frac{Indice - IndiceAttendu}{MaxIndice - IndiceAttendu}$.

Au dénominateur, d'autres fonctions que max peuvent être utilisées. L'équation 2.39 correspond au calcul de l'AMI, où $E(MI)$ représente la valeur attendue pour la MI. On constate que dans ce cas, il s'agit de la moyenne arithmétique.

$$AMI(C, R) = \frac{MI(C, R) - E(MI(C, R))}{\frac{H(C)+H(R)}{2} - E(MI(C, R))} \tag{2.39}$$

Nom	Expression	Intervalle
Information mutuelle [BDGS05]	$MI(C,R)$	$[0, min(H(C), H(R))]$
Information mutuelle normalisée		
NMI_{joint} [Yao03]	$\frac{MI(C,R)}{H(C,R)}$	$[0,1]$
NMI_{max} [Kva87]	$\frac{MI(C,R)}{max(H(C),H(R))}$	$[0,1]$
NMI_{sum} [Kva87]	$\frac{2\times MI(C,R)}{H(C)+H(R)}$	$[0,1]$
NMI_{sqrt} [SG02]	$\frac{MI(C,R)}{\sqrt{H(C),H(R)}}$	$[0,1]$
NMI_{min} [Kva87] [LGT08]	$\frac{MI(C,R)}{min(H(C),H(R)}$)	$[0,1]$

TABLE 2.2 – Résumé des méthodes basées sur l'information [VEB10].

	Y_1	$\cdots$	Y_q	$sums$
X_1	n_{11}	$\cdots$	n_{1q}	a_1
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	$\cdots$	n_{qr}	a_r
$sums$	b_1	$\cdots$	b_q	

TABLE 2.3 – Tableau de contingence de deux clusterings

Ces formes ajustées sont particulièrement adaptées dans le cas où le nombre d'objets est faible en comparaison au nombre de clusters – données de transcriptomique par exemple –. En effet, les valeurs de NMI non ajustées auront tendance à tendre vers 1 lorsque le nombre de clusters augmente.

Adjusted Rand Index (ARI) L'ARI [Ran71, HA85] est une mesure de qualité permettant de quantifier l'accord entre deux partitions d'un jeu de données. Il s'agit là encore d'une mesure ajustée.

La table 2.3 représente le tableau de contingence entre deux clusterings distincts, où chaque valeur n_{ij} correspond au nombre d'instances associées au cluster i de Y et au cluster j de X. L'ARI est calculée à partir de cette table grâce à l'équation 2.40.

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}} \tag{2.40}$$

Une valeur d'ARI de 1 indique un accord parfait – aux permutations près –, alors qu'une valeur de 0 indique une partition des objets qui n'est pas meilleure que le hasard.

2.4 Bilan du chapitre

Nous avons vu dans ce chapitre que le champ d'application des méthodes d'apprentissage automatique est très vaste. Bien que les algorithmes de fouilles soient très nombreux, nous avons fait le choix de les présenter de manière très simplifiée, en opposant l'apprentissage supervisé au non supervisé. Bien sûr, comme nous l'avons précisé par ailleurs, ce choix peut être très réducteur. Nous l'avons cependant fait dans un objectif de concision,afin de garder un cadre serré sur la problématique de ce document.

Pour ce qui est des méthodes supervisées, différentes approches permettent d'arriver à la même fin, à savoir déterminer la classe (ou étiquette) d'un objet précédemment inconnu, après une phase d'apprentissage sur des objets dont la classe est connue. Chaque algorithme a ses forces et ses faiblesses, dans la mesure où aucun algorithme ne peut donner les meilleurs résultats dans tous les cas.

Dans le cas des méthodes non supervisées, l'objectif est différent : il s'agit d'extraire des éléments à partir de données souvent imparfaites, en particulier, non préalablement étiquetées par des experts. Dans le cas du clustering, l'objectif est de regrouper les objets en fonction de leur (dis)similarité. Cette notion de similarité, ou de dissimilarité, est donc essentielle pour cette tâche. Il est essentiel de choisir *la bonne* mesure, adaptée aux données et permettant de quantifier efficacement la ressemblance ou l'éloignement entre deux objets. En effet, toutes les mesures ne se valent pas : certaines seront particulièrement adaptées à des données hétérogènes, d'autres seront sensibles à la dimension des données, ou encore aux différences d'échelle entre variables. La définition ou la sélection d'une bonne mesure de (dis)similarité est donc une tâche complexe, mais essentielle du processus de clustering. Ceci explique pourquoi nous avons choisi de nous concentrer sur cette notion au cours de cette thèse.

Notons que les frontières entre méthodes supervisées et non supervisées sont poreuses puisque certains concepts se retrouvent dans un cadre comme dans l'autre. C'est par exemple le cas des arbres de décision, qui représentent d'un côté une méthode d'apprentissage supervisé, mais que l'on retrouve par ailleurs dans les méthodes de détection de points aberrants. C'est aussi le cas des méthodes d'ensemble, pertinentes dans les deux cadres. Il est donc intéressant de ne pas se cloisonner dans l'une ou l'autre tâche, et de rechercher les concepts développés dans l'une qui pourraient être pertinents et utiles dans l'autre. C'est la direction que nous avons explorée dans cette thèse.

3

Analyse de graphes

Sommaire

Jusqu'à présent, nous avons décrit les méthodes de fouille de données dont le cadre d'application se limite aux données décrites sous formes tabulaires, où chaque attribut est encodé par une valeur d'un type spécifique pour chaque instance. Il ne s'agit cependant que d'une partie du champ des possibles. En effet, d'autres modes de représentation des données existent, permettant d'encoder et de représenter des informations particulières, telles que les relations entre objets. C'est le cas notamment des graphes, que nous allons décrire dans ce chapitre et sur lesquels nous allons nous concentrer. Nous verrons qu'il s'agit d'un modèle de représentation particulièrement intéressant, mais dont l'analyse peut s'avérer complexe.

L'intérêt de représenter les objets avec leurs relations est particulièrement marqué dans certains domaines d'applications tels que l'étude des réseaux sociaux ou des réseaux d'ordinateurs et l'analyse du web. Cependant, se limiter à ce type de domaines serait réducteur puisque les graphes peuvent être retrouvés et utilisés dans des domaines aussi variés que la biologie structurale (où les graphes représentent des réseaux de protéines pouvant former des complexes capables d'accomplir des fonctions), la chimie (où chaque molécule peut être représentée par un ensemble d'atomes reliés par différents types de liaisons chimiques), les villes intelligentes, etc.

Enfin, il est possible de fusionner plusieurs graphes, ou d'intégrer plusieurs sources de données sous forme d'un graphe afin de pouvoir tirer parti de l'information contenue dans plusieurs sources de données. C'est notamment ce qui a été fait dans le cadre du projet FIGHT-HF, où une base de connaissance biomédicale, la *FHF Knowledge Box*, a été développée.

Cette base de connaissance constitue une ressource clé afin de mettre en évidence des relations complexes et intéressantes – préalablement inconnues – entre pathologies, protéines, métabolites, etc. C'est donc tout naturellement que nous nous sommes penchés sur cette tâche d'analyse de graphe, dont l'exploitation des résultats, bien que complexe, peut mener à des hypothèses intéressantes à explorer sur un plan biomédical.

Après avoir donné quelques définitions dans la section 3.1, nous discuterons des tâches de recherche de clusters et leur évaluation, tout d'abord dans les graphes *simples* (section 3.2), puis dans les graphes attribués (section 3.3).

3.1 Introduction et définitions

Les graphes sont des modèles de données particulièrement intéressantes, permettant de représenter des relations entre objets.

Formellement, un graphe G est une structure définie comme une paire (V, E), où V est un ensemble de noeuds $\{v_1, v_2, \ldots, v_n\}$, et E un ensemble d'arêtes entre ces noeuds. Ces arêtes peuvent être dirigées d'un noeud vers un autre, auquel cas on parle d'un graphe orienté. De plus, les noeuds, comme les arêtes, peuvent être munis d'un ou plusieurs attributs. Le cas le plus courant est le cas du graphe pondéré, qui est un cas particulier de graphe attribué où une valeur numérique est associée à chaque arête. Formellement, une fonction de poids $w : E \to \mathbb{R}$ assigne une valeur à chaque arête. Cette valeur peut par exemple quantifier la force d'une relation entre deux noeuds.

Il existe différentes structures de données permettant de définir un graphe. La matrice d'adjacence en est une, couramment retrouvée dans les bibliothèques logicielles spécialisées dans le traitement de graphes. Soit $n = |V|$ le nombre de noeuds dans un graphe G[13]. La matrice d'adjacence A de ce graphe est une matrice de taille $n \times n$ où :

$$a_{i,j} = \begin{cases} 1 & \text{si} \quad (i,j) \in E \\ 0 & \text{sinon} \end{cases} \tag{3.1}$$

Dans le cas d'un graphe non orienté, cette matrice est symétrique, dans la mesure où la relation entre deux noeuds existe quel que soit le noeud de départ. Dans le cadre de ce document, nous ne considérerons que des graphes non orientés. Ainsi, les définitions de cette section concerneront ce type de graphe, bien qu'elles peuvent souvent être définies dans le cadre orienté.

Un graphe peut aussi être représenté sous la forme d'une liste d'adjacence. Cette représentation consiste à associer à chaque noeud la liste de ces voisins, *i.e* pour un noeud $u \in V$, l'ensemble des noeuds $v \in V$ pour lesquels $(u, v) \in E$. Notons que l'on retrouve parfois la notation $\Gamma(u)$, correspondant à l'ensemble des noeuds dans le voisinage de u. Cette notion de voisinage peut

13. Cette quantité est parfois retrouvée sous le terme *ordre* du graphe.

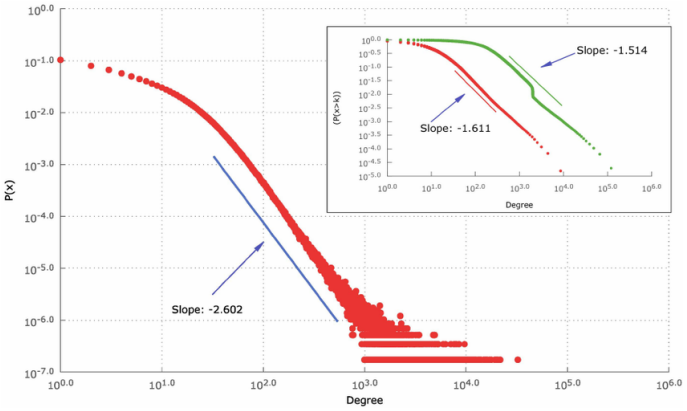


FIGURE 3.1 – Exemple de distribution de degrés [SKD+14]. On remarque ici une pratique courante, qui est de représenter cette distribution en utilisant une échelle logarithmique.

aussi être définie dans le cadre des graphes orientés, où l'on peut parler de prédécesseur et de successeurs : pour un noeud prédécesseur u, on note l'ensemble des successeurs, à savoir les noeuds connectés par une arête dirigée depuis u.

Les graphes sont des structures complexes dont on peut dériver de nombreuses propriétés.

Ce document n'a pas l'ambition d'effectuer une revue exhaustive de ces propriétés. En revanche, il est cependant important d'en définir quelques-unes.

Degré Le degré d'un noeud donné u, noté $deg(u)$, correspond au nombre d'arêtes incidentes à ce noeud. Un graphe est dit régulier si tous les noeuds de ce graphe ont le même degré.

Degrés minimal et maximal Le degré minimal noté $\delta(G)$ correspond au degré le plus faible de l'ensemble des noeuds du graphe , soit $min(deg(u) : u \in V)$. Le degré maximal $\Delta(G)$ quant à lui correspond au noeud dont le degré est le plus grand.

Séquence de degrés On appelle séquence de degrés la séquence $deg(u_1) \geq deg(u_2) \geq \cdots \geq deg(u_n)$. Fait intéressant, la somme de ces degrés est toujours un nombre pair. Il s'agit de ce qui est dénommé communément le lemme des poignées de mains.

Distribution de degrés La distribution des degrés représente pour chaque degré k, la fraction de noeuds du graphe ayant ce degré. Un exemple est présenté figure 3.1. On remarque dans ce cas précis qu'en utilisant une échelle logarithmique, la relation entre degrés et fraction de noeuds semble linéaire. Il s'agit ici d'un réseau invariant d'échelle (*scale-free network*), où la fraction suit la loi $P(k) \sim k^{-\gamma}$, où le coefficient γ est strictement positif. En d'autres termes, ce type de graphes possède une distribution de degrés qui suit une loi de puissance. Cela signifie que certains noeuds des graphes ont un degré bien plus grand que le degré moyen du graphe. Ces noeuds sont appelés *hubs*. Les réseaux invariants d'échelle sont très présents dans le monde réel, et on les retrouve tant dans les représentations du web que dans des réseaux biologiques [BA99]. Dans ces deux exemples, les *hubs* correspondent à des éléments centraux dans les relations entre objets, tels que des sites avec de nombreux liens, ou encore des protéines essentielles au bon fonctionnement d'un organisme [VCL+09].

Densité La densité d'un graphe correspond au nombre d'arêtes présentes dans le graphe, comparé au nombre d'arêtes maximal possible. On a donc :

$$D(G) = \frac{|E|}{\binom{n}{2}} \tag{3.2}$$

Lorsque de nombreuses arêtes existent dans le graphe, c'est-à-dire quand la densité est proche de 1, on parle de graphe dense. Dans le cas contraire, on parle de graphe creux.

Chemin entre deux noeuds u et v Un chemin entre un noeud u et v dans un graphe G correspond à une séquence d'arêtes dans E $(u, u_1), (u_1, u_2), \ldots, (u_k, v)$, commençant eu noeud u et se terminant au noeud v. On nomme longueur d'un chemin le nombre d'arêtes présentes dans ce chemin. Bien sûr, dans de nombreux graphes, plusieurs chemins peuvent exister entre chaque paire de noeuds. Le chemin dont la longueur est la plus courte correspond au plus court chemin. Notons qu'un chemin peut commencer et débuter par le même noeud. Si de tels chemins existent, on part de cycle.

Centralité intermédiaire La centralité intermédiaire, ou *betweenness centrality*, est une mesure basée sur les chemins les plus courts [Fre77]. Sa valeur pour un noeud v est obtenue par l'équation 3.3, où $\sigma_{st}(v)$ correspond au nombre de plus courts chemins entre le noeud s et le noeud t passant par v, et σ_{st} au nombre de plus courts chemins totaux entre s et t. Ici, les trois noeuds s, t et v sont toujours différents. Une centralité élevée peut correspondre par exemple à un *hub*, que nous avons défini précédemment. En effet, un *hub* étant un noeud du graphe ayant un fort degré, de nombreux chemins y passent donc. Ainsi, la probabilité que de nombreux plus courts chemins passent par ce *hub* est plus élevée que pour d'autres noeuds du graphe. Cette notion permet d'identifier les noeuds critiques d'un graphe, dans la mesure où les noeuds ayant une grande centralité intermédiaire correspondent aux noeuds dont la suppression a le plus grand impact sur les connexions entre les autres noeuds. La suppression de ces noeuds peut permettre par ailleurs de mettre en évidence des ensembles de noeuds déconnectés, pouvant avoir un intérêt dans les tâches de clustering.

$$b(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{3.3}$$

Enfin, de nombreux graphes particuliers – ou plus précisément, types de graphes – ont été définis au fur et à mesure des évolutions de ce domaine. Ces graphes particuliers sont remarquables de par certaines propriétés spécifiques, ou encore de par leur utilité pratique. On retrouve par exemple la notion de clique, correspondant à un sous-ensemble de noeuds tous connectés entre eux. Cette notion est par exemple particulièrement intéressante dans la découverte de groupes homogènes, par exemple dans un graphe modélisant des relations sociales [LP49]. Cette tâche est par ailleurs l'une des tâches majeures de l'analyse de graphe.

L'analyse, ou fouille de graphes est similaire à ce que l'on a pu voir précédemment, en ce sens que les mêmes types d'approches peuvent être appliqués. Une ambiguïté est cependant présente dans le cas des graphes. En effet, la question de la définition des objets se pose : s'agit-il des noeuds, ou des graphes ? Les deux réponses sont possibles – en fonction de la tâche – changeant drastiquement les algorithmes applicables.

Dans le cas du clustering, ce dernier peut donc être compris comme (i) regrouper des noeuds de manière homogène ou (ii) regrouper des graphes similaires entre eux, c'est-à-dire partageant les mêmes propriétés ou reliant les noeuds entre eux de manière similaire.

Les termes *between-graph* clustering et *within-graph* clustering peuvent être retrouvés [Kin19], et ont l'avantage de rompre cette ambiguïté. Le terme *recherche de communautés* est aussi retrouvé pour se référer au *within-graph* clustering. Ce dernier est par ailleurs très *connoté* analyse de réseaux sociaux, et peut paraître réducteur.

Intuitivement, l'objectif d'une méthode de clustering dans des graphes non attribués est de mettre en évidence des groupes de noeuds reliés entre eux par de nombreuses arêtes. De plus, les noeuds appartenant à différents clusters devraient posséder un nombre de connexions réduites entre eux : il doit y avoir un nombre réduit d'arêtes externes. Dit autrement, un sous-ensemble de noeuds forme un bon cluster si le sous-graphe induit par ces noeuds est dense, et que ces derniers sont relativement peu connectés aux autres noeuds [Sch07]. Ainsi, une approche stricte de cette vision revient à considérer comme clusters des ensembles de noeuds tous connectés entre eux : il s'agit de cliques. Cela ne correspond que très peu à des graphes que l'on retrouve dans la pratique [MGM16]. Il est donc nécessaire de mettre en œuvre des algorithmes permettent d'extraire des groupes homogènes sans pour autant qu'ils constituent des cliques.

Cette définition de la notion de clusters n'est cependant pas la seule. Là où celle que nous avons décrite correspond à considérer comme clusters les ensembles de noeuds avec une forte densité, une autre possibilité est de considérer comme similaires les noeuds ayant des motifs communs de connexion à d'autres noeuds. Cette vision correspond à la notion d'équivalence structurelle entre noeuds [LW71], plus particulièrement d'équivalence régulière (*regular equivalence*) [EB94]. En pratique, ces *motifs similaires* correspondent au fait que deux objets sont considérés comme équivalents s'ils sont reliés aux mêmes objets. Enfin, on peut voir les noeuds appartenant à une même communauté comme des noeuds ayant des probabilités similaires d'être reliés avec les autres noeuds par des arêtes. Il s'agit de la notion d'équivalence stochastique [HLL83].

La recherche de groupes dans les graphes est un sujet très actif, de par l'intérêt pratique et de possibles informations cruciales qui peuvent en ressortir : informations concernant des réseaux d'interactions protéines-protéines [CY06], segmentation de clients en fonction d'intérêts communs [RKSR02], etc. Ces exemples, bien que quelque peu vus et revus, permettent de mettre en exergue certaines difficultés liées au *meta*-clustering. Tout d'abord, dans la cadre des réseaux d'interactions biologiques, on peut se poser la question du type de noeuds. Dans le cadre du réseau de protéines, la question ne se pose pas, dans la mesure où l'ensemble des objets sont des protéines. Cependant, dans le cadre de la Edge-Box que nous avons mentionnée précédemment, nous sommes dans le cadre d'un graphe où l'ensemble des objets n'est pas du même type. On retrouve alors des protéines, liées à des maladies, elles-mêmes pouvant être liées à des médicaments, etc. [14]. Dans ce cas, la définition en amont de la tâche de clustering est critique : l'objectif est-il de trouver des clusters d'objets du même type, ou non ? Par ailleurs, une fois lesdits clusters obtenus, la question de leur interprétation est cruciale.

Les tâches d'analyses de graphes ne se limitent bien sûr pas à ces seules tâches de clustering. Outre ces tâches, dont la littérature est déjà bien fournie, on retrouve des approches spécifiques

14. On parle dans ce cas de graphes hétérogènes.

aux graphes, telles que la prédiction de liens. [MBC16] constitue une revue relativement récente de ces approches, qui bien qu'elles soient intéressantes dans de nombreuses applications (découverte de relations, recommandation, etc.), sortent du cadre de ce travail.

Dans ce chapitre – et plus généralement dans l'ensemble de ce document – nous traiterons uniquement le cas du clustering, et n'aborderons donc pas les approches de classification supervisée. Nous commencerons par traiter dans la section 3.2 du clustering de graphes simples, c'est-à-dire dans le cas de structures non augmentées d'attributs. Par la suite, nous développerons dans la section 3.3 les approches adaptées à des graphes attribués.

3.2 Clustering dans les graphes simples

Comme pour le clustering de vecteurs d'attributs que nous avons décrits dans le chapitre 2, il existe différentes approches fondamentalement différentes pour trouver des communautés dans les graphes. Nous présentons ici quelques exemples intéressants permettant de mettre en valeur certaines de ces approches. Là encore, il ne s'agit pas d'effectuer une liste exhaustive, mais plutôt un tour d'horizon rapide de quelques méthodes ou approches phares.

3.2.1 Clustering à base d'optimisation de mesure : Louvain

L'une des grandes catégories d'approches regroupe celles basées sur l'optimisation d'une mesure. L'algorithme **Louvain** [BGLL08] en est un représentant, dont le fonctionnement repose sur l'optimisation de la modularité. La modularité [New06] est une mesure permettant de quantifier la séparation d'un graphe en différentes communautés, et correspond à la proportion d'arêtes entre noeuds d'un même groupe, relativement à un graphe où les arêtes seraient présentes de manière aléatoire entre ces mêmes noeuds. Une modularité élevée correspond donc à un cas où l'on retrouve plus d'arêtes au sein des clusters que le nombre d'arêtes attendu si ces dernières étaient ajoutées aléatoirement.

Formellement, notons Q la modularité d'une partition donnée d'un graphe G. L'équation 3.4 présente son calcul, où A correspond à la matrice d'adjacence de G, $deg(i)$ et $deg(j)$ aux degrés des noeuds i et j, m au nombre d'arêtes de G, c_i et c_j aux communautés des noeuds i et j, et δ à la fonction delta de Kronecker [15]

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta(c_i, c_j) \tag{3.4}$$

Dans la méthode de Louvain, l'optimisation de la modularité est effectuée de manière itérative. Il s'agit d'une approche gloutonne, où dans un premier temps, chaque noeud du graphe constitue sa propre communauté.

La modification de modularité résultant du passage de chaque noeud dans les communautés de ses voisins, de manière séquentielle, est calculée. Les déplacements entraînant la plus forte augmentation de modularité sont gardés. Ce processus est répété pour l'ensemble des noeuds, jusqu'à ce qu'aucune variation forte de modularité ne soit observée. Notons que cette phase d'optimisation de modularité n'est qu'une phase de la méthode de Louvain. En effet, cette dernière comporte également une phase d'agrégation de communautés, considérant chaque cluster

15. $\delta(i, j) = 1$ si $i = j$, 0 dans les autres cas.

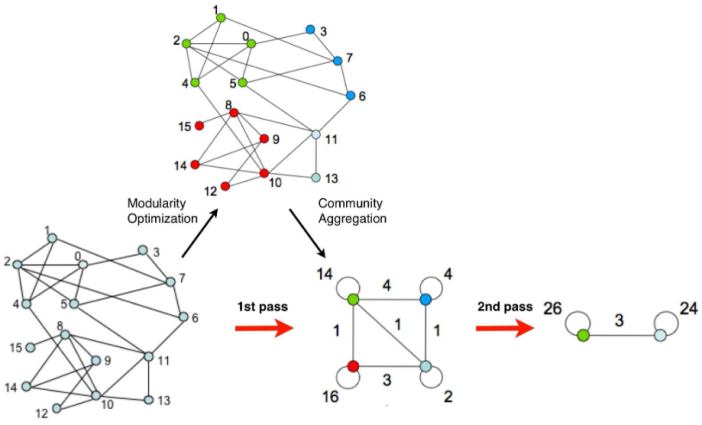


FIGURE 3.2 – Fonctionnement global de la méthode Louvain [BGLL08]. Il s'agit d'une méthode itérative – on voit ici deux *passes* –, où chaque itération est constituée de deux étapes : une première étape d'optimisation de la modularité, et une seconde étape d'agrégation des communautés. Les itérations sont stoppées lorsqu'il n'est plus possible d'augmenter la modularité.

retrouvé par la phase d'optimisation de la modularité comme un seul noeud du graphe. Elle permet donc par ailleurs l'obtention d'une hiérarchie de communautés. La figure 3.2 résume le fonctionnement de l'approche.

Cette approche est intéressante, tant sur le plan conceptuel qu'en termes de complexité – O(nlog(n)) –, mais n'est qu'une représentante de la famille des méthodes d'optimisation de mesure [TWvE19] [CNM04] [WT07]. Bien que donnant de meilleurs résultats que d'autres approches gloutonnes similaires, cette approche présente cependant quelques limites. En particulier, les résultats dépendent fortement de l'ordre des noeuds dans l'étape initiale de fusion de clusters. En effet, l'approche n'est pas déterministe dans la mesure où des séquences de noeuds différentes pourront ne pas mener aux mêmes clusterings.

Les résultats obtenus par la méthode sont par ailleurs à nuancer. En effet, celle-ci se basant sur l'optimisation d'une mesure en particulier – la modularité – les clusters obtenus ne sont pas toujours de bonne qualité [For10]. Cela met par ailleurs de nouveau en exergue la question de l'évaluation d'un clustering : un même clustering peut sembler être de bonne qualité selon un certain critère, et mauvais selon un autre.

Notons enfin que l'approche que nous avons choisi de développer ici n'est bien sûr pas la seule basée sur l'idée de l'optimisation d'une mesure. L'algorithme de Girvan-Newman [NG04] repose par exemple sur l'idée que les arêtes à couper prioritairement sont celles par lesquelles le plus grand nombre de plus courts chemins passent. Il s'agit là d'une vision diamétralement opposée à la méthode de Louvain, dans la mesure où il s'agit ici d'une approche divisive plutôt qu'agglomérative, et basée ici sur la notion de centralité plutôt que de modularité. Sa complexité est par ailleurs élevée : $\mathcal{O}(N^3)$.

3.2.2 Clustering à base de marche aléatoire : MCL

Une autre catégorie de méthodes de recherche de communautés regroupe celles basées sur le parcours aléatoire (*random walk*) du graphe. La version la plus simple de parcours aléatoire est la suivante. Étant donné un noeud de départ v_0, un noeud $v_1 \in \Gamma(v_0)$ est sélectionné aléatoirement. Cette procédure est répétée sur le nouveau noeud, jusqu'à ce qu'une certaine condition d'arrêt soit vérifiée. Une séquence de noeuds $v_0, v_1, \ldots, v_k$ est ainsi obtenue. La probabilité de transition d'un noeud v_i à v_j est couramment obtenue par la relation présentée équation 3.5. Rappelons par ailleurs que dans l'ensemble de cette thèse, nous ne considérons que les cas de graphes non dirigés. Ainsi, si un *marcheur* peut aller d'un noeud u à un noeud v, le parcours inverse est aussi possible.

$$p_{v_i v_j} = \begin{cases} \frac{1}{d(v_i)} \text{ si } v_j \in \Gamma(v_i) \\ 0 \text{ sinon} \end{cases} \tag{3.5}$$

On peut noter que ces transitions ne sont pas dépendantes des transitions antérieures, mais uniquement du noeud actuel. En d'autres termes, la séquence de noeuds visités est une chaîne de Markov.

L'une des idées sous-jacentes des approches basées sur un parcours aléatoire est qu'en visitant un cluster dense lors d'un parcours, on aura peu de chances de quitter ce dernier avant que de nombreux noeuds ne soient visités [VD00]. Un exemple connu est l'algorithme **MCL** (pour Markov Cluster Algorithm) [VD00]. L'idée principale de l'approche consiste à simuler un débit dans le graphe, et de l'augmenter dans les régions où ce débit est élevé, ou au contraire le réduire dans les zones où il est faible. Ainsi, dans les zones entre communautés, le courant va se réduire au fur et à mesure, mettant en évidence les groupes homogènes.

Une analogie intéressante consiste à considérer les noeuds d'un graphe comme les bâtiments d'une ville, et les arêtes comme les rues. Intuitivement, un piéton voyageant aléatoirement d'un bâtiment à l'autre aura tendance à parcourir plusieurs fois les rues d'un même quartier. Le débit de piétons aura dont tendance à être plus fort au sein des quartiers, et plus faible sur les rues ou ponts les reliant [16].

En pratique, l'algorithme fonctionne de la manière suivante. Dans un premier temps, la matrice d'adjacence correspondant au graphe non orienté, ainsi que deux paramètres e et r sont fournis. La matrice est ensuite normalisée par colonne. Cette normalisation est une manière très intéressante d'obtenir la matrice de transition dans un graphe non orienté. La puissance de e de cette matrice est ensuite calculée, correspondant à l'étape d'expansion de l'algorithme : le débit s'étend de noeud en noeud. Ensuite, une étape dite *d'inflation* contrôlée par le paramètre r permet de renforcer le débit dans les zones de fort débit, et de le réduire dans les zones de faible débit. Ces deux dernières étapes sont répétées jusqu'à convergence.

Les auteurs précisent que bien qu'ils n'aient pas prouvé qu'il y a toujours convergence, la pratique montre que cette dernière est souvent atteinte. Fait intéressant par ailleurs, l'état de convergence correspond le plus souvent à une matrice particulière, où pour une colonne donnée, toutes les valeurs sont identiques, et ce pour l'ensemble de la matrice.

16. Il s'agit ici d'agents aléatoires. Dans la pratique, ce comportement ne serait pas forcément observé.

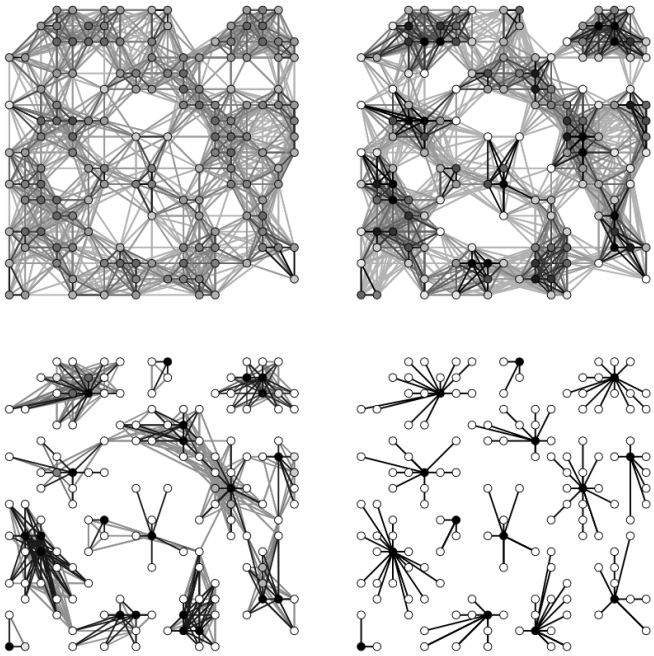


FIGURE 3.3 – 4 étapes de l'algorithme MCL [VD00]. On constate que le flux a tendance à être plus fort dans les zones fortement connectées, correspondant aux communautés.

La figure 3.3 présente la mise en évidence des communautés sous-jacentes d'un graphe itération après itération du processus décrit précédemment.

Cet algorithme, très intéressant sur le plan conceptuel, a une complexité initiale de $\mathcal{O}(n^3)$. L'auteur a par la suite proposé une approche permettant une réduction de cette complexité à $\mathcal{O}(nk)$, k étant ici le degré maximal pour un noeud.

3.2.3 Apprentissage de représentations

Enfin, l'apprentissage de représentations a connu un gain d'intérêt au cours de ces dernières années [ZYZZ18]. Leur objectif est d'obtenir une représentation fidèle des noeuds et de leurs relations dans un autre espace, *i.e*, réaliser un *plongement* (*embedding*). Une fois cette représentation obtenue, il peut être possible d'y appliquer des méthodes de clustering ne pouvant pas initialement être appliquées aux graphes.

Formellement, il s'agit d'apprendre une fonction $f : V \to \mathbb{R}^d$, avec $d \in \mathbb{N}$ qui préserve les informations relatives à la structure du graphe. En d'autres termes, les noeuds étant proches dans le graphe d'entrée doivent aussi l'être dans l'espace de sortie. La figure 3.4 présente un exemple de ce type de transformation sur un graphe.

DeepWalk [PARS14] et Node2Vec [GL16] font partie des méthodes les plus connues d'apprentissage de représentation. Quelle que soit la méthode, il est nécessaire de parcourir le graphe

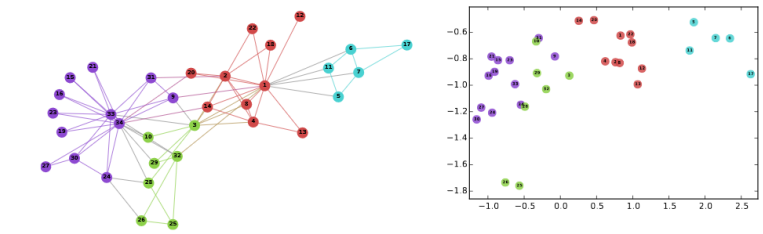


FIGURE 3.4 – Transformation des noeuds d'un graphe (à gauche) en points de $\mathbb{R}^2$ [PARS14].

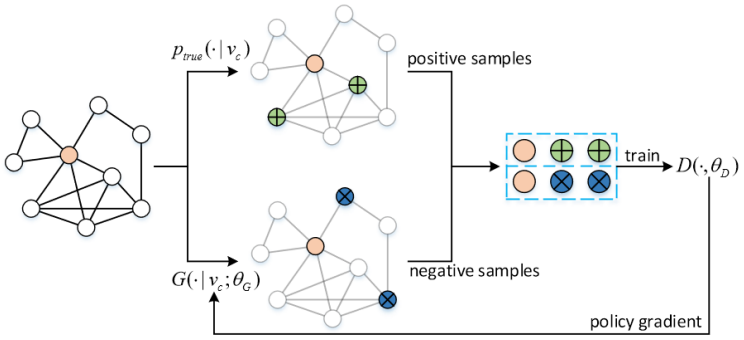


FIGURE 3.5 – Différentes étapes d'apprentissage dans la méthode CommunityGAN [JZZW19].

depuis de nombreux noeuds sans se cantonner aux voisins directs, afin de pouvoir capturer l'équivalence structurelle entre noeuds.

Dans DeepWalk, les séquences de noeuds obtenus par un parcours aléatoire sont considérées comme des séquences de mots, *i.e.* des phrases. Une méthode d'apprentissage de représentation dans le domaine de la fouille de texte, Skipgram [MSC+13] est par la suite appliquée à ces phrases.

Node2Vec est quant à elle une méthode très intéressante, basée sur un parcours aléatoire biaisé. Ce biais est modulable en fonction de deux paramètres, p, correspondant à la probabilité de retourner sur un noeud déjà visité, et q, la probabilité d'explorer des noeuds éloignés. Ces paramètres permettre de combiner deux types parcours : un parcours en largeur, permettant de quantifier l'homophilie, et un parcours en profondeur, permettant de quantifier l'équivalence structurale entre noeuds. CommunityGAN [JZZW19] est un exemple plus récent. La figure 3.5 présente le *framework* global de l'approche. La représentation apprise sous forme vectorielle est optimisée grâce à un réseau adverse génératif (*Generative Adversarial Net*, ou GAN).

Ces approches permettant d'obtenir des représentations concises des noeuds d'un graphe peuvent être utilisées dans différents contextes : prédictions de liens [GL16], annotation de protéines [SRA19], recommandations [ZXZ+18], et ce qui nous intéresse ici, dans la classification ou clustering de noeuds. Cependant, l'une des limitations est le côté *boîte noire* de l'approche, notamment dans la phase d'apprentissage en tant que telle. La représentation peut par ailleurs elle-même ne pas être explicite, et difficilement interprétable. De plus, nous avons vu ici que des exemples positifs sont nécessaires afin d'effectuer cet apprentissage, exemples qui ne sont

pas toujours disponibles, pour des raisons que nous avons déjà évoquées notamment en discutant de l'intérêt des approches non supervisées. Certaines approches, basées par exemple sur les autoencodeurs, peuvent permettre de dépasser cette limitation [KW16].

La représentation vectorielle obtenue par ces méthodes est par ailleurs intéressante dans la mesure où elle permet de simplifier le calcul d'une mesure de distance ou de similarité entre noeuds. En effet, des mesures courantes telles que la similarité cosinus peuvent être utilisées sans autre modification. Cette vision correspond à ce que nous cherchons à faire ici, à savoir mettre en œuvre une méthode permettant le calcul de similarité ou de distance entre noeuds d'un graphe.

3.2.4 Évaluation d'un clustering dans les graphes simples

Afin de pouvoir évaluer une méthode de clustering, plusieurs techniques sont disponibles. D'une part, pour des graphes extraits du monde réel (réseaux sociaux, réseaux informatiques, réseaux biologiques, etc.), il est possible d'annoter les noeuds par des humains, plus ou moins experts, afin d'obtenir une vérité terrain. Il est donc possible par la suite d'évaluer les performances d'un clustering en utilisant les métriques vues précédemment dans le cadre des données sous forme tabulaire, telles que la NMI ou l'ARI, présentés section 2.3.7.

Cependant, se baser uniquement sur les graphes réels pour évaluer une méthode présente certaines limites. Tout d'abord se pose la question de la validité des étiquettes. En effet, certains noeuds peuvent être étiquetés *par défaut*, sans appartenir clairement à un groupe particulier. De plus, cette tâche d'annotation est très coûteuse – en temps comme, malheureusement, en argent – et ne peut pas être effectuée sur de nombreux graphes. Cette tâche est potentiellement impossible sur de très grands réseaux, tels que des interactions entre utilisateurs des plateformes sociales les plus connues.

Il est donc intéressant d'avoir des méthodes permettant de générer des graphes présentant une structure de communauté, afin de pouvoir évaluer et comparer des méthodes de clustering. Fort heureusement, de nombreux auteurs se sont penchés sur la question, et ont proposé diverses approches afin d'effectuer cette tâche de génération de graphes.

Cavemen graphs Comme nous l'avons vu précédemment, une manière *naïve* de percevoir les clusters au sein des graphes est de considérer les cliques comme clusters. Bien que cette vision soit réductrice, les graphes *hommes des cavernes* sont basés sur cette vision et permettent une évaluation simple d'une méthode de clustering. Il s'agit ici de créer un graphe à partir de k cliques. Deux extensions sont couramment retrouvées : les *connected cavemen graphs* et les *relaxed cavemen graphs* [For10]. Les premiers sont obtenus en supprimant une arête de chacune de ces cliques et en la remplaçant par une arête reliant deux noeuds de deux cliques différentes, autour d'un cycle commun. De ce fait, on crée donc un seul graphe à partir de plusieurs cliques initiales [Wat04]. Un exemple de ce type de graphe est présenté figure 3.6

La version relâchée quant à elle introduit un nouveau paramètre p. Ce dernier représente la probabilité que chaque arête puisse être redirigée, et permet de générer des graphes plus réalistes. Ces derniers restent cependant très différents de ceux que l'on peut trouver, sur le terrain.

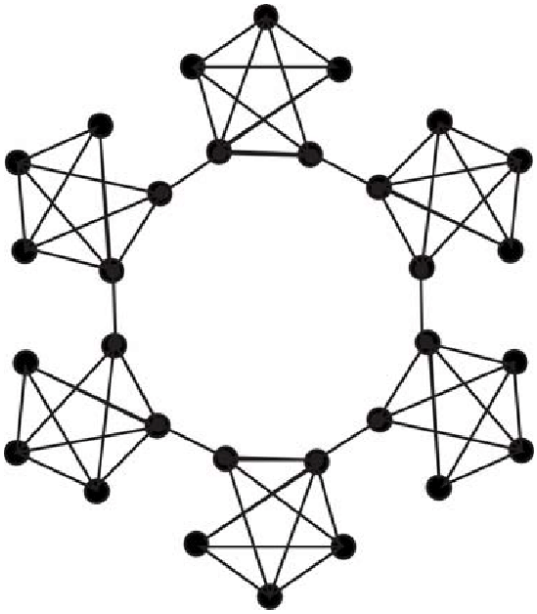


FIGURE 3.6 – Exemple de *connected cavemen graph*. On remarque qu'une arête propre à chaque clique a été redirigée pour créer une relation entre cliques.

Stochastic Block Models (*SBM*) Les modèles de blocs stochastiques [NS01] sont des modèles génératifs utilisés afin de générer des graphes présentant une structure de clusters. Ces graphes sont définis par un nombre n de noeuds, un nombre de clusters l, un vecteur de probabilités $\alpha = (\alpha_1, \ldots, \alpha_l)$ et une matrice $W \in \mathbb{R}^{l \times l}$ où chaque entrée appartient à l'intervalle $[0, 1]$. Le vecteur de probabilités décrit la distribution des noeuds du graphes dans chacune des l communautés.

Par exemple, $\alpha = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ correspond à des noeuds distribués dans 4 clusters distincts de manière équiprobable. La matrice W correspond quant à elle aux probabilités de création d'arêtes entre clusters. Ainsi, la matrice :

$$W = \begin{bmatrix} 0,40 & 0,01 & 0,01 \\ 0,01 & 0,40 & 0,01 \\ 0,01 & 0,01 & 0,40 \end{bmatrix} \tag{3.6}$$

correspond à des probabilités de créations d'arêtes entre noeuds d'un même cluster de $0,40$, et des probabilités de créations d'arêtes entre noeuds de clusters différents de $0,01$. On notera ici la flexibilité de la méthode, permettant la gestion fine des propriétés de chaque cluster. On remarquera que cette formulation est en fait une généralisation d'un graphe aléatoire généré selon le modèle d'Erdös-Renyi [CK01] [ER60].

Lancichinetti–Fortunato–Radicchi (*LFR*) Les générateurs précédents, bien que donnant des réseaux avec une forte structure en communautés, peuvent manquer de réalisme. Il est par ailleurs complexe de contrôler finement les graphes obtenus, de par le manque de paramètres. La méthode proposée par A. Lancichinetti, S. Fortunato et F. Radicchi [LFR08] permet de générer des graphes synthétiques prenant en compte les distributions des degrés des noeuds et des tailles de communautés. Une implémentation de ce générateur peut être retrouvée dans de nombreuses

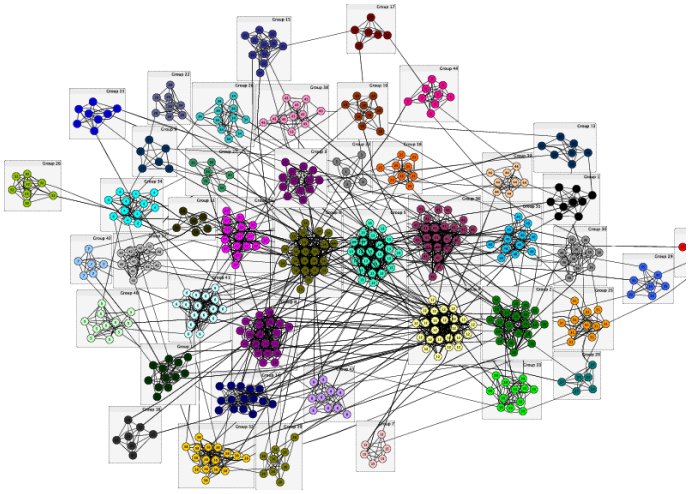


FIGURE 3.7 – Exemple de graphe généré grâce à l'approche LFR, avec les paramètres suivants : $k = 44$, $n = 500$, $\gamma = 2$, $\beta = 1$, $\mu = 0.1$, $d_{moyen} = 10$ et $d_{max=25}$ [PMB13]

bibliothèques, telles que networkx. Un exemple de graphe généré par cette approche est présenté figure 3.7. On constate visuellement que celui-ci semble plus proche d'un graphe que l'on pourrait rencontrer en réalité.

HyGen Metzler et *al.* ont observé que de nombreux graphes semblent suivre un modèle où la distribution de degrés suit une forme d'hyperbole [MGM16]. Ces derniers proposent une méthode de génération de graphes prenant en compte cette observation, *HyGen*. Leur approche permet par ailleurs de générer des communautés n'ayant pas de distribution de degrés quasiment uniformes au sein d'une même communauté, ce qui n'est pas forcément le cas de *LFR*. La figure 3.8 présente les paramètres utilisés dans *HyGen*, à savoir la taille du cœur de la communauté γ et la hauteur de la queue de la distribution H.

De tous ces modèles, le modèle LFR est probablement le plus couramment utilisé dans les tâches d'évaluation d'une méthode de clustering, comme le témoigne le nombre de citations. Bien que cette métrique ne soit pas forcément toujours pertinente, elle indique cependant l'intérêt porté par une méthode, et porte le modèle LFR à la première place dans les méthodes présentées ici, avec plus de 2400 citations sur Google Scholar, contre 1124 pour les SBM.

3.3 Clustering dans les graphes attribués

Les graphes attribués sont des graphes dont les noeuds et/ou les arêtes peuvent être décrits par un ou plusieurs attributs. Dans le cadre de ce document, nous ne traiterons que du cas où les noeuds sont attribués. Plusieurs termes peuvent être retrouvés dans la littérature pour se rapporter à ce même type de graphes : graphe augmenté d'attributs (*attribute augmented graph*) [ZCY09], graphes aux vecteurs d'attributs (*feature vector graph*) [GFBS10], ou encore graphe attribué [CBP13]. Nous utiliserons le terme de graphe attribué dans ce document.

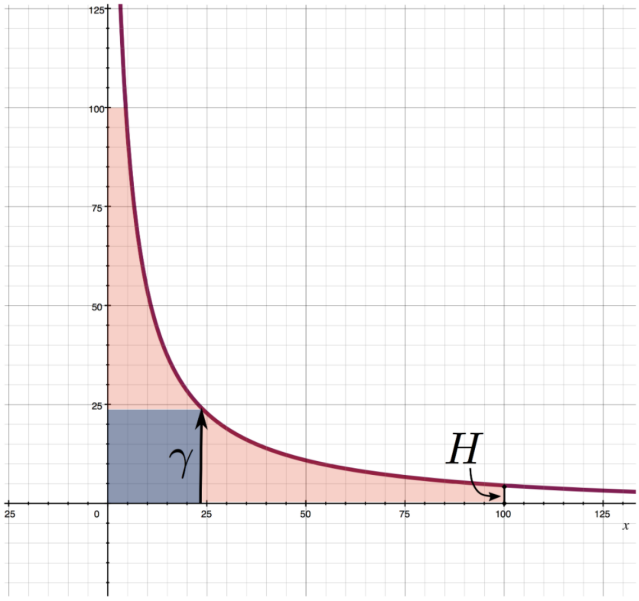


FIGURE 3.8 – Le modèle hyperbolique possède deux paramètres : γ, qui représente la taille du cœur de la communauté, et H, la hauteur de la queue de la distribution. L'influence de ces paramètres sur la distribution est présentée dans [MGM16].

Un graphe dont les noeuds sont attribués [17] est un graphe $G = (V, E, F)$, où V et E représentent les noeuds et les arêtes telles que décrites précédemment, et où chaque noeud $v \in V$ est associé à un vecteur d'attributs de dimension d $(f_1(v), f_2(v), \cdots, f_d(v))$ décrivant les noeuds du graphe.

Le but du clustering dans un graphe attribué reste le même, à savoir la partition de l'ensemble des noeuds V en groupes homogènes. Cependant, ici, l'homogénéité à rechercher n'est pas que structurale – *i.e*, faire en sorte que les noeuds au sein d'une communauté soient plus liés entre eux qu'à ceux des autres clusters, par exemple –, mais aussi sur l'espace des attributs. La figure 3.9 présente un exemple de graphe attribué. Ce dernier est issu de la Edge Box, décrite au début de ce chapitre.

L'extrait présenté dans la figure constitue une petite partie de la base, et représente les relations entre une maladie – ici l'insuffisance cardiaque – et quelques protéines. Pour des raisons de clarté, le nombre de résultats a été ici limité à 50, et l'on constate d'ores et déjà que la lisibilité et la compréhension par l'humain sont complexes, et ce d'autant plus qu'il ne s'agit ici que d'un maigre extrait. À titre d'exemple, le métagraphe complet de la Edge Box est présenté figure 3.10 . On comprend donc ici l'intérêt d'algorithmes permettant d'effectuer ce type de tâches, ou tout du moins les faciliter.

17. Nous utiliserons le terme graphe attribué pour parler du cas des graphes où les noeuds sont attribués par la suite.

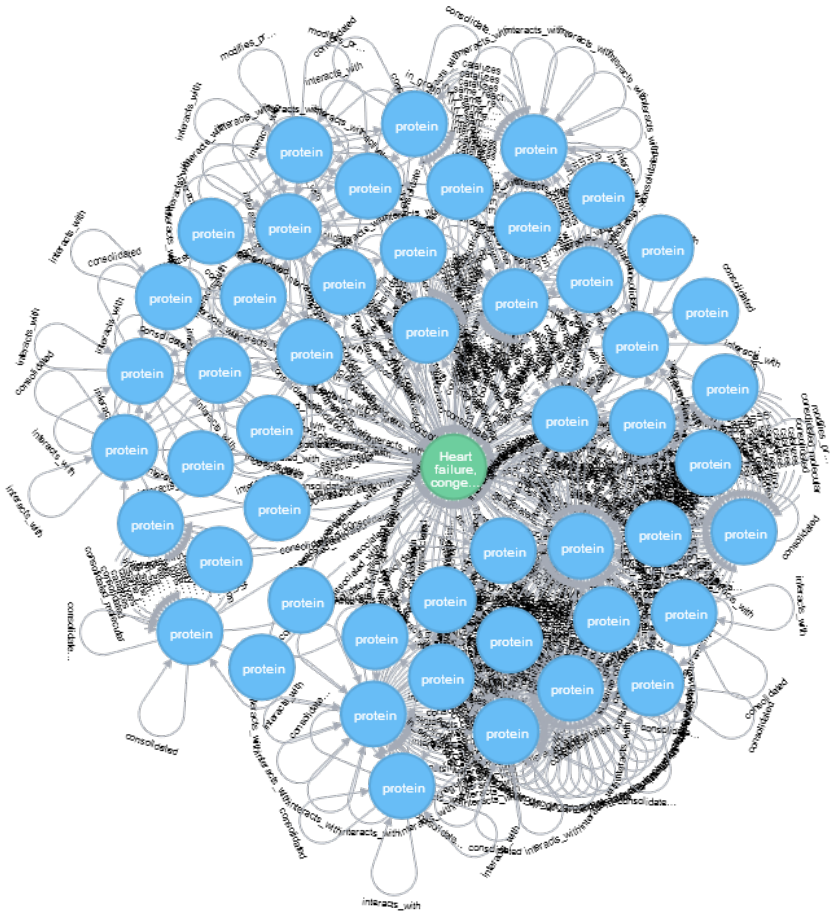


FIGURE 3.9 – Exemple de graphe attribué, issu d'une base de connaissance biomédicale, la FIGHT-HF GK box.

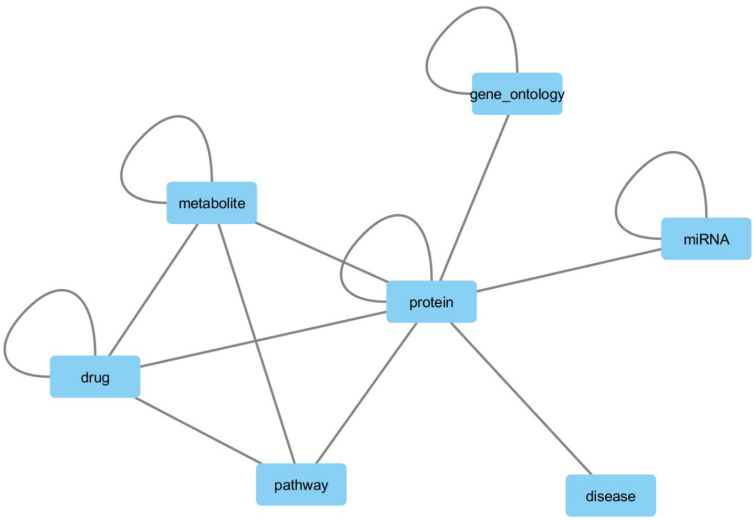


FIGURE 3.10 – Métagraphe de la FIGHT-HF GK Box.

Une revue récente [Chu19] propose de distinguer 3 classes de méthodes de clustering de graphes attribués, en fonction du moment où les informations contenues dans la structure et celles contenues dans les attributs sont fusionnées : (i) les méthodes de fusion précoce (*early fusion*), (ii) celles de fusion simultanée (*simultaneous fusion*) et (iii) celles de fusion tardive (*late fusion*). Bien sûr, il convient de noter que cette classification n'est là encore pas la seule que l'on peut retrouver. Dans [FGKB18], les auteurs proposent une classification en (i) approches topologiques, où les informations contenues dans les attributs sont utilisées comme informations topologiques, (ii) approches *attribute-based* où l'information topologique est fusionnée avec l'information des attributs dans une mesure globale de (dis)similarité, et enfin (iii) les approches hybrides, où les informations contenues dans les deux espaces sont considérées séparément selon une approche d'ensemble. Nous utiliserons cette classification ici, afin de décrire succinctement le panorama des méthodes disponibles. Il convient cependant de ne pas se restreindre à cette classification, utile afin d'organiser ce document, mais là encore imparfaite. Notons par ailleurs que [BCMM15] constitue une bonne référence récente sur les approches de clustering de graphes attribués.

3.3.1 Approches topologiques

Ces approches contournent le problème de l'hétérogénéité structure/attributs en effectuant une fusion des deux espaces de manière préalable à tout clustering. Cette catégorie correspond fortement à celle des méthodes de fusion précoce proposée dans [Chu19]. Il s'agit donc ici de résoudre les problèmes soulevés par l'hétérogénéité du graphe en le transformant en graphe homogène.

Un exemple de ce type d'approches est présenté figure 3.11 et consiste à transformer les proximités entre noeuds dans l'espace des attributs en poids sur leur relation, permettant par la suite d'appliquer une technique de clustering adaptée aux graphes pondérés. Ainsi, plus un ensemble de noeuds est similaire dans l'espace des attributs, plus le poids associé à la relation entre ces noeuds sera élevé. L'application d'un algorithme de clustering adapté aux graphes pondérés permettra de mettre en évidence des groupes de noeuds étant proches non seulement de par leurs relations structurales, mais aussi de par leur proximité dans l'espace des attributs.

L'une des approches les plus simples pour effectuer cette tâche est d'utiliser le *matching coefficient*, qui consiste à compter le nombre d'attributs ayant la même valeur pour chaque paire de noeuds du graphe. Il s'agit notamment de l'approche proposée dans [NAJ03]. Le calcul de cette valeur est présenté équation 3.7, avec le calcul de $s_k(i,j)$ présenté équation 3.8.

$$S_{i,j} = \begin{cases} \sum_k s_k(i,j) & \text{si} \quad (i,j) \in E \quad \text{ou } (j,i) \in E \\ 0 & \text{sinon} \end{cases} \tag{3.7}$$

$$s_k(i,j) = \begin{cases} 1 & \text{si } f_k(i) = f_k(j) \\ 0 & \text{sinon} \end{cases} \tag{3.8}$$

L'utilisation de cette mesure soulève cependant un autre problème : quid des attributs continus ? En effet, la définition du *matching coefficient* fait que cette approche n'est valide que dans les cas d'attributs catégoriels. Une discrétisation est possible, au prix cependant d'une perte d'information. Une autre approche est proposée dans [SC08]. Les auteurs étendent l'approche utilisant le *matching coefficient* en ajoutant le cas où un attribut est continu. Dans ce cas précis, la contribution au poids de l'arête d'un attribut est obtenue par le complément à 1 de la différence

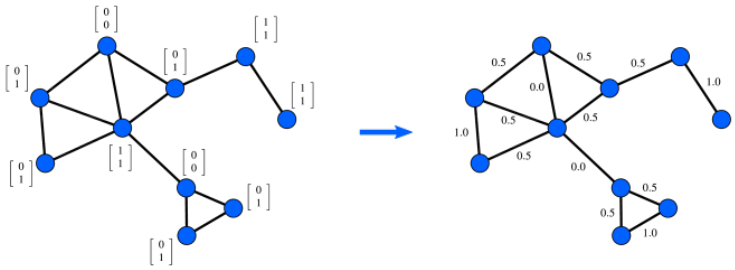


FIGURE 3.11 – Transformation des attributs des noeuds en poids sur les arêtes, permettant l'application de méthodes de clustering adaptées aux graphes non attribués [Chu19].

des valeurs de ces attributs pour les deux noeuds concernés, après une étape de normalisation sur l'intervalle $[0, 1]$. Ainsi, pour deux attributs ayant une valeur proche, cette différence sera proche de 0, et donnera donc un poids proche de 1.

Bien que relativement simple conceptuellement, ces approches ont cependant de grosses limitations. En particulier, nous voyons ici que les poids ne dépendent que des relations d'ordre 1 entre noeuds. Ainsi, uniquement les proximités avec les voisins directs des noeuds sont considérées, ne prenant donc pas en compte des informations potentiellement importantes contenues dans des noeuds plus éloignés dans le graphe.

Un autre type d'approche consiste à transformer l'information contenue dans les attributs en nouveaux noeuds. Dans SA-cluster [ZCY09] [CZY11] par exemple, des noeuds représentant les attributs sont ajoutés au graphe initial, et possèdent des arêtes avec les noeuds ayant cette valeur pour cet attribut. Un exemple présentant l'approche générale est présenté figure 3.12. Une fois cette transformation effectuée, il est ainsi possible d'appliquer une méthode de recherche de communautés adaptée à des graphes non attribués. Bien qu'intéressante et simple conceptuellement, les problèmes de cette approche sont cependant multiples. Tout d'abord, l'augmentation du nombre d'attributs et/ou du nombre de modalités entraîne une explosion du nombre de noeuds et d'arêtes dans le graphe, pouvant poser problème en termes de performances. De plus, cette approche est limitée à des attributs qualitatifs. En effet, la création de noeuds représentant des modalités n'est plus triviale pour un attribut numérique. Une solution est ici la discrétisation de cet attribut, posant d'autres questions sur la méthodologie à appliquer, et ajoutant de la complexité au clustering global. Dans [CZY11], les auteurs proposent certaines solutions afin de contourner ces limitations, telles que l'application de méthodes de diminution de la dimension pour diminuer le nombre de noeuds dans le cas qualitatif avec de nombreuses modalités.

Enfin, certaines approches se basent sur l'optimisation d'une mesure. I-Louvain [CLGEZ15] est par exemple une méthode de ce type. La prise en compte des attributs des noeuds est réalisée en ne considérant plus uniquement la mesure de modularité utilisée par l'algorithme Louvain, mais en y ajoutant une mesure basée sur l'inertie (*inertia-based modularity*), $Q_{inertie}$. Au-delà de cette modification, I-Louvain est très proche de la méthode originale, dans la mesure où il s'agit d'une méthode itérative visant à optimiser ce nouveau critère $QQ^{+}(\mathcal{P})$ [18] $= Q_{NG}(\mathcal{P}) + Q_{inertie}(\mathcal{P})$, où $Q_{NG}(P)$ correspond à la modularité vue précédemment. Dans la classification de [Chu19], il s'agit ici d'une approches de fusion simultanée, prenant en compte la topologie et l'espace des

18. $\mathcal{P}$ représente ici une partition du graphe.

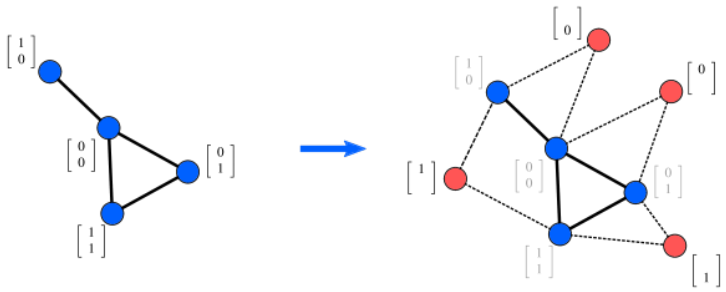


FIGURE 3.12 – Transformation des attributs en noeuds, permettant l'application de méthodes de clustering classiques sur des graphes attribués [Chu19].

attributs au sein d'un même processus. Notons par ailleurs qu'ici, la fusion des deux espaces n'est pas explicite : là où les approches précédentes effectuaient cette étape de fusion en ajoutant des noeuds ou des poids aux relations, ici le graphe d'entrée n'a pas à être modifié.

3.3.2 Approches basées sur les attributs

Ce type d'approches vise à calculer une matrice de (dis)similarités entre chaque paire de noeuds, afin d'y appliquer par la suite des méthodes de clustering plus générales. C'est ici où la classification de [Chu19] diffère de celle de [FGKB18]. En effet, ce type d'approche est classée dans la catégorie des méthodes de fusion précoce, dans la mesure où les informations contenues dans les deux espaces sont ici agrégées préalablement à l'étape de clustering à proprement parler. Le terme *basée sur les attributs* est le choix effectué par [FGKB18].

Cependant, cette terminologie peut porter à confusion, dans la mesure où l'ensemble des méthodes applicables aux graphes attribués doivent à un moment ou à un autre se baser sur les attributs. Bien que n'étant pas tout à fait en accord avec ce choix terminologique, nous avons fait le choix de le garder en tant que titre de section. Ce choix est basé sur le fait que nous nous sommes basés sur la classification proposée par [FGKB18] ici, et respecterons donc les choix effectués par les auteurs.

Le plus couramment dans ces approches, il s'agit d'obtenir une mesure de distance unifiée, en effectuant une combinaison linéaire de deux distances, telle que présenté équation 3.9. Dans cette équation, d_T correspond à la distance topologique, d_S la distance dans l'espace des attributs, et $\alpha \in [0, 1]$ à un paramètre permettant de pondérer l'importance de chaque espace.

$$d(u, v) = \alpha d_T(u, v) + (1 - \alpha) d_S(u, v) \tag{3.9}$$

Cette vision présente de nombreux avantages. Elle laisse notamment une grande liberté quant aux choix des distances à utiliser, ce qui peut être intéressant en fonction des données à explorer. L'expérimentateur peut par exemple appliquer la distance euclidienne ou la similarité cosinus attributs continus des noeuds [CLEZG12], et la joindre à une distance topologique de son choix. Dans [VVOCA13], les auteurs font le choix d'utiliser des noyaux afin de modéliser la notion de proximité. Deux noyaux sont ainsi définis : un noyau K_0 sur la structure – tel que le noyau de chaleur (*heat kernel* défini dans [KL02]) –, et un ensemble de noyaux K_k sur les k variables descriptives des noeuds. Cet ensemble est agrégé par la suite en un noyau unique K_T, obtenu

par la relation présentée équation 3.10.

$$K_T(u,v) = \alpha_0 K_0(u,v) + \sum_k \alpha_k K_k(f(u)_d, f(v)_d). \tag{3.10}$$

Dans cette équation, $\sum_k \alpha_k = 1$. On a donc une pondération plus fine, à l'échelle de l'attribut.

Dans cette approche d'agrégation de dissimilarités, le choix du paramètre α permet par ailleurs de moduler l'importance de chaque aspect. Cela n'est cependant par chose aisée et constitue l'une des limites de ce type d'approches. Bien que laissant une certaine flexibilité, son paramétrage est le plus souvent arbitraire, et on retrouve régulièrement dans la littérature une valeur fixée à $\alpha = 0.5$, donnant la même importance à la topologie et aux attributs.

Il existe cependant possible d'apprendre ce paramètre, ou, plus précisément, de le fixer de manière plus guidée. En effet, le sous-entendu que la topologie et les attributs portent une information de la même importance n'est pas toujours vérifié. Les importances respectives peuvent par ailleurs être dépendantes de la tâche à accomplir. Par exemple, dans le cas d'un clustering d'individus dans un réseau social ayant pour finalité le ciblage d'une campagne publicitaire, il est tout à fait pertinent de considérer que les attributs attachés aux noeuds, tels que les passions ou les activités sportives, soient plus importants que leurs relations.

L'une des solutions peut être de définir une fonction objectif à minimiser ou maximiser, et de l'évaluer en fonction de α. Il est par exemple possible d'utiliser les mesures d'évaluation d'un clustering vues section 2.3.7, et de mesurer leur évolution en fixant α à différentes valeurs. Une approche de ce type est proposée dans [DV12].

Algorithme 5 : Méthode proposée par [DV12] afin d'optimiser la valeur de α, paramètre permettant de pondérer l'agrégation des mesures.

Entrées : un intervalle ϵ
Sortie : Une valeur de α

$\alpha \leftarrow 1$;
$\Delta \leftarrow \infty$;
tant que $\Delta > 0$ **faire**
  Clustering du graphe avec la valeur de α ;
  $\alpha' \leftarrow \alpha - \epsilon$;
  $\Delta \leftarrow (QNewman(\alpha') - QNewman(\alpha)) + (QAttr(\alpha') - QAttr(\alpha))$;
  $\alpha' \leftarrow \alpha$
fin
retourner α

Cela peut cependant être très coûteux en ressources, dans la mesure où l'évaluation d'une seule valeur peut être très chronophage. L'utilisation de méthodes telle que l'optimisation bayésienne peut ici être envisageable, afin d'éviter une recherche exhaustive du meilleur poids.

Parfois, l'agrégation peut par ailleurs être réalisée sur des mesures autres que des distances. Dans [DV12], l'approche proposée par les auteurs est basée sur la modularité de Newman, que nous avons présentée section 3.2. Nous avons cependant vu lors de sa présentation que cette mesure ne permet que de quantifier la qualité d'un clustering sur la base de la topologie du graphe, et pas des attributs des noeuds. Afin de prendre en compte ces derniers, les auteurs proposent d'utiliser une fonction de mesure de la similarité, notée $SimA(i, j)$. Une modularité *composite*, obtenue par agrégation des deux mesures, est obtenue selon l'équation 3.11. Le calcul de $S(i, j)$ est présente équation 3.12. On remarquera que l'on retrouve ici une partie de l'expression de la modularité Q de l'équation 3.4, mais uniquement pour une paire d'individus [19].

$$Q_{composite} = \sum_{C} \sum_{i,j \in C} \alpha S(i, j) + (1 - \alpha) SimA(i, j) \tag{3.11}$$

$$S(i, j) = \frac{1}{2m}(A_{i,j} - \frac{d_i d_j}{2m}) \tag{3.12}$$

Contrairement aux approches topologiques vues précédemment, ce type de méthodes présente de nombreux avantages. Tout d'abord, moins de transformations sont nécessaires, dans la mesure où aucune aucune information non présente à l'état initial dans les données n'est ajoutée – ni noeuds, ni arêtes ou pondérations sur ces dernières. De plus, tout du moins dans le cas d'agrégation de mesures de distance ou de similarité, une très grande liberté est laissée à l'expérimentateur sur les distances à utiliser. Cela peut dépendre notamment des données, car comme nous l'avons déjà discuté section 2.3.1, toutes les distances ne se valent pas en fonction du type de données décrivant les objets à regrouper.

Ces méthodes ne sont cependant pas exemptes de défauts. Leur complexité est de manière générale assez conséquente, dans la mesure où un nombre de valeurs de l'ordre de n^2 doit être calculé, stocké et utilisé dans le pire des cas. De plus, la méthode d'agrégation des différentes mesures est discutable, et peut là encore influencer fortement les mesures obtenues, et donc les clusterings pouvant en découler.

3.3.3 Approches hybrides ou d'ensemble

Enfin, les approches hybrides consistent à considérer les deux informations séparément. Chaque aspect est soumis à un algorithme spécifique, avant d'agréger les résultats de chaque clustering. Ces approches correspondent à la catégorie dite de fusion tardive de [Chu19]. La figure 3.13 résume leur fonctionnement global.

Cette vision de la fusion a pour avantage de permettre l'utilisation de méthodes reconnues, tant dans la recherche de communautés dans les graphes non attribués que dans le clustering de vecteurs d'attributs. Des méthodes d'ensemble de clustering [SG02], que nous avons décrites section 2.19, peuvent être utilisées afin d'agréger les résultats des différents clusterings obtenus.

Cette approche est intéressante, et permet par ailleurs de ne pas se limiter à une combinaison de deux approches uniquement – l'une pour la topologie, l'autre pour les attributs. En effet, cette approche d'ensemble permet de combiner les résultats de nombreux modèles, en fonction

19. La raison est ici que la phase de considération globale de la modularité de Newman est obtenue via les deux sommes, permettant de considérer les objets et leurs clusters respectifs.

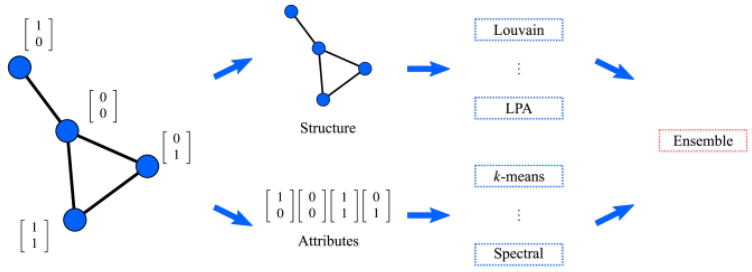


FIGURE 3.13 – Méthode de fusion tardive (*late fusion*, combinant deux approches : (i) une première appliquée à la topologie, et (ii) une deuxième appliquée à l'espace des attributs [Chu19].

de leurs forces et faiblesses. Elle est par ailleurs très similaire aux approches vues dans la section précédente, où des combinaisons de dissimilarités peuvent être utilisées. La différence repose cependant ici sur le moment où les méthodes sont agrégées, et sur leur mode d'agrégation.

On trouve un exemple récent de ce type d'approche de fusion tardive dans [LLZG20]. Leur proposition consiste dans un premier temps à obtenir deux partitions du graphe $\mathcal{P}_s$ et $\mathcal{P}_a$, l'une par le biais d'une méthode classique sur la structure, l'autre basée uniquement sur les attributs. Une fois ces partitions obtenues, deux matrices d'adjacence A_s et A_a, où $d_{ij} = 1$ si i et j sont reliés par une arête dans la partition correspondante, $\mathcal{P}_s$ et $\mathcal{P}_a$, respectivement. Une matrice d'adjacence agrégée est ensuite obtenue par la relation $A = \alpha A_s + (1 - \alpha)A_a$.

Cette approche donne de bons résultats, mais reste sujette aux défauts de ce type d'approches, à savoir une certaine complexité temporelle, ainsi que la dépendance au paramètre α.

On peut enfin retrouver d'autres approches, telles que des méthodes probabilistes [YML13] ou des méthodes d'apprentissage profond [WPH+19].

3.3.4 Évaluation du clustering dans les graphes attribués

Comme pour les graphes non attribués, une approche consiste à générer des graphes synthétiques contenant des communautés. La tâche est ici plus complexe, dans la mesure où la définition de communauté n'est plus aussi simple que dans le cas de noeuds non attribués.

Une possibilité est d'ajouter des attributs manuellement à chaque noeud généré par l'une des approches de génération de graphe simple présentée section 3.2.4. Ces attributs peuvent par exemple être générés par échantillonnage à partir de distributions paramétrées différemment d'un cluster à l'autre.

Des approches spécifiques aux graphes attribués ont par ailleurs été proposées. Dans [LMRZ15], les auteurs proposent une approche permettant de générer des graphes dont les noeuds sont attribués, et ayant des propriétés proches de graphes que l'on peut retrouver en pratique.

Ce modèle est basé sur trois hypothèses :

1. Chaque nouveau noeud est connecté à un noeud existant avec une probabilité proportionnelle au degré de ce noeud. Ce faisant, le modèle respecte la propriété d'attachement préférentiel.
2. L'appartenance d'un noeud à une communauté est liée à la fois au nombre d'arêtes entre ce noeud et ceux de sa communauté, mais aussi à leur proximité dans l'espace des attributs.
3. Enfin, l'homophilie est respectée. Cette hypothèse, présentée notamment dans [MSLC01], stipule que deux noeuds ont une probabilité plus grande d'être reliés s'ils partagent des attributs.

Les auteurs ont par ailleurs étendu par la suite leur méthode afin de permettre la génération de graphes dynamiques, *i.e.* évoluant au cours du temps [LMBZ17].

L'évaluation du clustering effectué sur ces graphes peut ensuite être réalisée via les métriques développées dans le cas des graphes non attribués, section 3.2.4.

3.4 Bilan du chapitre

Nous avons vu dans ce chapitre que la recherche de groupes au sein de ces graphes peut être d'intérêt dans de nombreuses applications, notamment dans celles où les relations entre objets présentent un intérêt en pratique. Bien qu'il ne s'agisse pas d'une tâche aisée, de nombreuses méthodes, basées sur différentes caractéristiques de ce type de structures, ont été proposées. Dans le cas des graphes attribués, la tâche est d'autant plus complexe qu'un nouvel espace de représentation doit être considéré. Là aussi, différentes approches ont été présentées dans la littérature.

Dans cette thèse, nous nous concentrons principalement sur les notions de similarité et dissimilarités avant de nous intéresser aux méthodes [20]. Certaines approches à base de distance ont été proposées dans le cadre des graphes attribués. L'une d'entre elles consiste à agréger les distances obtenues sur la topologie d'une part et sur l'espace des attributs d'autre part. C'est cette approche que nous avons suivie, et que nous reverrons donc dans le cadre de notre contribution présentée au chapitre 5.

20. Bien que les deux soient souvent indissociables dans le cadre du clustering.

4

Forêts d'arbres extrêmement aléatoires pour le calcul de similarités dans les données hétérogènes

Sommaire

4.1 Motivation : place du choix et du calcul de (dis)similarités dans le processus d'ECD

Comme nous l'avons vu au cours des chapitres précédents, de nombreuses méthodes de clustering se basent sur une mesure permettant de quantifier la (dis)similarité entre les objets afin de mettre en évidence des groupes homogènes. Certains algorithmes sont d'ailleurs définis avec une mesure en particulier. C'est le cas par exemple de la méthode des k-moyennes décrites au sein du chapitre 2, où la distance euclidienne est implicitement utilisée, même si certaines implémentations de cet algorithme permettent de modifier cette mesure. D'autres algorithmes laissent plus de liberté quant au choix de la mesure, tels que des algorithmes de clustering hiérarchique, pour lesquels la matrice de (dis)similarités peut être pré-calculée, laissant plus de flexibilité à l'utilisateur.

Le choix d'une mesure dépend donc de l'algorithme et de son implémentation. Comme nous l'avons noté au chapitre 2, le choix d'une telle mesure n'est pas trivial et dépend du type de données. De même, plusieurs étapes du processus d'ECD dépendent de ce choix. En particulier, lors de la phase de prétraitement, il peut être nécessaire de normaliser les données, éliminer les points aberrants, ou encore d'éliminer les variables corrélées, en fonction de la mesure choisie.

Le choix d'une mesure de (dis)similarité est particulièrement délicat dans le cas de données hétérogènes, c'est-à-dire incluant des variables de types différents [21]. Ce type de données, bien que rarement présent dans les *cas d'école*, est très courant dans les applications réelles. Cela est notamment le cas dans les données biomédicales où les variables sont continues (mesures biologiques, données omiques) ou qualitatives (co-morbidités, symptômes, sexe, CSP, prise de médicaments...). Nous avons par exemple abordé le cas concret des données patients recueillies et agrégées dans le cadre du projet FIGHT-HF. L'exploration et l'exploitation de ce type de données s'avère souvent délicate en pratique, notamment pour des expérimentateurs habitués aux méthodes de fouille classiques, sur des données homogènes. En effet, au delà de la question de l'algorithme de clustering à choisir, se pose aussi la question de la mesure à utiliser si cette dernière est un paramètre de la méthode.

Nous nous concentrerons dans ce chapitre sur un cas d'utilisation d'une mesure de (dis)similarité : le clustering de données hétérogènes. Les approches les plus courantes consistent à discrétiser les variables continues [FDM+18] ou à transformer les variables qualitatives en variables quantitatives [GJ03]. Cependant, ces approches obligent souvent à établir des hypothèses sur la distribution des données et impliquent des transformations heuristiques qui entraînent une perte d'information.

L'une des contributions présentée dans cette thèse est une nouvelle méthode d'apprentissage de (dis)similarité basée sur des arbres aléatoires, que nous avons nommée *Unsupervised Extremely randomized Trees* (UET). Le fait que nous nous sommes penchés sur les méthodes à base d'arbres aléatoires n'est pas dû au hasard. En effet, ces structures présentent de nombreux intérêts pratiques, que nous pensions pouvoir être utiles dans le cadre de nos travaux et que nous développerons dans ce chapitre. L'utilisation d'arbres dans les processus non supervisés n'est pas nouvelle : nous nous sommes concentrés sur l'amélioration d'une méthode existante et son extension à d'autres types de données. L'approche que nous proposons peut en effet être appliquée

21. Nous parlons bien ici d'objets décrits sous forme de vecteurs d'attributs ou de tuples d'attributs.

sur des données hétérogènes. Notre motivation est ici de proposer une approche s'appuyant sur une méthode bien connue afin de simplifier les processus d'exploration, préparation et prétraitement des données hétérogènes, en particulier dans le type d'applications présenté précédemment relatives à des données cliniques.

Ce chapitre est organisé de la manière suivante. Après une présentation de l'approche historique des forêts d'arbres aléatoires dans le cadre non supervisé dans la section 4.2, nous décrirons en détail l'approche que nous proposons dans la section 4.3. Dans la section 4.4, nous présenterons les résultats de l'évaluation de notre approche, et montrons qu'elle donne de bons résultats – y compris avec des données cliniques –, tout en ayant des propriétés intéressantes. Enfin, nous discuterons et conclurons ce chapitre dans la section 4.5.

4.2 Une approche versatile de calcul de similarités dans le cadre de données hétérogènes : les forêts d'arbres non supervisées

4.2.1 Les forêts d'arbres aléatoires : URF.

Une approche possible de calcul de dissimilarité entre objets décrits par des variables hétérogènes est de considérer une dissimilarité globale comme une somme pondérée de distances spécifiques. Cette méthode est par exemple celle utilisée dans l'algorithme des k-Prototypes [Hua98], où une dissimilarité $D(x, y)$ entre deux objets x et y est obtenue par la relation présentée équation 4.2.1.

$$D(x, y) = (1 - \alpha)D_e(x, y) + \alpha D_c(x, y) \tag{4.1}$$

Ici, $D_e(x, y)$ correspond à la distance euclidienne pour un sous-ensemble d'attributs continus, $D_c(x, y)$ le nombre d'attributs qualitatifs ayant des valeurs qui diffèrent, et α le paramètre permettant de moduler le poids de chaque distance. Cette approche est similaire à celle utilisée par une mesure populaire, la distance de Gower [Gow71], que nous avons décrite en détail dans la section 2.3.1. Cependant, ces méthodes d'agrégation de mesures présentent une limitation majeure, qui est la nécessité de fixer le paramètre de poids α. La valeur de ce paramètre peut cependant être déterminée par une étape d'optimisation [vdH15]. Ici, nous avons des mesures de dissimilarités *figées*, qui permettent néanmoins de capturer et de représenter la proximité des objets.

Un autre type d'approche consiste à utiliser les méthodes d'apprentissage de distance (*distance learning*), dont l'objectif est d'apprendre une fonction de distance sur les objets. Cet apprentissage peut être effectué dans un cadre non supervisé, par exemple par le biais de méthodes d'apprentissage profond [JHCL18].

Une alternative est d'utiliser des arbres aléatoires. Dans [SH06], Shi *et al.* proposent une méthode basée sur les forêts d'arbres aléatoires, *Unsupervised Random Forest* (*URF*), permettant d'effectuer cette tâche de calcul de dissimilarité entre objets décrits sous forme de vecteurs d'attributs. Les arbres de décision nécessitant des données étiquetées dans leur phase d'induction, il est légitime de se demander comment ces derniers peuvent être utilisés dans un cadre non supervisé. Les auteurs proposent dans leur travail une approche intéressante permettant de s'affranchir des étiquettes, en générant des instances synthétiques. Ces instances permettent par la

suite de construire les arbres sur la base d'une classification binaire discriminant les deux classes, *i.e.* instances observées et instances synthétiques. La méthode qu'ils proposent est basée sur l'intuition que plus les objets sont similaires, plus ils seront dirigés vers les mêmes feuilles au sein d'une forêt. Cette observation permet de dériver une mesure de similarité. L'arbre de décision qui est construit devient ainsi un instrument pour la mesure de similarité.

Les auteurs proposent deux méthodes de génération d'instances, *addcl1* et *addCl2*. Les deux méthodes fonctionnent d'une manière similaire, en réalisant un échantillonnage aléatoire à partir des données observées, mais diffèrent au niveau de la méthode d'échantillonnage : dans addCl1, les instances synthétiques sont obtenues en effectuant un échantillonnage sur les valeurs observées des variables, tandis que dans addCl2, un échantillonnage sur l'hyperrectangle contenant les valeurs observées est effectué. Ainsi, les instances synthétiques peuvent prendre des valeurs non observées dans le jeu de donnée initial, l'échantillonnage étant effectué sur les intervalles compris entre les valeurs minimales et maximales pour chaque variable.

Une étiquette est ensuite attribuée à ces nouveaux objets. Présentons une application concrète de ces deux méthodes de génération. La table 4.1 présente un exemple de jeu de données d'entrée. Les tables 4.2, 4.3 et 4.4 présentent la création de trois instances synthétiques par la méthode *addCl1*.

TABLE 4.1 – Jeu de données initial. La même étiquette est attribuée à chaque instance.

Instance	Attribut #1	Attribut #2	Etiquette
1	5.1	3.5	1
2	7.0	3.2	1
3	6.4	2.8	1

Instance	Attribut #1	Attribut #2
1	5.1	3.5
2	7.0	3.2
3	6.4	2.8

Instance	Attribut #1	Attribut #2	Etiquette
1	5.1	3.5	1
2	7.0	3.2	1
3	6.4	2.8	1
4	5.1	3.2	0

TABLE 4.2 – *addCl1* : une instance synthétique est ajoutée en échantillonnant une valeur observée pour chaque variable du jeu de données initial. Une étiquette différente des données observées est attribuée à cette instance.

Instance	Attribut #1	Attribut #2
1	5.1	3.5
2	7.0	3.2
3	6.4	2.8

Instance	Attribut #1	Attribut #2	Etiquette
1	5.1	3.5	1
2	7.0	3.2	1
3	6.4	2.8	1
4	5,1	3.2	0
5	6.4	3.5	0

TABLE 4.3 – *addCl1* : une seconde instance synthétique est ajoutée selon la même procédure.

Instance	Attribut #1	Attribut #2
1	5.1	3.5
2	7.0	3.2
3	6.4	2.8

Instance	Attribut #1	Attribut #2	Etiquette
1	5.1	3.5	1
2	7.0	3.2	1
3	6.4	2.8	1
4	5,1	3.2	0
5	6,4	3.5	0
6	5.1	2.8	0

TABLE 4.4 – *addCl1* : une troisième instance est ajoutée selon la même procédure. On obtient un jeu de données équilibré.

La méthode de génération *addCl2* utilisant un échantillonnage à partir de l'hyperrectangle pour chaque variable observée, la première étape est de déterminer leurs intervalles. Dans cet exemple, on a :

— Attribut #1 : $[5.1, 7.0]$
— Attribut #2 : $[2.8, 3.5]$

Instance	Attribut #1	Attribut #2	Etiquette
1	5,1	3.5	1
2	7,0	3.2	1
3	6,4	2.8	1
4	5,5	2.9	0
5	6,7	3.1	0
6	5,9	3.4	0

TABLE 4.5 – Jeu de données synthétique obtenu par la méthode *addCl2*.

Maintenant que le jeu de données est étiqueté, une forêt d'arbres aléatoires peut être construite. L'intuition derrière *URF* se base sur une notion que nous avons vue dans la sous-section 2.2.3, qui est qu'un arbre de décision divise un espace en régions, par le biais des partitions effectuées à chaque noeud. Ainsi, deux objets similaires devraient se retrouver plus fréquemment dans la même région de l'espace après division. Cette observation est par ailleurs l'idée utilisée par les arbres d'isolation (*isolation trees*), utilisée dans la recherche d'*outliers* [LTZ08], que nous avons aussi décrite section 2.3.5.

Dans le cas des forêts d'arbres non supervisées, on ne considère que les régions de l'espace les plus petites, c'est-à-dire celles correspondant aux feuilles des arbres de décision. Dans la mesure où l'on peut considérer que chaque feuille contient des objets similaires, une procédure de calcul de similarité peut être définie. En effet, si deux objets i et j sont présents dans la même feuille d'un arbre, la mesure globale de similarité entre ces deux objets est incrémentée de 1. Cette similarité est par la suite normalisée en divisant par le nombre d'arbres dans la forêt, produisant une similarité dans l'intervalle $[0, 1]$. Ceci simplifie la transformation de cette mesure de similarité en mesure de dissimilarité, voire de distance.

4.2.2 Limitations des URF

Cette extension au cadre non supervisé présente plusieurs avantages. Tout d'abord, cela nous offre une méthode de calcul de similarités héritant de certains points forts des arbres aléatoires, telles que la résistance aux variables corrélées. De plus, et il s'agit là du point le plus important à nos yeux, on peut noter que le processus de partition peut être défini en fonction du type de l'attribut choisi, à chaque étape de construction de l'arbre. Il est donc possible d'envisager, d'une part, l'application de la méthode sur des données hétérogènes, et d'autre part, sur des données sous d'autres formes que des vecteurs d'attributs.

Cependant, *URF* présente quelques limites. Premièrement, la phase de génération de données synthétiques est une étape contribuant à la complexité à la méthode. De plus, les résultats obtenus par cette approche dépendent fortement des données synthétiques, et donc de la procédure de génération. En effet, les auteurs précisent dans leur contribution que les dissimilarités obtenues varient fortement en fonction de différentes réalisations des données synthétiques, et recommandent d'agréger les dissimilarités obtenues par plusieurs forêts [SH06], par le biais de la moyenne en particulier. Il s'agit d'une limitation sérieuse de la méthode dans la pratique, pour de gros jeux de données. Ceci peut expliquer pourquoi les forêts d'arbres non supervisés ne sont pas souvent retrouvées dans la littérature, et ce plus d'une décennie après leur présentation. Elles ont néanmoins été utilisées avec succès dans certains cas [SSB+05].

Cette approche nous parait cependant prometteuse, et nous l'avons améliorée dans cette contribution en :

1. Diminuant drastiquement sa complexité, afin de faciliter son utilisation et son déploiement
2. Permettant la prise en compte d'attributs hétérogènes
3. Parallélisant la construction des arbres

4.3 UET : Forêts d'arbres extrêmement aléatoires non supervisés pour le calcul de similarités

4.3.1 Motivation et description

Comme nous l'avons vu précédemment, les limitations principales d'*URF* sont liées à la nécessité de générer des données synthétiques, apportant un biais et ajoutant un niveau de complexité à l'algorithme. Ainsi, s'affranchir de cette phase de génération permettrait de drastiquement améliorer les performances de l'approche.

Nous avons décrit dans la sous-section 2.2.4 les arbres extrêmement aléatoires (*Extremely randomized Trees*, ou *Extra Trees*, *ET*)) [GEW06], un algorithme d'induction d'arbres ajoutant une couche de processus aléatoires par rapport aux forêts d'arbres aléatoires classiques. En effet, là où le seuil de partition est optimisé dans l'algorithme des forêts aléatoires, ce dernier est sélectionné de manière aléatoire dans les ET. La construction de ces forêts dépend fortement de deux paramètres : K, le nombre d'attributs à échantillonner pour chaque partition, et n_{min} le nombre minimum d'objets dans un noeud pour qu'il puisse subir un processus de partition. En d'autres termes, plus la valeur de n_{min} est faible, plus les arbres seront profonds et plus le nombre de feuilles sera grand. Ce paramètre est nommé *smoothing parameter*, et peut être précieux dans le cas de données bruitées. En effet, de grandes valeurs de n_{min} ont tendance

à donner de meilleurs résultats dans ce cas [GEW06]. Le nombre d'arbres M est bien sûr un paramètre à spécifier afin de construire la forêt, mais est là bien plus classique.

L'un des points particulièrement intéressants est le paramètre K. En effet, pour une valeur $K = 1$, le processus de partition aléatoire ne se fait que sur une variable. Les arbres ainsi obtenus ne dépendent donc plus des étiquettes, dans la mesure où l'ensemble des partitions est effectué de manière aléatoire, sans contrôle des classes d'appartenance. Il est donc possible de tirer parti de cette indépendance vis-à-vis des étiquettes dans le cadre non supervisé. En effet, la génération de données synthétiques est hypothétiquement superflue dans ce cas, et les arbres ainsi construits peuvent être utilisés de la même manière que dans l'algorithme des URF. K ne sera donc plus un paramètre dans la méthode que nous proposons, dans la mesure où il sera fixé à 1, et les partitions se feront donc totalement aléatoirement.

Suite à cette constatation, nous avons dans un premier temps proposé *addCl3*, une procédure ne générant pas d'instances synthétiques, mais étiquetant le jeu de données de manière aléatoire. Son implémentation est triviale, et consiste à créer une liste contenant $\lfloor \frac{n_{obs}}{2} \rfloor$ fois l'étiquette 0 et $1 - \lfloor \frac{n_{obs}}{2} \rfloor$ fois l'étiquette 1. Pour chaque objet du jeu de données, une étiquette est tirée sans remplacement de cette liste.

Cette approche est bien plus économe en ressources qu'*addCl1* et *addCl2*, tant en termes de puissance de calcul que de mémoire nécessaires, dans la mesure où aucun objet synthétique ne doit être créé. Cependant, elle souffre d'une de leurs limitations, qui est que les arbres construits dépendent fortement des réalisations objets-étiquettes. Ainsi, deux étiquetages différents peuvent mener à des arbres complètement différents, de manière totalement arbitraire. L'une des solutions à ce problème est de construire plusieurs forêts sur la base de différents jeux de données augmentés des données synthétiques, et d'agréger leurs résultats. Dans [SH06], les auteurs ont constaté qu'effectuer la moyenne des résultats de 5 forêts de 5000 arbres chacune semblait donner un modèle robuste.

Cette astuce pénalise cependant la méthode en termes d'utilisation de ressources. L'une des approches proposées par un relecteur de [SH06] est de plutôt construire une seule forêt sur une grande quantité de données générées – à condition d'effectuer une pondération afin de prendre en compte le déséquilibre entre les étiquettes.

Il est possible d'aller encore plus loin. En effet, dans la mesure où les arbres sont induits de manière totalement aléatoire sans considération aucune des étiquettes, il est possible de s'affranchir complètement de toute phase de génération, qu'elle soit d'instances ou d'étiquettes. Nous proposons donc un nouvel algorithme, les forêts d'arbres extrêmement aléatoires non supervisés (*Unsupervised Extra Trees*, *UET*), basé sur un principe similaire à celui des *URF*. Cet algorithme est présenté dans l'algorithme 6. Un élément intéressant de cet algorithme est que le calcul de la similarité peut aussi être défini de diverses manières, faisant de la méthode une approche particulièrement modulaire. Un exemple de méthode de calcul est présenté dans l'algorithme 7. On peut voir ici UET comme une méthode qui construit les arbres, auxquels on applique un *module* permettant de calculer une similarité.

La fonction *ExtraTreesEnsemble* est celle présentée dans l'algorithme 3, à savoir la méthode originale présentée dans [GEW06]. Une forêt d'arbres extrêmement aléatoires est donc construite

Algorithme 6 : Algorithme des Unsupervised Extremely Randomised Trees

Entrées : $\mathcal{L}$, M, n_{min}
Sortie : Une matrice de similarité S

$K \leftarrow 1$;
$T \leftarrow ExtraTreesEnsemble(\mathcal{L}, M, n_{min}, K = 1)$;
$S = calcul\_similarit(\mathcal{L}, T, M)$;
retourner S

Algorithme 7 : Algorithme permettant d'obtenir une similarité à partir des ensembles d'arbres extrêmement aléatoires, *calcul_similarit*.

Entrées : $\mathcal{L}, T, M$
Sortie : S

$S \leftarrow 0_{n,n}$;
pour $\mathbf{x}_i \in \mathcal{L}$ **faire**
  pour $\mathbf{x}_j \in \mathcal{L}$ **faire**
    $S_{i,j}$ = nombre de coocurrences des objets $\mathbf{x}_i$ et $\mathbf{x}_j$ dans les feuilles de chaque arbre de T;
  fin
fin
retourner $\frac{S}{M}$

sur la base des données $\mathcal{L}$. Dans l'implémentation présentée, une matrice de similarité, de taille correspondant aux nombres de paires d'objets $N \times N$ est initialisée. Pour chaque arbre de la forêt, les cooccurrences d'objets dans les feuilles sont énumérées afin de mettre à jour les similarités : les valeurs de similarités sont incrémentées pour chaque paire d'objets cooccurrant dans une feuille. Enfin, cette matrice est normalisée par le nombre d'arbres, afin d'obtenir des valeurs dans l'intervalle $[0, 1]$. Nous nommerons cette méthode l'approche fréquentiste de calcul de similarité.

La complexité de la phase de construction de la forêt d'arbres est de l'ordre de $Nlog(N)$. Cependant, la construction de la matrice de similarité fait que la complexité globale de la méthode est de l'ordre de $O(N^2)$.

L'une des caractéristiques intéressantes de cette approche est que le processus de partition peut être effectué sur de nombreux types de données, si tant est que le couple (type de données, procédure de partition) soit bien défini. Par exemple, pour une variable continue prenant des valeurs dans l'ensemble A, la partition peut être effectuée par échantillonnage dans $\mathcal{U}(min(A), max(A))$, où $\mathcal{U}(a, b)$ représente la distribution uniforme ayant a et b comme bornes. Une fois ce seuil choisi, la partition peut être effectuée en dirigeant les objets vers l'un ou l'autre des noeuds enfants en fonction de la valeur prise par l'attribut considéré. Cette méthode de partition n'a cependant du sens que dans le cadre des variables numériques (continues ou discrètes) ou ordinales, où la notion d'ordre existe. Pour des variables purement qualitatives, un tel ordre n'est pas défini.

Un bon exemple est donné dans [KR09] :

> [...] we could choose 1 = blue eyes, 2 = brown eyes, 3 = green eyes and 4 = gray eyes. [...] it is not because gray eyes are given a higher code number than brown eyes

that they would in some sense be better.

Il est possible d'adapter l'approche de partition dans ce cadre. À partir de l'ensemble des modalités pour la variable qualitative considérée, l'on peut tirer de manière aléatoire l'une d'entre elles. Il est ensuite possible d'effectuer une partition sur la base de cette variable, tous les objets ayant cette modalité étant dirigés vers un premier noeud enfant, les autres vers le second.

L'algorithme 8 résume ces deux approches de partition possible, telles qu'implémentées. Il s'agit ici de la fonction *pick_a_random_split* mentionnée dans la section 2.2.3 lorsque nous avons discuté des arbres extrêmement aléatoires.

Algorithme 8 : Processus de partition

Entrées : Liste des valeurs A pour une variable tirée a, son type t
Output : une partition s

si $t ==$ *numrique ou* $t ==$ *ordinal* **alors**
  valeur de division a_c tirée de $\mathcal{U}(min(A), max(A))$;
  retourner *la division* $[a < a_c]$ *et* $[a \geq a_c]$*;*
fin
si $t ==$ *qualitative* **alors**
  valeur de division a_c tirée de set(A);
  retourner *la division* $[a \in unique(A) \backslash a_c]$ *et* $[a = a_c]$
fin

Nous avons ici présenté globalement les principes l'algorithme des UET. Abordons maintenant ses implémentations.

4.3.2 Implémentations

Plusieurs implémentations sont possibles. Le premier chemin que nous avons suivi est celui de l'utilisation de la structure d'arbres. Cette dernière a pour avantage de coller à la vision conceptuelle de l'approche. Cependant, il ne s'agit pas de l'implémentation la plus performante. En effet, certains de ses éléments peuvent être coûteux, notamment en mémoire. Il s'agit par exemple du stockage de diverses informations, telles que les objets contenus dans chaque noeud, les divisions, etc. Bien que minime dans certains cas, cela peut rapidement se faire sentir dans les méthodes d'ensemble, dans la mesure où l'algorithme de construction d'arbres est répété un grand nombre de fois. Pour les jeux de données de taille plus conséquente que les classiques jeux d'évaluation (*Iris*, *Soybean*, etc.), ce surcoût peut devenir problématique.

Nous nous sommes notamment heurtés à ces problèmes de *passage à l'échelle*[22] lors de l'évaluation de notre méthode sur de plus gros jeux de données, que nous verrons dans les sections suivantes. Dans la mesure où nos procédures d'évaluation consistent souvent à répéter une même expérience plusieurs fois, le moindre surcoût devient significatif, et limite le nombre d'expériences qu'il est possible d'effectuer. Il est par ailleurs assez problématique pour un utilisateur final d'avoir besoin d'une grande quantité de ressources pour déployer la méthode.

22. Tout relatif, dans la mesure où il ne s'agit pas de jeu de données massif.

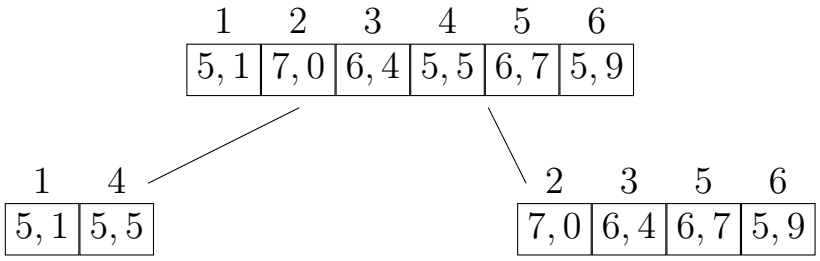


FIGURE 4.1 – Exemple de partition sur le jeu de données présenté à la table 4.6. Les cellules représentent les valeurs prises par l'attribut considéré. Au-dessus de chaque valeur, on retrouve l'identifiant de l'objet correspondant.

Une autre approche beaucoup moins coûteuse en ressources est de ne pas se baser sur des structures d'arbres classiques, mais de ne considérer que des listes d'identifiants des objets. Une partition revient ainsi à obtenir deux nouvelles listes, contenant les identifiants des objets *à gauche*, et les identifiants des objets *à droite*. Les données, *i.e.* les vecteurs d'attributs correspondant à chaque identifiant, ne sont ainsi stockées qu'à un seul endroit de la mémoire. Lorsqu'une partition est réalisée, un sous-ensemble de la colonne correspondant à l'attribut considéré est extrait. Ce sous-ensemble ne contient que les objets concernés par la partition, et peut donc être obtenu en utilisant les identifiants des objets comme indices. Le seuil de la partition est sélectionné et comparé aux valeurs individuelles pour chaque objet, avant de diviser la liste des identifiants en deux, en fonction de ces valeurs. Il est par ailleurs possible de supprimer les informations du niveau précédent à chaque nouveau niveau de *l'arbre*, une fois les calculs nécessaires réalisés, dans l'objectif de réduire d'autant plus l'empreinte mémoire.

Afin d'illustrer nos propos, considérons le jeu de données présenté dans la table 4.6, constitué de six objets, décrits par deux attributs. Supposons que la partition s'effectue ici sur l'attribut #1, et que le seuil est fixé à $5,6$. La figure 4.1 présente un exemple de partition dans ce cas. On retrouve les valeurs prises par l'attribut #1 dans la première liste, ainsi que les identifiants des objets correspondants. La partition est ensuite effectuée, et les objets 1 et 4 se retrouvent dans le même noeud, et les objets 2, 3, 5 et 6 se retrouvent dans un autre noeud. Ainsi, en ne considérant que les valeurs pointées par ces identifiants, il est possible de ne travailler qu'avec ces derniers : on peut ici dire par exemple que le noeud enfant de gauche contient les objets 1 et 4. Effectuer une partition revient donc à rechercher dans le jeu de données initial les valeurs pointées par les identifiants de la liste en cours, les comparer au seuil, et créer deux sous listes d'identifiants en conséquence.

Identifiant	Attribut #1	Attribut #2
1	5,1	3,5
2	7,0	3,2
3	6,4	2,8
4	5,5	2,9
5	6,7	3,1
6	5,9	3,4

TABLE 4.6 – Exemple de jeu de données sur lequel est appliqué UET.

Au-delà de l'intérêt de cette implémentation en termes d'utilisation de mémoire, il s'avère qu'elle est aussi intéressante en termes de temps de calcul. En effet, l'utilisation de structures simples telles que les listes et d'opérations classiques sur ces dernières permet de s'appuyer sur des efforts d'optimisation au niveau des langages de programmations et des compilateurs. Elle se prête par ailleurs particulièrement au stockage dans certains types de bases de données telles que les bases orientées colonne. Elle a cependant une limitation vis-à-vis de l'implémentation utilisant des structures d'arbres : la réutilisation des arbres construits. En effet, stocker les arbres induits peut permettre notamment d'appliquer la méthode à de nouveaux objets sans avoir à reconstruire une forêt complète, ou encore pour ajouter de nouveaux arbres à une forêt existante.

Enfin, il est tout à fait possible de paralléliser l'algorithme. Dans la mesure où une forêt est constituée de plusieurs arbres indépendants, il est en effet relativement simple d'envisager l'induction d'arbres individuels sur des fils d'exécution (*threads*) différents. Il faudra toutefois considérer les accès concurrents à l'objet commun depuis l'ensemble des *threads*, ici, la matrice de similarité globale, cette dernière étant modifiée à la fin de chaque construction d'arbre.

Nous avons fait le choix de l'implémentation sous forme de listes, que nous avons parallélisée. Celle-ci est effectuée en C++, et s'appuie massivement sur la bibliothèque Eigen [GJ+10] pour le travail sur les matrices et OpenMP [Ope08] pour le parallélisme. Le code est disponible sur le dépôt Git suivant `https://gitlab.inria.fr/kdalleau/uetcpp`.

Maintenant que nous avons décrit l'approche que nous proposons ainsi que son implémentation, plusieurs questions se posent, notamment sur le plan des performances. Avant d'aborder dans la section suivante son évaluation empirique, il est nécessaire d'effectuer une évaluation de l'influence des paramètres sur les similarités calculées, afin de ne pas se lancer dans des évaluations à l'aveugle.

4.3.3 Calibration de la méthode

Rappelons que la construction d'une forêt d'arbres extrêmement aléatoires est paramétrée par trois éléments :

1. K, le nombre d'attributs aléatoirement sélectionnés afin de déterminer une partition. Ce dernier permet de moduler le processus de sélection d'attributs,
2. n_{min} ou *smoothing parameter*, correspondant au nombre d'objets suffisant pour pouvoir effectuer une partition sur un noeud,
3. M, le nombre d'arbres par forêt.

Dans l'approche que nous proposons, nous avons vu que le paramètre K n'en est plus un, dans la mesure où nous souhaitons obtenir des arbres totalement aléatoires. Cependant, il serait intéressant d'évaluer ses performances en fonction de n_{min} et M. C'est ce que nous avons fait, les résultats obtenus étant présentés dans cette section.

Pour chaque évaluation présentée ici, la procédure suivante est répétée 10 fois :

1. Construire une matrice de similarité S avec UET,
2. Transformer cette matrice de similarité *via* en matrice de dissimilarité D en utilisant la transformation $D = \sqrt{1 - S}$ [SH06],

Dataset	# Objets	# Variables	# Classes
Iris	150	4	3
Wine	178	13	3
Wisconsin	699	9	2

TABLE 4.7 – Caractéristiques des 3 jeux de données utilisés.

3. Effectuer un clustering hiérarchique agglomératif sur la base de cette matrice de distance.
4. Calculer l'*ARI*.

Trois jeux de données sont utilisés afin d'évaluer l'influence des paramètres sur notre méthode : *Iris*, *Wine* et *Wisconsin*. Leurs caractéristiques sont présentées dans la table 4.7.

4.3.3.1 Influence du nombre d'arbres

L'influence du nombre d'arbres a aussi été étudiée dans [GEW06], où ce paramètre est nommé *averaging strength*. Pour des méthodes stochastiques telles que les forêts d'arbres (extrêmement) aléatoires – dans leur cadre supervisé habituel –, l'erreur moyenne est une fonction décroissante de M. En effet, comme nous avons pu le voir dans la section 2.2.4, augmenter le nombre de prédicteurs permet de réduire la variance.

Il est intéressant d'évaluer l'influence de ce paramètre sur les résultats de notre méthode, dans le cadre non supervisé. Pour ce faire, nous avons suivi le protocole décrit précédemment, en faisant varier M. Nous avons ensuite comparé les ARI en utilisant le test de Kruskal-Wallis [KW52]. Ce test non paramétrique nous permet de vérifier s'il existe bien une différence statistiquement significative entre les ARI obtenues.

Les résultats sont présentés à la figure 4.2. Les valeurs de M testées vont de 50 à 400, avec un pas de 50. On constate qu'il y a très peu de différences entre les ARI obtenues pour les 3 jeux de données. Le test statistique confirme cette observation ($p > 0.1$). Cela confirme par ailleurs l'observation faite par les auteurs de [GEW06], où des valeurs de $M > 40$ donnent de bons résultats.

Cependant, cette valeur ne peut être considérée comme vérité absolue. En effet, nous avons effectué cette étude préliminaire sur un nombre très réduit de jeux de données, qui ne sont pas représentatifs de la vaste quantité de jeux de données disponibles. Ceux-ci sont par ailleurs de faible taille et de faible dimension. Afin d'avoir un peu de marge, il serait donc préférable de sélectionner un nombre d'arbres plus grand pour la suite de nos expériences. Cependant, le temps de construction de la forêt étant une fonction croissante du nombre d'arbres, il est aussi préférable de garder ce nombre raisonnable. Nous avons fait le choix de fixer $M = 200$ comme valeur par défaut. Notons que cette valeur reste bien inférieure à celle recommandée dans les URF, à savoir 5000.

4.3.3.2 Influence de n_{min}

Les forêts d'arbres extrêmement aléatoires ont tendance à produire des arbres ayant trois à quatre fois plus de feuilles que les RF. Dans la mesure où dans notre approche les similarités

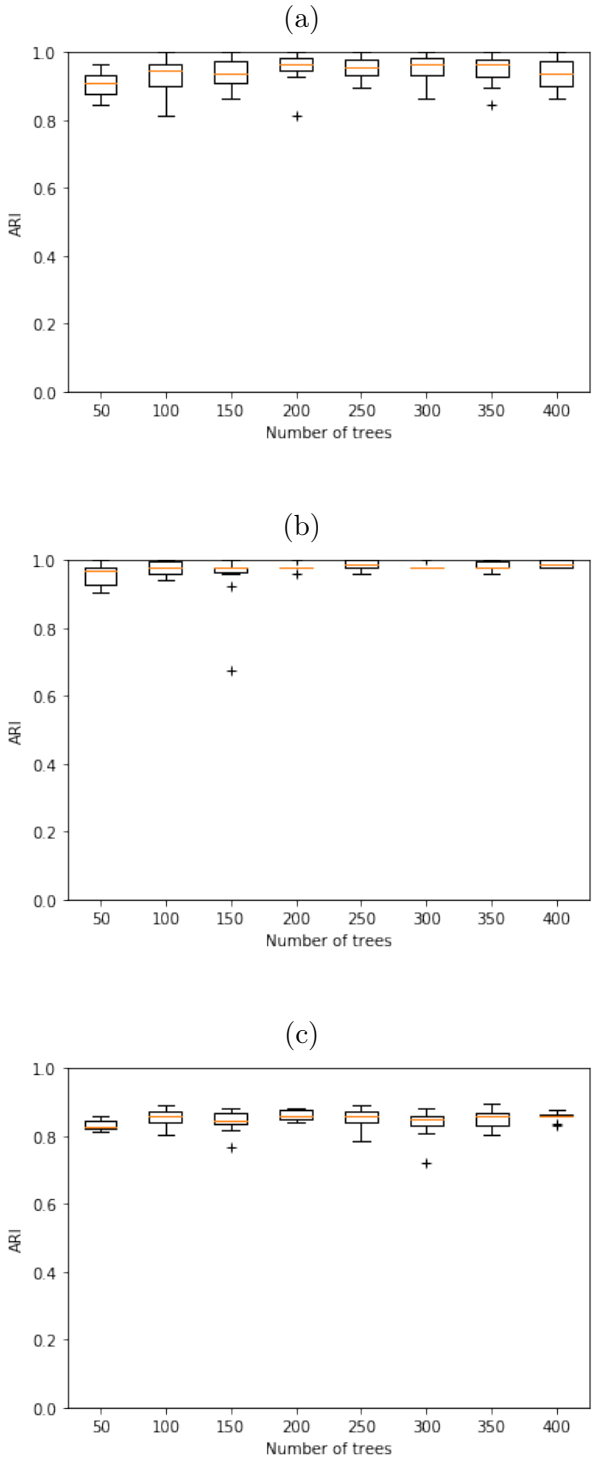


FIGURE 4.2 – ARI obtenues en effectuant un clustering agglomératif sur la base des distances obtenues par UET sur les jeux de données Wine (a), Iris (b) et Wisconsin (c) lorsque le nombre d'arbres M varie. Les ARI restent relativement stables.

sont obtenues par comptage de cooccurrences nombre de co-occurrences dans les feuilles, cela peut avoir un impact majeur sur les résultats. Il est ainsi intéressant de contrôler la croissance des arbres afin de grouper des objets similaires dans les mêmes feuilles plus fréquemment.

Le paramètre de lissage n_{min} est une manière de contrôler cette croissance. Comme indiqué par Geurts *et al.*, la valeur optimale de ce paramètre dépend du bruit présent dans les données. Ils montrent en effet dans [GEW06] que des valeurs plus grandes de n_{min} sont nécessaires lorsque les données sont bruitées. Dans les UET, le bruit est extrême, dans la mesure où notre approche revient à avoir des étiquettes attribuées aléatoirement.

Comme précédemment, nous appliquons le même protocole afin d'avoir une première vision de l'influence de ce paramètre sur les ARI. Nous faisons ici prendre à n_{min} des valeurs correspondant à des pourcentages du nombre d'objets dans chaque jeu de données, dans la mesure où ces valeurs sont plus intéressantes que des valeurs fixes pour tous les jeux de données.

Les résultats sont présentés à la figure 4.3. On constate ici que les résultats sont très sensibles aux variations de ce paramètre. Le test de Kruskal-Wallis nous donne ici effectivement des différences statistiquement significatives ($p < 0.1$). Des valeurs de n_{min} entre 20% et 30% du nombre d'objets semblent donner les meilleurs résultats.

Ces résultats préliminaires sur l'influence de ces deux paramètres sur UET semblent nous indiquer que l'approche est robuste vis-à-vis du nombre d'arbres dans l'ensemble, mais très sensible à la taille des arbres, plus précisément au nombre d'individus requis pour considérer un noeud comme une feuille.

4.3.4 Procédure de détermination automatique des paramètres

Comme nous venons de le voir, certains paramètres peuvent avoir une influence importante sur la qualité des similarités obtenues. L'ensemble de nos évaluations a été réalisé sur la base des paramètres *optimaux* déterminés par nos expériences empiriques.

Cependant, cette *optimalité* est discutable. En effet, les expériences ayant servi à la détermination de ces valeurs sont d'un nombre très limité, et sur peu de nombre de jeux de données. Il est difficile de croire que, bien que les résultats soient congruents, ces valeurs constituent une vérité absolue dans l'ensemble des cas. Il est donc important de rechercher une méthode permettant de fixer ces paramètres en fonction des jeux de données, afin d'obtenir les meilleures (dis)similarités possibles [23].

La difficulté réside ici dans le fait que cette optimisation de paramètres s'effectue dans un cadre non supervisé. Sans données étiquetées, il est en effet nécessaire de quantifier la qualité de la mesure obtenue sans *vérité terrain*, issues d'avis d'experts ou de connaissances métierC. Plusieurs approches sont cependant possibles.

23. De nouveau, nous jugeons la qualité des similarités obtenues en ayant une tâche de clustering comme objectif.

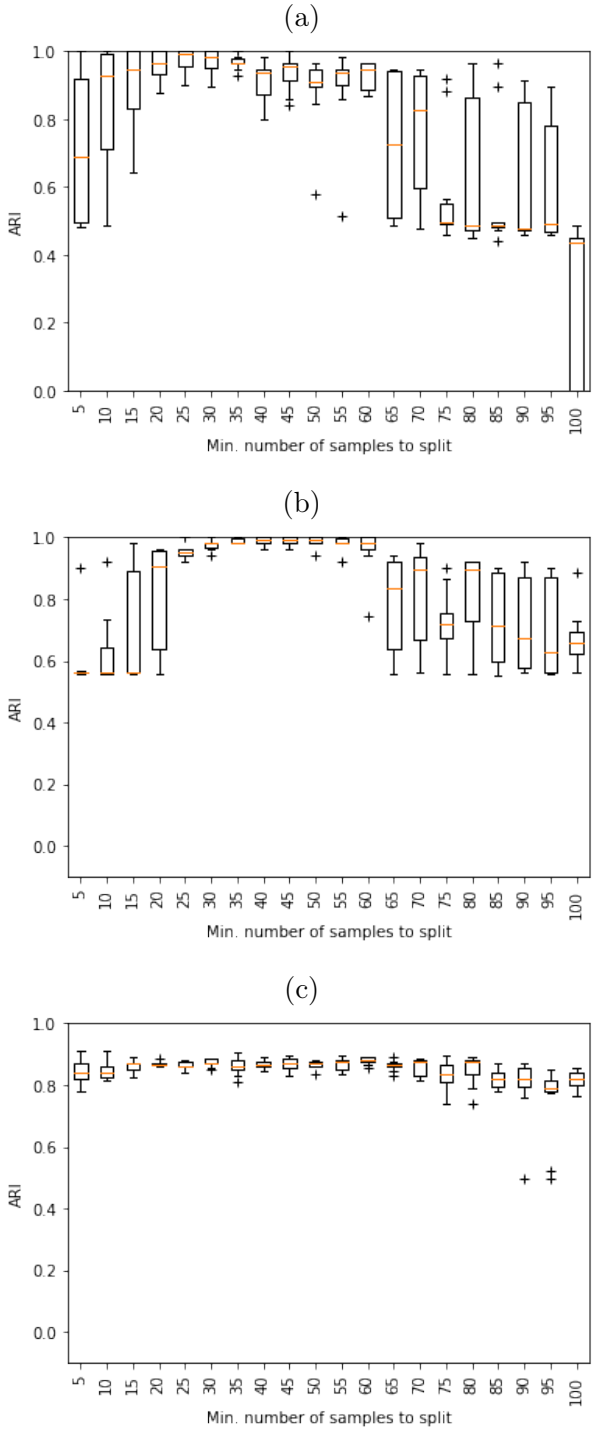


FIGURE 4.3 – ARI obtenues en effectuant un clustering agglomératif sur la base des distances obtenues par UET sur les jeux de données Wine (a), Iris (b) and Wisconsin (c) en faisant varier n_{min}. Les valeurs en abscisse correspondent à des pourcentages du nombre d'objets dans les jeux de données.

Jeu de données	#Instances	#Variables	#Classes
Synth1	282	7	2
Iris	150	4	3
Wine	178	13	3
Wisconsin	699	9	2
Lung	32	56	3
Breast tissue	106	9	6
Pima	768	8	2
Parkinson	195	22	2
Ionosphere	351	34	2

TABLE 4.8 – Jeux de données utilisés pour l'évaluation de la procédure de détermination automatique des paramètres

Premièrement, il est possible d'utiliser des mesures de qualité intrinsèques telles que celles que nous avons décrites dans la section 2.3, dédiée aux méthodes non supervisées. Cependant, cela nécessiterait d'effectuer un clustering à ce stade, ce qui n'est pas souhaitable dans la mesure où (i) le choix d'un algorithme de clustering induirait un biais supplémentaire et (ii) la complexité de la méthode d'optimisation de paramètres pourrait être fortement accrue.

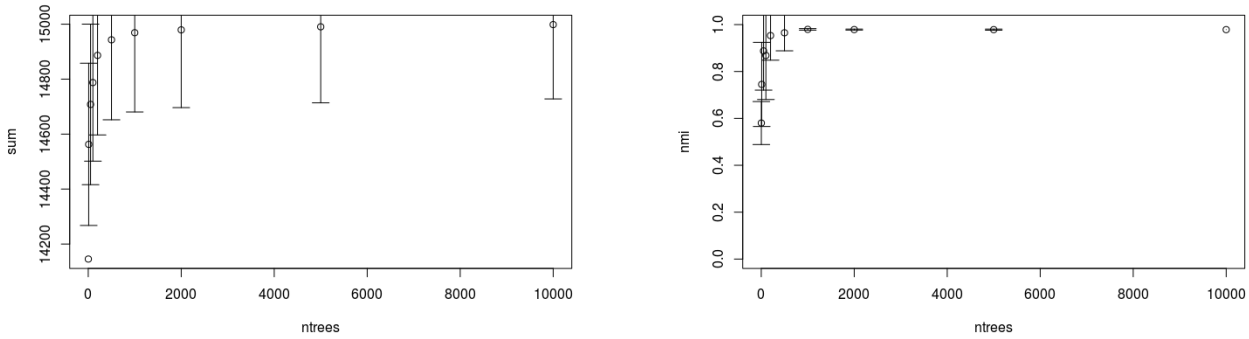
Une autre option est d'utiliser directement les similarités obtenues. Concernant le nombre d'arbres optimal, on sait qu'au-delà d'un certain nombre d'arbres, le fait d'en ajouter n'améliore pas les performances. Ainsi, une possibilité est d'estimer le nombre d'arbres pour lequel les similarités obtenues convergent, *i.e.* restent stables en ajoutant des arbres. Une approche consiste à calculer pour différentes valeurs de M la valeur de la somme des similarités obtenues. Considérer la somme plutôt que quelques valeurs permet la prise en compte globale des similarités, et limite le risque de ne considérer que d'éventuels *outliers*. Il est aussi intéressant d'évaluer l'évolution de cette somme en fonction de n_{min}.

Le protocole est le suivant. Pour chaque valeur de M et n_{min} et chaque jeu de données, nous calculons la somme des dissimilarités, notée DS_Σ. Dans la mesure où les jeux de données utilisés ici sont étiquetés, nous calculons aussi la NMI. La comparaison de l'évolution de la NMI et celle de la somme des similarités en fonction du nombre d'arbres et de n_{min} est ainsi possible. Les similarités sont normalisées au préalable par la formule présentée équation 4.2, avant d'être transformées en dissimilarité par la même relation qu'utilisée préalablement dans ce document, à savoir $\sqrt{1-S}$. Les expériences sont répétées 20 fois pour chaque valeur, et les moyennes et écarts-types sont reportés.

$$\frac{S - min(S)}{max(S) - min(S)} \tag{4.2}$$

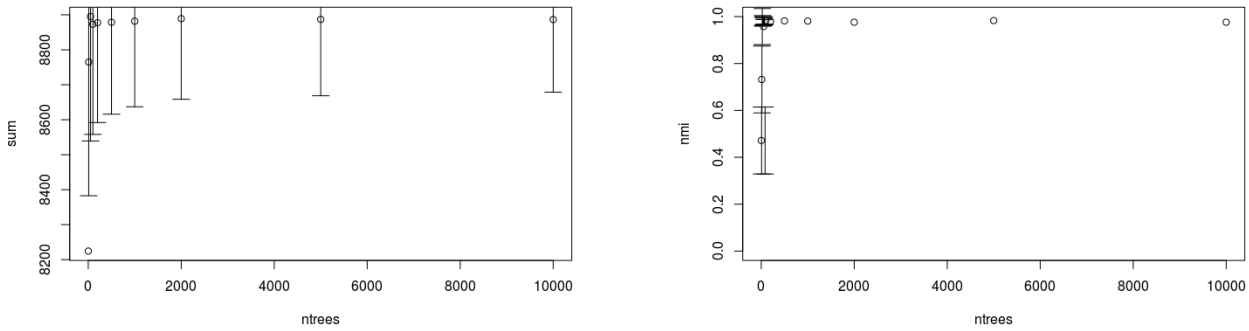
Les jeux de données utilisés sont présentés dans la table 4.8.

Nombre d'arbres Nous avons considéré des valeurs de
$M \in (2, 10, 50, 100, 200, 500, 1000, 2000, 5000, 1000)$. Les valeurs de n_{min} sont ici fixées à 30% du nombre d'objets. Les figures 4.4, 4.5, 4.6 et 4.7 présentent les évolutions de DS_Σ et de NMI en fonction des valeurs de M. Pour des raisons de concision, nous avons fait le choix de ne présenter



(a) (b)

FIGURE 4.4 – Évolution de la somme DS_Σ (a) et de la NMI (b) en fonction de M (ou n_{trees}) pour le jeu de données Synth1



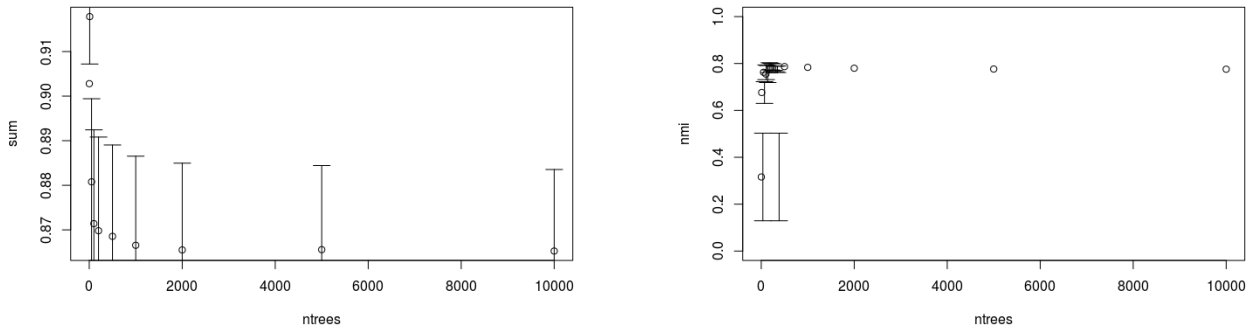
(a) (b)

FIGURE 4.5 – Évolution de DS_Σ (a) et de la NMI (b) en fonction de M (ou n_{trees}) pour le jeu de données Iris

que ces quatre jeux de données, dans la mesure où les résultats pour les autres jeux de données sont similaires.

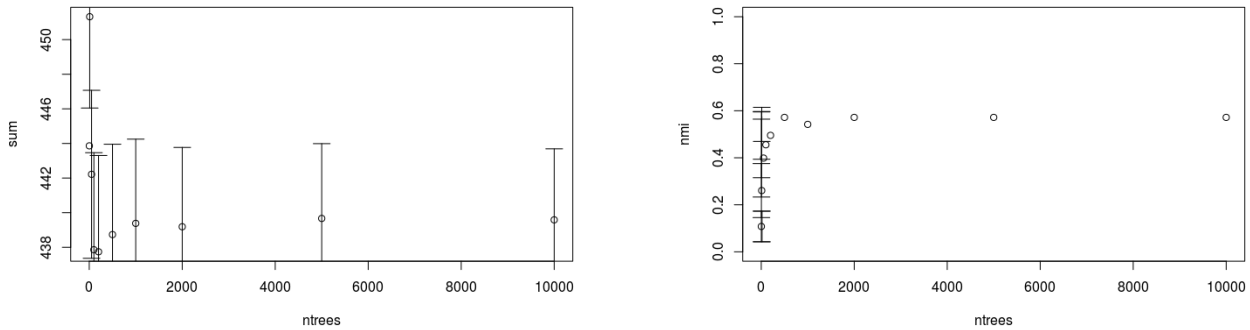
L'hypothèse que nous avions formulée semble être confirmée par l'expérience : lorsque les sommes convergent, les NMI convergent elles aussi. Il est important d'avoir une idée d'une borne basse du nombre d'arbres, afin d'éviter d'augmenter de manière conséquente le temps de calcul sans gain concernant la qualité de la mesure, et *in fine* du clustering. Sur ces exemples, il semble qu'une convergence souvent atteinte à partir de $M = 1000$, même dans les cas de données où le nombre de variables est très supérieur au nombre d'objets, tels que *Lung*.

Au-delà de cette valeur qui ne vaut que pour ces jeux de données bien spécifiques, il semble possible d'obtenir une borne basse pour ce paramètre, en définissant la notion de convergence au niveau de la somme des dissimilarités. Cette variation peut être par exemple fixée à 1% de la valeur précédente. Il s'agit ici d'une valeur arbitraire, qui nous permet d'évaluer une possible convergence sans allonger outre mesure le processus d'optimisation.



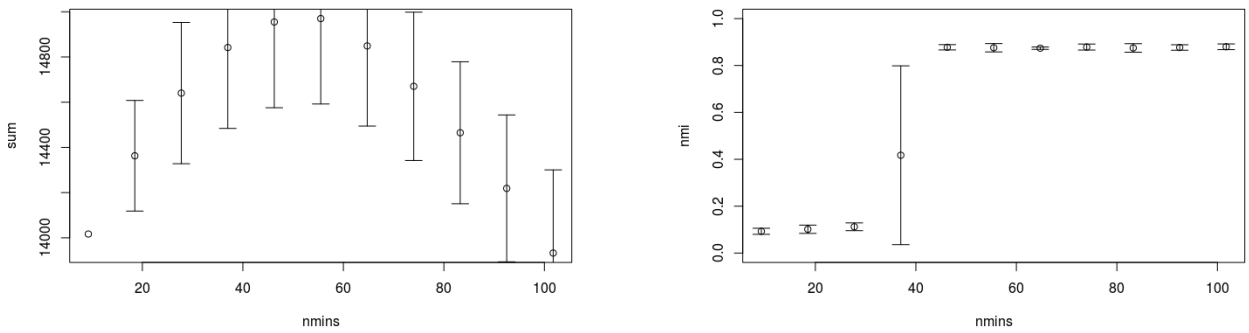
(a) (b)

FIGURE 4.6 – Évolution de DS_Σ (a) et de la NMI (b) en fonction de M (ou n_{trees}) pour le jeu de données Winsconsin



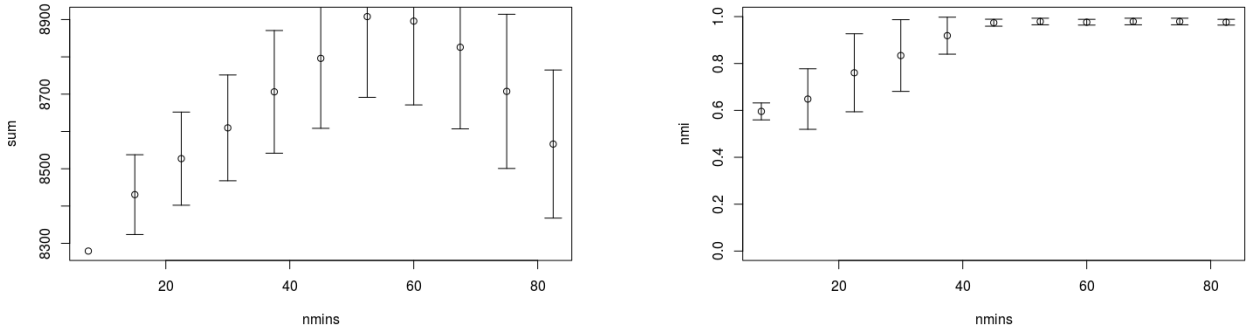
(a) (b)

FIGURE 4.7 – Évolution de DS_Σ (a) et de la NMI (b) en fonction de M (ou n_{trees}) pour le jeu de données Lung



(a) (b)

FIGURE 4.8 – Évolution de DS_Σ (a) et de la NMI (b) en fonction de n_{min} pour le jeu de données Synth1



(a) (b)

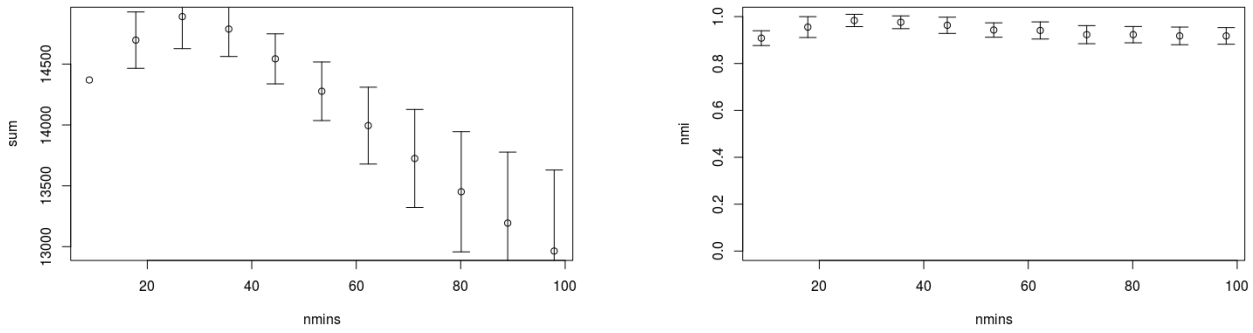
FIGURE 4.9 – Évolution de DS_Σ (a) et de la NMI (b) en fonction de n_{min} pour le jeu de données Iris

Nombre minimal d'instances pour qu'une partition soit réalisée Concernant les valeurs de n_{min}, nous avons réalisé la même expérience, en fixant le nombre d'arbres à 1000. Ici, il est plus difficile d'émettre une hypothèse a priori sur la relation entre la somme des dissimilarités et l'évolution de la qualité de la mesure.

Empiriquement, nous avons constaté un comportement intéressant. En effet, il semble qu'un maximum de DS_Σ en faisant varier n_{min} semble correspondre à une bonne valeur – voire une valeur optimale – de NMI. C'est par exemple le cas sur les jeux de données *Synth1*, *Iris*, *Wine* et *Lung*, dont les résultats sont présentés dans les figures 4.8, 4.9, 4.10 et 4.11.

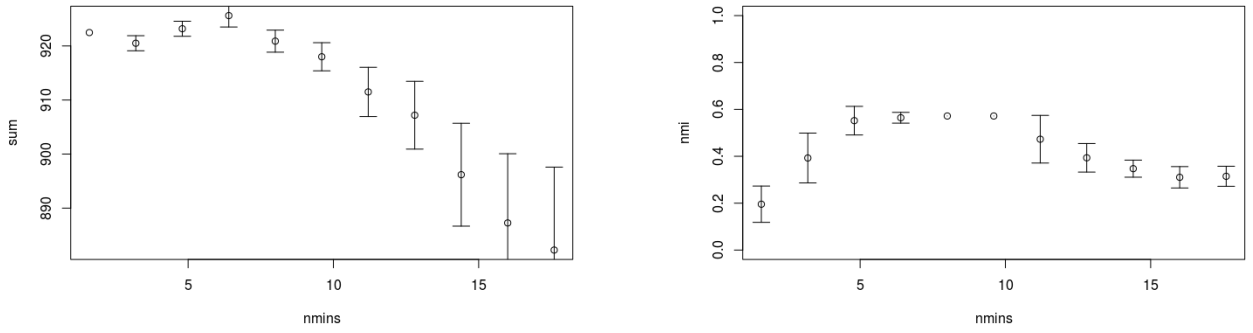
Pour les autres jeux de données présentés dans la table 4.8, cette relation est moins nette, mais semble tout de même être présente, comme on peut le constater dans les figures 4.12, 4.13, 4.14, 4.15 et 4.16.

Une explication possible de la relation entre une valeur maximale de DS_Σ et une valeur optimale de n_{min} peut-être la suivante : lorsque ce paramètre est fixé de manière optimale,



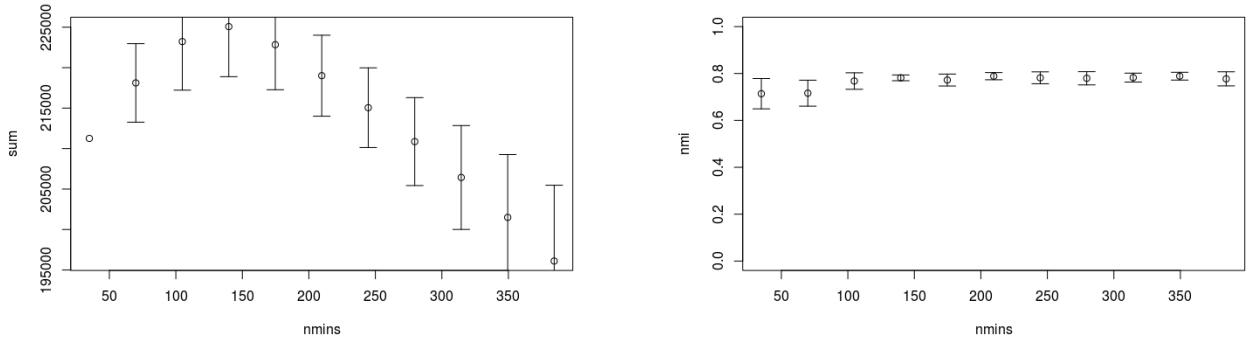
(a) (b)

FIGURE 4.10 – Évolution de DS_Σ (a) et de la NMI (b) en fonction de n_{min} pour le jeu de données Wine



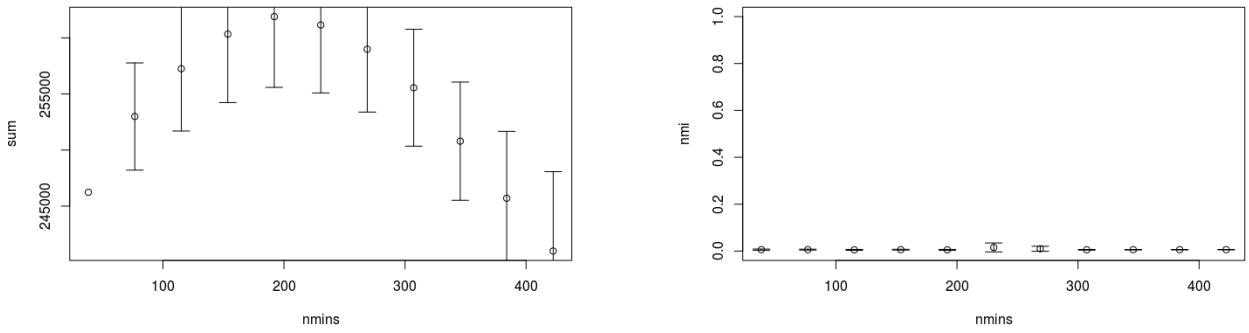
(a) (b)

FIGURE 4.11 – Évolution de DS_Σ (a) et de la NMI (b) en fonction de n_{min} pour le jeu de données Lung



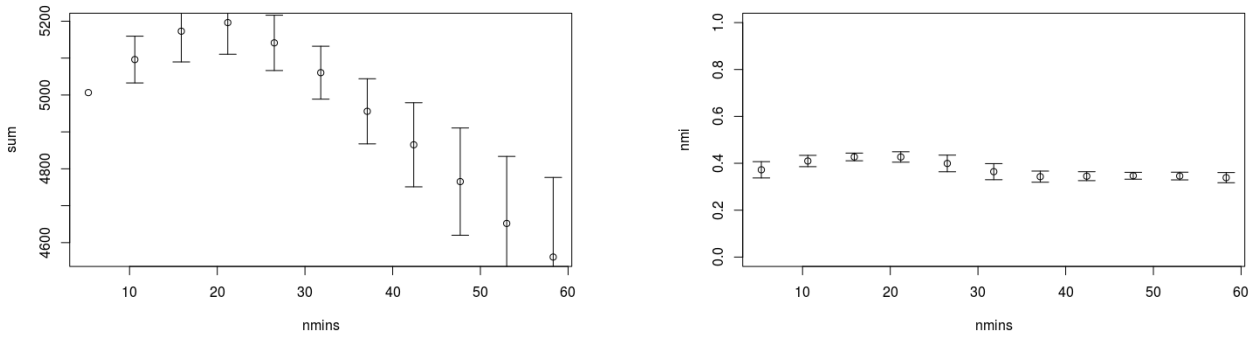
(a) (b)

FIGURE 4.12 – Évolution de DS_Σ (a) et de la NMI (b) en fonction de n_{min} pour le jeu de données Wisconsin



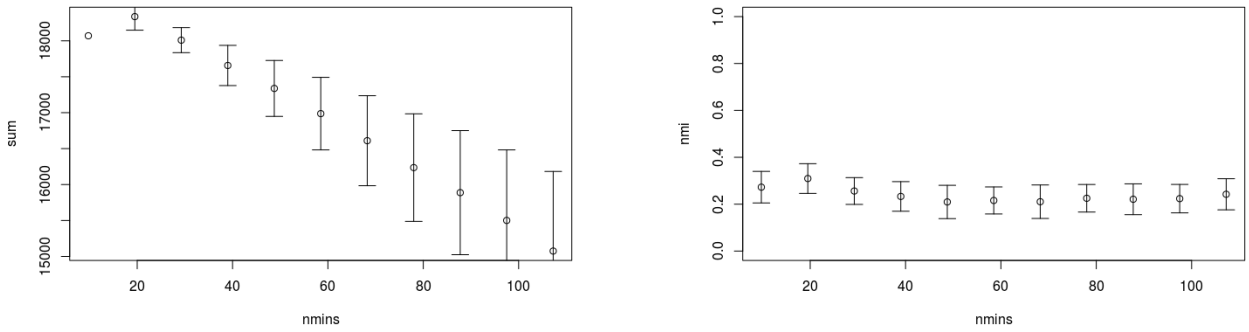
(a) (b)

FIGURE 4.13 – Évolution de DS_Σ (a) et de la NMI (b) en fonction de n_{min} pour le jeu de données Pima



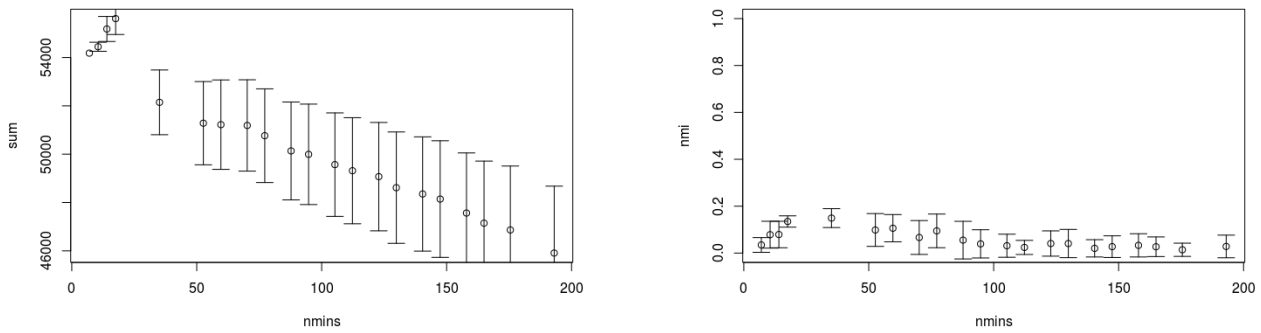
(a) (b)

FIGURE 4.14 – Évolution de DS_Σ (a) et de la NMI (b) en fonction de n_{min} pour le jeu de données Breast



(a) (b)

FIGURE 4.15 – Évolution de DS_Σ (a) et de la NMI (b) en fonction de n_{min} pour le jeu de données Parkinson



(a) (b)

FIGURE 4.16 – Évolution de DS_Σ (a) et de la NMI (b) en fonction de n_{min} pour le jeu de données Ionosphere

le pouvoir discriminant de la forêt d'UET est le plus fort. Ainsi, les distances moyennes entre objets sont plus grandes. Lorsque l'on s'éloigne de cette valeur optimale, la distance moyenne entre objets devient plus faible, rendant la mesure moins discriminante pour la recherche de clusters.

La recherche exhaustive du maximum de la somme peut être en revanche très coûteuse, dans la mesure où il est nécessaire de construire une forêt pour chaque valeur de n_{min}. Pour de gros jeux de données, cette approche n'est pas envisageable. Des méthodes d'optimisation telles que l'optimisation bayésienne [Moč75] peuvent ici être envisagées, afin d'effectuer cette recherche de maximum de manière efficace.

L'optimisation automatique du nombre d'arbres et du *smoothing parameter* peut être intéressante dans les applications. Cependant, cette étude ayant été réalisée a posteriori dans la chronologie de la thèse, les résultats qui suivent n'utilisent pas cette méthode d'optimisation.

Maintenant que nous en savons plus sur l'influence et le choix des paramètres, présentons les évaluations que nous avons réalisées ainsi que leurs résultats.

4.4 Évaluation des forêts d'arbres extrêmement aléatoires non supervisés

Il est nécessaire d'évaluer les performances des forêts d'arbres extrêmement aléatoires non supervisés sur des cas concrets. Cette évaluation peut être réalisée de nombreuses manières différentes. Nous avons vu lors de la description d'UET que cette dernière dépend de certains paramètres.

Nous commencerons par présenter dans la sous-section 4.4.1 la confirmation de quelques caractéristiques théoriques de la méthode. En effet, l'un des intérêts de développer une méthode de calcul de dissimilarités à base d'arbres aléatoires est de capitaliser sur les forces de ces structures. Il convient donc d'évaluer si cela se confirme en pratique.

Nous présenterons ensuite dans les sous-sections 4.4.2, 4.4.3, 4.4.4 et 4.4.4 les résultats obtenus sur des jeux de données synthétiques et réels, dans plusieurs cadres d'évaluation.

Nous utiliserons régulièrement deux mesures permettant de quantifier la qualité d'un clustering que nous avons décrites précédemment, au sein de la section 2.3.7.2, à savoir la *NMI* et l'*ARI*.

4.4.1 Confirmation de quelques caractéristiques théoriques d'UET

Commençons ici par évaluer certaines caractéristiques d'UET. Les expériences présentées ici sont basées sur la comparaison des similarités moyennes intracluster avec les similarités moyennes interclusters, par le biais de leurs différences.

La différence moyenne Δ est calculée de la manière suivante. Dans un premier temps, la moyenne arithmétique des similarités entre paires d'objets ayant la même étiquette est calculée. Il s'agit de la similarité intracluster moyenne μ_{intra}. De la même manière, la moyenne arithmétique des similarités entre paires d'objets n'ayant pas la même étiquette est calculée, et l'on obtient la similarité moyenne intercluster μ_{inter}.

Enfin, la différence $\Delta = |\mu_{intra} - \mu_{inter}|$ est calculée. Dans nos expérimentations, Δ est calculé sur 20 itérations. Dans les sections suivantes, la moyenne des Δ obtenus entre itérations est notée $\bar{\Delta}$, et l'écart type σ.

4.4.1.1 Capacité à discriminer les clusters

La première question que l'on peut se poser concerne les (dis)similarités obtenues, plus précisément, si ces dernières permettent effectivement de discriminer des objets appartenant à différents clusters. En effet, si cela n'est pas le cas, alors la méthode n'a que peu d'intérêt pratique.

Trois jeux de données ont été générés pour cette tâche : deux jeux de données composés de 1000 instances et sans clusters, $NoC4$ et $NoC50$, dont la différence réside dans le nombre d'attributs, respectivement 4 et 50. Les colonnes de ces jeux de données sont générées par échantillonnage d'une distribution normale $\mathcal{N}(0, 1)$. Un troisième jeu de données, $C4$, possède lui des objets divisés en clusters. Composé de 1000 instances et 4 attributs, ses 500 premières lignes sont obtenues par tirage dans une distribution uniforme $\mathcal{U}(0, 0.5)$ pour une colonne et $\mathcal{U}(1, 2)$ pour une deuxième. Les 500 lignes suivantes sont quant à elles obtenues de la même manière, par tirage dans $\mathcal{U}(0.5, 1)$ et $\mathcal{U}(0, 1)$. Deux colonnes sont ajoutées et obtenues par discrétisation des colonnes précédemment générées. Ces colonnes nous permettent de tester notre approche sur des données hétérogènes.

Les valeurs de $\bar{\Delta}$ et σ sont présentées dans la table 4.9. On constate que dans les cas où les données sont aléatoires, et donc a priori sans cluster, les valeurs de $\bar{\Delta}$ sont proches de 0. Cela signifie que les similarités intercluster et intracluster moyennes sont proches, ce qui est attendu. En revanche, dans le cas où les objets peuvent être séparés en clusters, on constate que les valeurs de $\bar{\Delta}$ ne sont pas nulles, et sont au contraire beaucoup plus proches de 1. Cela semble indiquer que les similarités obtenues par UET ont un sens, et permettent effectivement de discriminer des objets appartenant à des groupes différents.

Jeu de données	$\bar{\Delta}$	σ
NoC4	0.00042	0.00003
NoC50	0.00007	0.00003
C4	0.68417	0.00341

TABLE 4.9 – Différences moyennes entre similarités intercluster et intracluster sur les trois jeux de données synthétiques.

4.4.1.2 Robustesse vis-à-vis des transformations monotones des attributs

L'un des avantages de la méthode que nous proposons est son invariance aux transformations monotones d'attributs. En effet, comme précisé par J.H. Friedman dans [Fri06], cette propriété permet une certaine robustesse vis-à-vis de la présence d'*outliers*, ainsi qu'aux changements d'échelles des variables.

Nous vérifions ici ces propriétés sur deux deux jeux de données synthétiques, générés par la les méthodes *make_ blobs* et *make_ moons* de *Scikit-learn* [PVG$^+$11]. Le premier jeu de données, que nous avons nommé *blob500*, est constitué de 500 objets, décrits par 5 attributs et associés à 1 parmi 3 clusters. L'autre jeu de données, *moon500*, contient 500 objets décrits par 2 attributs et dont la particularité est la distribution en clusters non convexes.

La procédure expérimentale consiste à calculer dans un premier temps les valeurs de $\bar{\Delta}$ sur les données initiales, avant de multiplier par ou ajouter un scalaire issu de $\mathcal{U}(2, 100)$ à une ou plusieurs colonnes. $\bar{\Delta}$ est recalculé après chacune de ces transformations.

Les tables 4.10 et 4.11 présentent les variations de $\bar{\Delta}$ et σ en fonction des transformations effectuées sur leurs variables. On peut constater qu'UET est bien robuste aux transformations monotones d'un ou plusieurs des attributs. En effet, les valeurs de $\bar{\Delta}$ sont stables, les variations que l'on peut observer étant liées à la nature stochastique de la méthode. Ces résultats semblent donc confirmer ce qui était attendu, dans la mesure où les processus de partition dans les arbres aléatoires ne sont pas sensibles aux modifications d'un attribut si chaque valeur est *déplacée* de la même manière.

4.4.1.3 Influence des variables corrélées

La présence de variables corrélées peut poser problème pour certaines méthodes [Sam03] en pratique. C'est par exemple le cas pour la distance euclidienne.

Il est donc intéressant d'avoir une mesure de (dis)similarité et/ou un algorithme de clustering peu sensible aux données redondantes, afin d'éviter la tâche chronophage de prétraitement consistant à rechercher ces variables corrélées et à en sélectionner. Nous proposons ici d'évaluer cette propriété pour UET.

Opération	Nombre de variables	$\bar{\Delta}$	σ
Multiplication	0	0.2981	0.0044
Multiplication	1	0.2991	0.0029
Multiplication	2	0.2992	0.0036
Addition	0	0.2987	0.0037
Addition	1	0.2976	0.0045
Addition	2	0.2970	0.0035

TABLE 4.10 – Influence de la multiplication ou de l'addition par un scalaire sur $\bar{\Delta}$ pour le jeu de données *moon500*.

Opération	Nombre of variables	$\bar{\Delta}$	σ
Multiplication	0	0.3283	0.0072
Multiplication	1	0.3297	0.0060
Multiplication	2	0.3285	0.0067
Addition	0	0.3250	0.0053
Addition	1	0.3296	0.0046
Addition	0	0.3267	0.0059

TABLE 4.11 – Influence de la multiplication ou de l'addition par un scalaire sur $\bar{\Delta}$ pour le jeu de données *blob500*

Nous avons utilisé le jeu de données *blob500*, présenté précédemment, avec 10 attributs au lieu de 5. Chaque colonne du jeu de donnée est remplacée de manière itérative par une combinaison linéaire d'une autre colonne, avant de calculer la nouvelle valeur de $\bar{\Delta}$ et σ. La même procédure est réalisée, en remplaçant chaque colonne une à une par des valeurs aléatoires. La figure 4.17 présente l'évolution de $\bar{\Delta}$ lorsque le nombre de colonnes modifiées augmente.

On constate que la présence de variables corrélées n'a que peu d'effet sur $\bar{\Delta}$. Ceci est une propriété très intéressante, réduisant le nombre de tâches de prétraitement à effectuer.

4.4.1.4 Résistance au bruit dans les données

L'exploration de données peut être complexifiée par la présence de bruit. Il est donc important d'avoir une méthode relativement résistante au bruit, dans la mesure où ce dernier est souvent introduit ou présent dans les données. Nous avons évalué le comportement d'UET vis-à-vis du bruit sur trois jeux de données synthétiques (*moon500* et *blob500*) et réels (*Iris*).

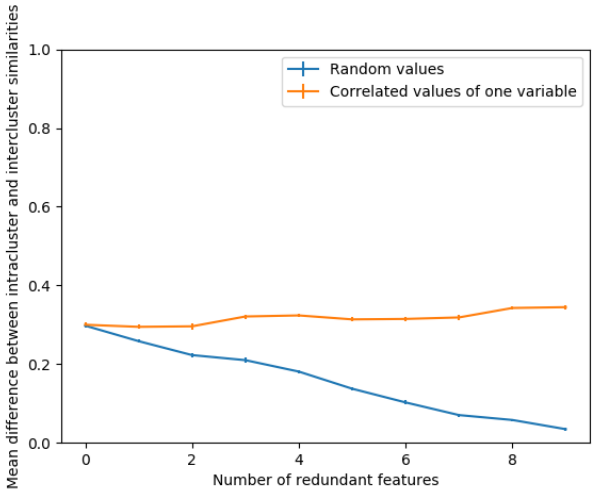


FIGURE 4.17 – Évolution de la différence entre la moyenne des similarités intracluster et la moyenne des similarités intercluster lorsque (i) les attributs sont remplacés par une combinaison linéaire d'autres attributs et lorsque (ii) les attributs sont remplacés par des valeurs aléatoires . L'axe des abscisses représente le nombre d'attributs modifiés par cette procédure.

Dans un premier temps, 25 jeux de données *moon500* sont générés. Chaque jeu de données est généré en utilisant la même graine (*seed*) [24]. Nous avons ensuite ajouté un bruit gaussien, en modifiant son écart type de 0 – *i.e.* sans bruit – à 0,48. Cette tâche est rendue aisée par la relative simplicité de l'implémentation dans *Scikit-learn*, utilisé pour la génération de ce jeu de donnée. La figure 4.18 présente quelques exemples de projections de ces données lorsque le bruit augmente.

Concernant *blob500* et *Iris*, contrairement à *moon500*, aucune méthode de génération de bruit dédiée n'existe. Nous avons donc ajouté ce bruit manuellement, en incrémentant chaque valeur x par une valeur tirée de $\mathcal{U}(-st \times x, st \times x)$, st représentant un facteur de *force* du bruit.

Comme dans les évaluations précédentes, nous avons calculé les valeurs de $\bar{\Delta}$ pour chaque incrément de bruit. Les valeurs obtenues sont présentéesà la figure 4.19. Bien que l'on observe une diminution attendue de $\bar{\Delta}$, on constate que cette dernière a une pente qui reste acceptable. De plus, une observation est intéressante dans le cas de *moon500*. En effet, alors que pour des valeurs de bruit supérieures à $0,20$ l'on a constaté un début de fusion des clusters, on remarque qu'UET reste capable de discerner les deux clusters en termes de $\bar{\Delta}$.

Les caractéristiques que nous avons évaluées dans cette section sont intéressantes dans la mesure où elles permettent de réduire le nombre de tâches de prétraitement à effectuer dans le cadre d'un clustering. Dans [HTF09], Hastie *et al.* décrivent une méthode *off-the-shelf* comme une méthode pouvant être directement appliquée aux données sans avoir à effectuer de nombreuses tâches coûteuses de prétraitement ou réglage de précision (*fine-tuning*) des paramètres de la

24. Cette graine nous permet d'obtenir à chaque lancement le même jeu de données, utile ici pour des raisons de reproductibilité des expériences.

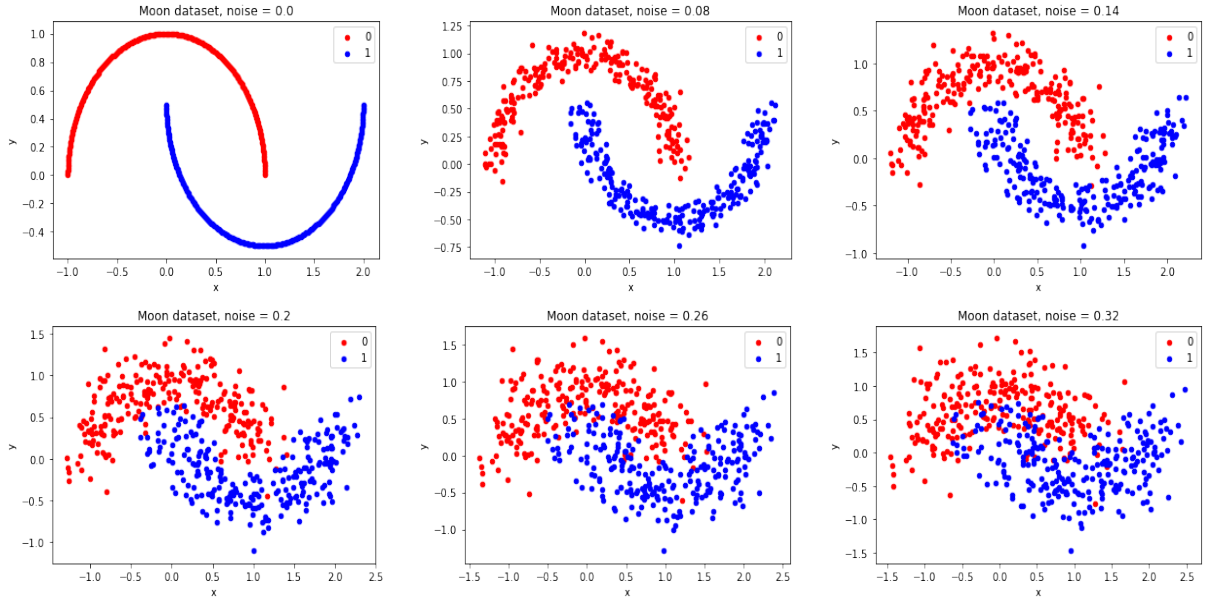


FIGURE 4.18 – Évolution pour 6 jeux de données *moon500* lorsque le bruit augmente. On constate que lorsque le bruit est supérieur à $0,20$ les limites entre clusters fusionnent et deviennent difficiles à discerner.

procédure d'apprentissage. Nous pensons que l'approche que nous proposons constitue une bonne méthode *off-the-shelf* à des fins de clustering.

4.4.2 Évaluation des performances d'UET sur des données numériques, qualitatives et hétérogènes

L'une des forces d'UET est sa versatilité, dans la mesure où elle peut être appliquée à différents types de variables. Évaluons ici les performances réelles de l'approche dans différents cadres : numérique, qualitatif et hétérogène.

4.4.2.1 Données numériques

Commençons par évaluer UET dans le cas le plus simple et courant, à savoir celui des données homogènes numériques. Ici, l'ensemble des attributs décrivant les objets sont des nombres réels. Les jeux de données utilisés ici sont au nombre de cinq. Trois d'entre eux sont synthétiques ($S1_n$, $S2_n$ et $S3_n$), et deux d'entre eux sont issus d'expérimentations réelles. Ces jeux de données sont décrits dans la table 4.12.

$S1_n$, $S2_n$ et $S3_n$ sont obtenus par tirage pour chaque colonne et chaque cluster dans une distribution différente. Ainsi, pour un attribut donné, les variables décrivant les objets d'un même cluster sont issues de la même distribution. *Iris* et *Wisconsin* sont extraites de l'UCI [25].

25. `https://archive.ics.uci.edu/ml/index.php`.

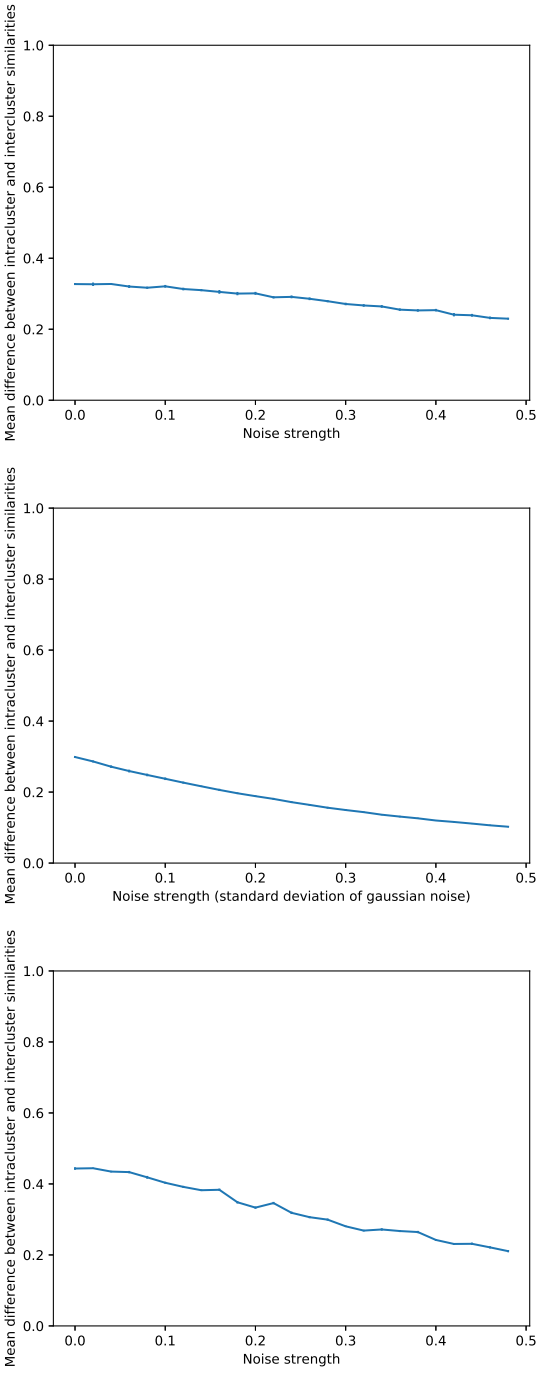


FIGURE 4.19 – Évolution de $\bar{\Delta}$ lorsque le bruit augmente dans trois jeux de données (de haut en bas : *blob500*, *moon500* and *Iris*.)

Nom	# instances	# variables	# Classes	$\bar{\Delta}$	σ
$S1_n$	500	2	2	0.4685	0.0015
$S2_n$	450	2	3	0,5290	0,0007
$S3_n$	500	4	2	0,4873	0,0082
Iris	150	4	3	0.4312	0.0056
Wisconsin	699	9	2	0.2259	0.0048

TABLE 4.12 – Différences moyennes en similarités intracluster et intercluster obtenues par UET, notées $\bar{\Delta}$. σ représente l'écart type.

Nom	# instances	# variables	# Classes	$\bar{\Delta}$	σ
$S1_c$	500	2	2	0.3878	0.0056
$S2_c$	500	4	2	0,3387	0,006
Soybean	266	35	19	0.2370	0.0062

TABLE 4.13 – Différences moyennes en similarités intracluster et intercluster obtenues par UET, notées $\bar{\Delta}$. σ représente l'écart type.

Les résultats, présentés dans la même table, indiquent que la méthode donne effectivement une différence significative entre les similarités intercluster et les similarités intracluster. Cela semble confirmer les observations préliminaires de la section 4.4.1, montrant qu'UET est capable de discriminer – en termes de similarités – des objets appartenant à des groupes différents lorsque leurs attributs sont continus.

4.4.2.2 Données qualitatives

Nous évaluons ici le comportement d'UET sur des données purement qualitatives.Trois jeux de données sont utilisés à cet effet : $S1_c$, $S2_c$ et *Soybean*. Comme précédemment, $S1_c$ et $S2_c$ sont obtenus par tirage, cette fois-ci dans différentes distributions multinomiales, avec trois modalités. *Soybean* est quant à lui extrait de la base de l'université d'Irvine, comme *Iris* et *Wisconsin*.

Les jeux de données sont décrits dans la table 4.13. Cette table reporte aussi les résultats obtenus. Dans le cadre qualitatif, la même constatation que dans le cadre numérique est faite : la méthode semble pouvoir discriminer les objets appartement à différents clusters, les $\bar{\Delta}$ étant significativement plus grand que 0.

4.4.2.3 Données hétérogènes

Enfin, passons au cadre hétérogène. Ce dernier étant l'un des cas ayant motivé le développement d'UET, il est intéressant de voir comment la méthode se comporte dans ce cas.

Les jeux de données utilisés pour cette évaluation sont présentés dans la table 4.14. Trois jeux de données synthétiques sont générés par une approche similaire aux cas précédents. Deux jeux

Nom	# Instances	# Continues	#qualitatives	# Classes	$\bar{\Delta}$	σ
$S1_h$	500	2	2	2	0.3009	0.0061
$S2_h$	500	2	4	2	0,321 0	0,0058
$S3_h$	500	4	4	2	0,355 7	0,0063
Credit	690	7	6	2	0.0648	0.0026
CMC	1473	5	4	3	0.0118	0.0003

TABLE 4.14 – Différences moyennes des similarités intracluster et intercluster obtenues par UET, notées $\bar{\Delta}$. σ représente l'écart type.

de données *réels*, *Credit* et *CMC*, ont quant à eux été extraits du dépôt de l'UCI [26].

Les résultats présentés dans la table 4.14 nous montrent que dans l'ensemble des jeux de données synthétiques, les performances d'UET sont bonnes, les mêmes constatations que celles effectuées dans les deux cadres homogènes pouvant être faites sur les valeurs de $\bar{\Delta}$. En revanche, la même chose ne peut pas être dite concernant les jeux de données réels. En effet, les valeurs de $\bar{\Delta}$ sont ici proches de zéro, signifiant qu'aucune différence n'existe entre les similarités intercluster et intracluster.

Cette observation peut être liée au fait qu'il n'existe aucune structure de clusters de ces jeux de données. En effet, les deux jeux de données utilisés ici sont dédiés aux tâches de classification supervisée. Les étiquettes peuvent donc ne pas correspondre à un clustering des objets obtenu par le biais d'un algorithme dédié.

4.4.3 Comparaison des résultats d'UET avec des résultats de la littérature

Dans les sections précédentes, nous nous sommes concentrés sur une évaluation empirique des différences entre similarités intracluster et intercluster. Bien que cette approche soit intéressante dans la mesure où elle réduit le biais pouvant être apporté par le choix d'un algorithme de clustering spécifique, il est intéressant de comparer les résultats que nous pouvons obtenir en effectuant un clustering sur la base des distances obtenues par UET avec des résultats de clusterings de la littérature.

Le protocole est le suivant. Pour chaque jeu de données, *UET* est lancé dix fois, et les matrices de similarité obtenues sont agrégées, par le biais de la moyenne arithmétique. La matrice ainsi obtenue est transformée en matrice de distance en utilisant la relation $D = \sqrt{1 - S}$, et un clustering agglomératif hiérarchique est effectué avec le nombre de clusters connu fixé. La qualité du clustering est évaluée par le biais de la NMI.

Le clustering est effectué vingt fois, et nous reportons les NMI moyennes ainsi que leurs écarts-types. Cette évaluation a été effectuée en utilisant la bibliothèque *Scikit-learn* et notre implémentation d'UET.

26. `https://archive.ics.uci.edu/ml/datasets.php`

Jeu de données	#Instances	#Variables	#Classes
Iris	150	4	3
Wine	178	13	3
Wisconsin	699	9	2
Lung	32	56	3
Breast tissue	106	9	6
Isolet	1559	617	26
Pima	768	8	2
Parkinson	195	22	2
Ionosphere	351	34	2
Segmentation	2310	19	7

TABLE 4.15 – Jeux de données utilisés pour le benchmarking.

Jeu de donnée	UET - NMI	Littérature - NMI
Wisconsin	**78.33** ± 3,25	73,61 ± 0,00
Lung	**29.98** ± 6.17	22.51 ± 5.58
Breast tissue	**74.48** ± 2.92	51.18 ± 1.38
Isolet	61.22 ± 1,47	**69,83** ± 1,74
Parkinson	**25.50** ± 6,14	**23,35** ± 0,19
Ionosphere	**13.47** ± 1.11	**12.62** ± 2.37
Segmentation	**69,62** ± 2,14	60,73 ± 1,71

TABLE 4.16 – Évaluation comparative avec les résultats de [EA10]. Les meilleurs résultats sont indiqués en gras. En cas d'égalité, les deux valeurs sont indiquées en gras.

Les dix jeux de données utilisés dans cette section sont présentés dans la table 4.15, et sont tous issus du dépôt de l'UCI.

Nous allons nous comparer ici aux résultats de [EA10]. Dans leur travail, les auteurs présentent les NMI moyennes obtenues par application de l'algorithme des k-moyennes sur vingt itérations. Leur méthode étant une méthode de sélection de variables, une comparaison juste n'est pas triviale. Nous constatons cependant dans les résultats présentés dans la table 4.16 que les NMI obtenues par UET sont compétitives dans la plupart des cas. Une constatation notable est le fait que dans certains cas, les résultats obtenus par UET – sans sélection de variables donc – sont meilleurs que les résultats obtenus dans [EA10], où une sélection est faite. C'est, par exemple, le cas pour le jeu de donnée *Breast Tissue*.

Jeu de donnée	UET (ARI - Temps)	URF (ARI - Temps)
Wisconsin	**87.13 - 22,00 s**	82,92 - 968,71 s
Lung	**23.24 - 0,66 s**	6,52 - 86,93 s
Breast tissue	**58.85 - 1,93 s**	39.05 - 99.40 s
Isolet	**28,04 - 119.29 s**	-
Parkinson	**25,21 - 3.79 s**	12.68 - 279,30 s
Ionosphere	**7,93 - 7.91 s**	**7.28** - 727,30 s

TABLE 4.17 – Évaluation comparative entre URF et UET. Les meilleurs résultats sont indiqués en gras. En cas d'égalité, les deux valeurs sont indiquées en gras.

4.4.4 Comparaison des dissimilarités obtenues via des UET avec d'autres dissimilarités

Ici, nous nous proposons de comparer les dissimilarités obtenues avec notre approche avec, d'une part, celles obtenues par URF, et d'autre part, avec la distance euclidienne. Commençons par considérer URF, la méthode à l'origine de son développement. Afin d'effectuer cette comparaison, nous avons utilisé l'implémentation R fournie par les auteurs [27]. La mesure de qualité de clustering utilisée est l'ARI, obtenue après l'application de la méthode de partitionnement autour de représentants (*Partitioning Around Medoids*, *PAM*) sur les matrices de distances obtenues par chacune des méthodes.

Pour URF, cent forêts de 2000 arbres sont construites, avec une valeur de $m_{try} = \lfloor \sqrt{n_{attributs}} \rfloor$ [28]. Concernant UET, une forêt de 200 arbres est construite, avec $n_{min} = \lfloor \frac{N}{3} \rfloor$. Le processus est répété vingt fois, et les matrices de similarités S sont agrégées par le biais de la moyenne arithmétique. Ces expériences sont réalisées sur une machine équipée d'un processeur Intel i7-6600U (2,6 GHz) et 16 Go de DDR4 RAM (2133 MHz).

Nous avons comparé les ARI obtenues, ainsi que les temps de calcul (en secondes) pour chaque méthode. Les résultats sont présentés dans la table 4.17. On constate qu'en termes de temps, les performances d'UET sont bien supérieures à celles d'URF, tout en donnant des clusterings de qualité similaire ou meilleure que cette dernière. Concernant le jeu de donnée *Isolet*, nous avons dû stopper le calcul avec les URF, dans la mesure où ce dernier ne se terminait pas en un temps raisonnable sur notre machine. Nous avons cependant effectué ce calcul sur une autre machine, pour arriver à une ARI de $28,39$.

Ces résultats, notamment concernant les bonnes performances en termes de temps de calcul, étaient attendus et sont ici confirmés. En effet, UET évitant la phase coûteuse de génération d'instances, sa complexité pratique est bien inférieure à celles des URF.

Il est enfin intéressant de comparer les dissimilarités pouvant être obtenues avec UET avec les distances euclidiennes sur les mêmes données. Pour chaque jeu de données, nous comparons les

27. `https://labs.genetics.ucla.edu/horvath/RFclustering/RFclustering.htm`.

28. m_{try} correspond au nombre de variables sélectionnées à chaque noeud pour effectuer une partition.

Jeu de donnée	UET - NMI	Euclidienne - NMI
Wisconsin	**79.32** ± 2,12	66,03 ± 0,00
Lung	**28.64** ± 4,90	**28,20** ± 0,00
Iris	**98.21** ± 1,5	80,58 ± 0,00
Wine	**95.01** ± 4.74	41.58 ± 0.00
Parkinson	**30.37** ± 3,90	0,00 ± 0,19
Soybean	**85.02** ± 1.58	71.86 ± 0.00
Pima	**2,80** ± 0,90	0,00 ± 0,00

TABLE 4.18 – Évaluation comparative entre UET et distance euclidienne. Les meilleurs résultats sont indiqués en gras. En cas d'égalité, les deux valeurs sont indiquées en gras.

clusterings obtenus par clustering hiérarchique agglomératif sur la base de deux dissimilarités : euclidienne, et UET. La qualité de chaque clustering est quantifiée par la NMI.

Pour des jeux de données numériques, nous appliquons la distance euclidienne sans transformations préalables. En revanche, pour ce qui est des jeux de données qualitatifs, nous commençons par effectuer un *One-Hot encoding*. Cet encodage permet de transformer chacune des variables en autant de variables booléennes que de modalités. Chacune de ces nouvelles variables ne prend que deux valeurs, à savoir 0 dans le cas où cette modalité n'est pas celle présente dans la description d'une instance donnée, ou 1 dans le cas contraire. Il s'agit donc d'une *binarisation* des attributs. Cela permet d'éviter la considération de notion d'ordre entre les valeurs, au prix d'une explosion potentielle de la dimensionnalité. L'implémentation de *Scikit-learn* est utilisée afin de réaliser ce prétraitement.

Les résultats sont présentés dans la table 4.18. Les clusterings obtenus en utilisant les dissimilarités issues d'UET sont de meilleure qualité que ceux utilisant la distance euclidienne ($p < 0.01$). La distance euclidienne a cependant comme avantage son caractère déterministe. Ainsi, il n'est pas nécessaire de répéter une expérience dans la mesure où la distance euclidienne restera fixe d'une itération à l'autre. Ceci explique par ailleurs l'écart type de 0 dans ce cas.

4.4.5 Application à des données cliniques

Au delà des *benchmarks* et études présentés précédemment, notre participation à une étude dans le cadre du projet FIGHT-HF a été l'occasion d'appliquer la méthode que nous proposons sur des données cliniques. Ce travail a donné lieu à une publication récemment acceptée dans *Scientific reports* [PDD+21]. Les résultats présentés dans cet article dépassent le périmètre d'UET. Il s'agit en effet d'évaluer différentes méthodes de clustering sur des données hétérogènes, afin de guider les cliniciens sur le choix de la méthode à choisir parmi le catalogue de méthodes disponibles, pouvant être déroutant au premier abord. UET a donc tout à fait sa place dans les méthodes testées ici.

Parmi les méthodes de clustering considérées, trois font partie de la catégorie qui nous intéresse dans le cadre de cette thèse, à savoir les méthodes à base de (dis)similarités. Quatre autres

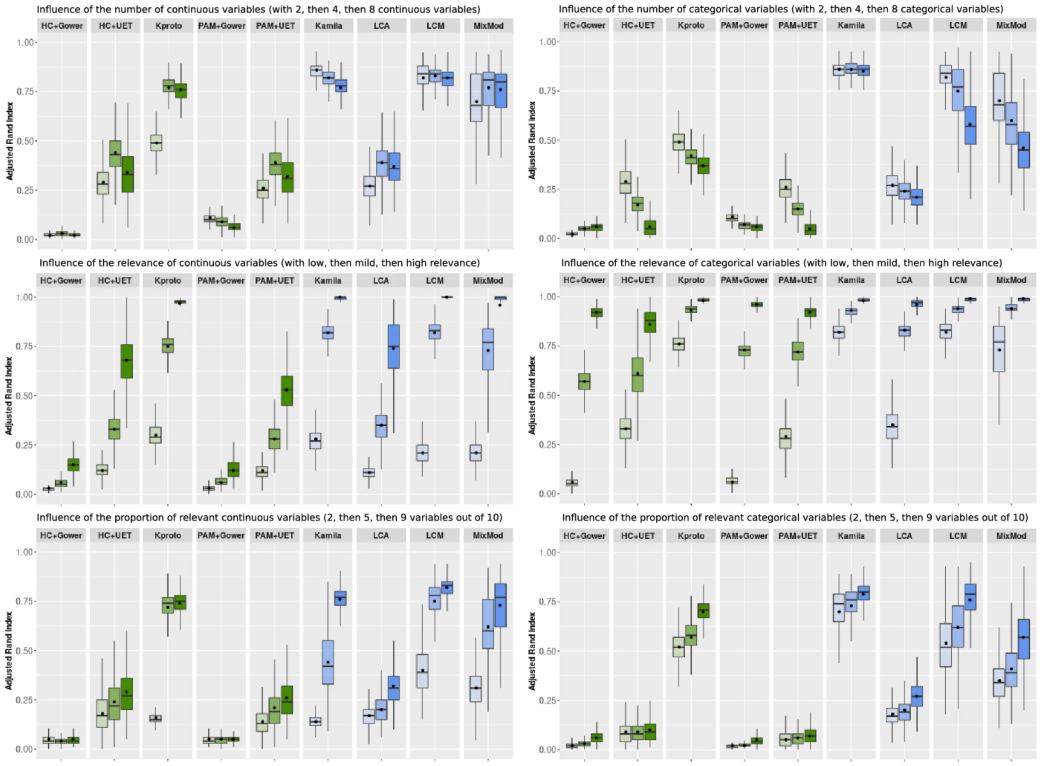


FIGURE 4.20 – Résultats obtenus après clustering *via* différentes approches de clustering sur des données hétérogènes. Ces données sont générées à façon en fonction des besoins, et varient selon un paramètre à chaque fois : le nombre de variables, la proportion de variables bruitées, et la force du bruit [PDD+21] [29].

sont d'un type que nous n'avons pas considéré ici, à savoir basées sur les modèles (*model-based clustering*). Les trois méthodes basées sur les dissimilarités sont UET, Gower, ainsi que la méthode des k-prototypes. Dans l'article, le choix a été fait de distinguer cinq méthodes. En effet, Gower comme UET ne sont pas des méthodes de clustering *per se*. Elles sont donc associées à deux méthodes de clustering à base de distance, la méthode des k-medoides et une approche de clustering hiérarchique.

Dans un premier temps, cette étude consiste à évaluer toutes ces méthodes sur des données synthétiques, obtenues en faisant varier successivement plusieurs paramètres : le nombre de variables continues ou qualitatives, l'information qu'elles contiennent, la proportion de variables bruitées, ou encore la taille de la population. La figure 4.20 présente quelques résultats intéressants obtenus suite à cette évaluation sur des données synthétiques.

Nous nous concentrerons sur les méthodes basées sur la notion de dissimilarité, dans la mesure où il s'agit du coeur du travail présenté dans cette thèse. Parmi ces méthodes, on constate tout d'abord qu'UET, sous ces deux variations, donne des résultats bien supérieurs à ceux obtenus par la distance de Gower, pourtant relativement courante. Ces résultats sont encourageants pour UET. On constate cependant des résultats moins bons face à la méthode des k-prototypes, cette

dernière donnant systématiquement de meilleures ARI que les autres approches basées sur la dissimilarité. On peut voir qu'UET n'est pas la panacée et ne donne pas systématiquement les meilleurs résultats, ce qui est cohérent avec le théorème avec le théorème *no free lunch* mentionné dans le premier chapitre. La méthode reste cependant supérieure à la distance de Gower, l'une des plus connues.

L'un des points intéressants de ce travail est l'application d'UET sur des données cliniques. En effet, nous n'avons jusqu'à présent vu comment la méthode se comportait que dans des cas d'école, sur des données de *benchmark*, simulées ou non. La réalité est bien sûr souvent plus complexe que ces jeux de données, et il est intéressant de voir les résultats d'UET sur ce type de données.

Le jeu de données cliniques utilisé ici est issu de l'étude EPHESUS [PBR+08], une étude clinique en double aveugle contre placebo, conduite sur 6632 patients ayant été récemment victimes d'un infarctus du myocarde aigü, avec une fraction d'éjection ventriculaire inférieure à 40%. Ces patients ont par ailleurs des comorbidités telles que de l'insuffisance cardiaque ou du diabète. Dans cette étude, une mortalité ainsi qu'un taux d'hospitalisation significativement plus faible ont été observés chez les patients traités par la substance active, l'éplérénone. Les données recueillies au cours de cette étude clinique sont utilisées à des fins de clustering, afin d'éventuellement mettre en évidence des groupes de patients, sur la base de seize variables hétérogènes. Six d'entre elles sont continues (âge, indice de masse corporelle, fraction d'éjection ventriculaire gauche, taux de filtration glomérulaire estimé, taux de potassium et taux de sodium), et dix variables qualitatives binaires, parmi lesquelles on retrouve le genre, ou encore la présence ou absence de divers facteurs de risque (hypertension, antécédent de crise cardiaque, angine de poitrine, etc.).

L'objectif est ici d'analyser les patients présents dans chaque cluster sous trois angles différents : (i) est-ce que la survie est statistiquement différente entre patients de différents groupes, (ii) est-ce que l'efficacité du traitement diffère de manière significative entre les groupes, et enfin (iii) quelles sont les variables clés permettant de distinguer les groupes de patients retrouvés ? Afin de répondre à ces questions, les courbes de Kaplan-Meier ainsi que des modèles de Cox ont été utilisés.

Pour les besoins de l'étude, le nombre de clusters est fixé à quatre. La figure 4.21 résume les résultats obtenus avec les cinq méthodes sélectionnées. Les effets moyens du traitement pour chaque cluster, ainsi que leur intervalles de confiance à 95% sont représentés par des *graphiques en forêt*. Des graphiques en radar, ainsi que le calcul des différences moyennes standardisées (ASMD) sont quant à eux utilisés afin de représenter l'influence de chaque variable dans les clusters obtenus.

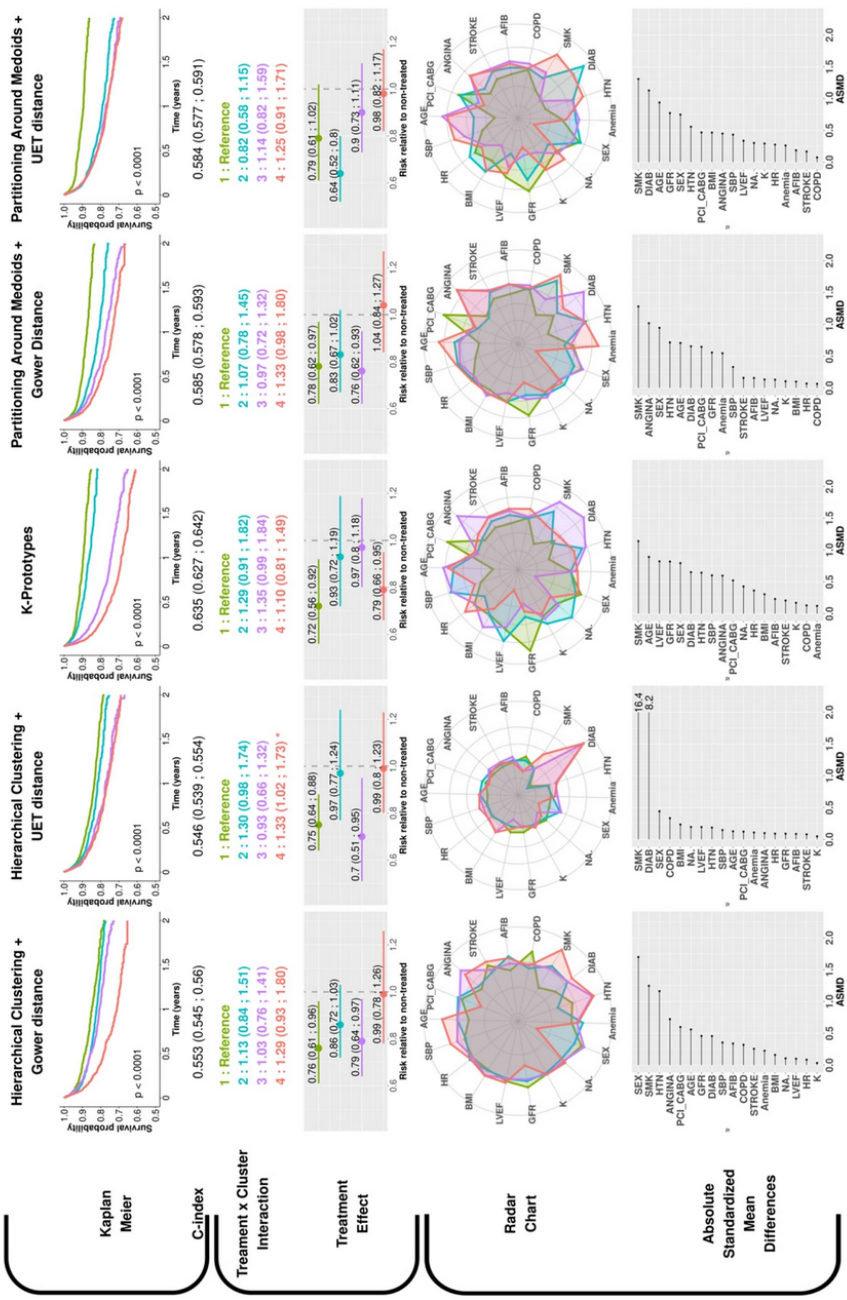


FIGURE 4.21 – Résultats obtenus après clustering *via* les méthodes basées sur les dissimilarités sur le jeu de données clinique EPHESUS. On constate des différences majeures entre méthodes, tant au niveau pronostic, prédictif qu'en termes de variables descriptives [PDD+21].

On constate ici plusieurs choses. Tout d'abord, quelle que soit la méthode, le premier groupe (en vert) présente le plus faible risque d'hospitalisation ou de décès lié à une insuffisance cardiaque, alors que le quatrième groupe (en rouge) présente le risque le plus élevé. De plus, là encore pour l'ensemble des méthodes, le genre et le statut tabagique sont retrouvés dans les 5 plus grandes *ASMD*, ce qui signifie que ces critères sont importants dans l'ensemble des clusterings obtenus. Deux méthodes donnent cependant des graphiques en radar bien distinct : les k-prototypes et notre approche Il est intéressant de noter que les groupes obtenus par UET suivi du clustering hiérarchique se dénotent par deux variables qui sont des facteurs de risque bien connus : le statut tabagique et l'antécédent de diabète. Les courbes de Kaplan-Meier montrent par ailleurs que les groupes ayant le plus grands nombres de patients avec ces caractéristiques sont les groupes de patients les plus à risque. Cette étude pratique montre qu'UET peut avoir un intérêt applicatif, tant de par sa compatibilité avec les données hétérogènes que par la mise en évidence de clusters ayant un intérêt clinique.

4.5 Bilan du chapitre

Dans cette première contribution, nous avons présenté une approche de calcul de similarités basée sur des arbres extrêmement aléatoires. Ces similarités peuvent aisément être transformées en dissimilarités, et être utilisées à des fins de clustering. La méthode proposée peut cependant être utilisée dans d'autres cadres où des (dis)similarités peuvent être nécessaires telles que les algorithmes de prédiction à base d'instances (KNN), les moteurs de recherche, les systèmes de recommandation. . ..

Notre approche est basée sur les forêts d'arbres aléatoires non supervisées et étend cette approche par l'utilisation d'arbres extrêmement aléatoires. En paramétrant ces arbres pour qu'un seul attribut soit sélectionné à chaque étape de partition, des arbres totalement aléatoires sont construits. L'intérêt de ces arbres totalement aléatoires est que, contrairement aux URF, aucune phase de génération d'instances synthétiques n'est nécessaire. Cela est très bénéfique en termes de complexité. Nous avons montré notamment montré que notre approche est de 40 à 100 fois plus rapide que les URF.

Notre problème initial concernait le cas des données hétérogènes. UET permet de résoudre ce problème, les arbres pouvant être induits tant sur des données homogènes (continues ou qualitatives) que sur des données hétérogènes. UET présente par ailleurs quelques propriétés intéressantes pouvant diminuer de manière significative la complexité d'une tâche d'exploration et de préparation de données. Par exemple, l'invariance vis-à-vis des transformations monotones d'attributs évite à l'utilisateur d'effectuer une tâche de mise à l'échelle (*scaling*) des données, nécessaire par exemple si ce dernier utilisait la distance euclidienne. De plus, en pratique, des questions peuvent se poser concernant les attributs fortement corrélés. La robustesse de notre approche dans le cas où des variables sont redondantes peut constituer une solution dans ces cas.

Nos évaluations empiriques montrent que les (dis)similarités calculées en utilisant UET sont principalement influencées par un paramètre, n_{min}. D'autres paramètres, tels que le nombre d'arbres par forêt, ne semblent pas influencer de manière conséquente la qualité des dissimilarités, pour des valeurs de $M > 50$. Augmenter le nombre d'arbres entraîne une augmentation du temps de calcul, il est donc nécessaire de contenir ce paramètre à son minimum.

Nous avons par la suite approfondi cette notion de paramètres, en travaillant sur une méthode d'optimisation automatique. Il s'avère que la valeur précédemment fixée de $M = 50$, bien que donnant de manière générale de bons résultats sur les données utilisées ici, n'est pas la valeur optimale dans tous les cas. Cela était attendu, dans la mesure où une valeur aussi faible ne peut permettre de donner des dissimilarités assez fidèles et stables pour de grands jeux de données. Nous avons montré qu'une mesure, la somme des similarités, peut permettre de guider le choix du nombre d'arbres. En effet, cette dernière converge lorsque le nombre d'arbres optimal est atteint. Nous avons aussi constaté que cette mesure peut être utilisée afin d'optimiser le paramètre n_{min}. La sélection d'une bonne valeur pour ce paramètre est d'autant plus importante que les résultats subséquents en dépendent fortement.

Nous avons analysé les clusters obtenus avec les mesures obtenues par UET avec (i) des résultats de la littérature et (ii) les résultats d'URF, sur plusieurs jeux de données. La qualité des clusterings, effectués par un clustering hiérarchique agglomératif, est mesurée soit par la NMI, soit par l'ARI. Ce paramètre est choisi en fonction de la disponibilité des résultats dans la littérature considérée. Cette évaluation empirique nous a donné des résultats prometteurs. Nous avons prolongé ces évaluations par une étude comparative sur de nombreux jeux de données synthétiques, ainsi qu'un jeu de données cliniques.

Maintenant que nous avons une méthode fonctionnant sur des données présentées sous forme tabulaire, on peut se demander s'il est possible d'utiliser le même type d'approche sur des données modélisées sous d'autres formes. Plus précisément, nous avons mentionné précédemment le fait que dans le cadre de notre projet, nous devons travailler sur des données complexes, pouvant être sous forme de graphes – potentiellement attribués –. Dans le chapitre suivant, nous montrerons qu'il est en fait possible d'étendre ce type d'approche aux graphes.

5

Forêts d'arbres extrêmement aléatoires pour le calcul de similarités dans les graphes

Sommaire

Comme nous avons vu le voir au cours du chapitre 3, les graphes sont des structures permettant de représenter de manière expressive des relations entre objets, ainsi que les propriétés de ces derniers, dans le cas des graphes attribués. Le clustering de graphes – tout du moins, le *within graph clustering* que nous considérons ici – consiste à rechercher des groupes de noeuds ayant plus d'arêtes entre eux, relativement à d'autres noeuds. On peut la retrouver dans deux nombreux domaines, tels que les réseaux informatiques ou la biologie. Il s'agit d'une tâche complexe, et de nombreuses approches ont été proposées, dont nous avons donné un aperçu dans le chapitre 3.

Nous avons vu leurs limitations, notamment dans le cadre des graphes attribués, où les méthodes impliquent parfois des modifications coûteuses en ressources des graphes d'origine, telles que la transformation des attributs en noeuds, pouvant entraîner une augmentation notable de la taille du graphe. Nous avons mentionné au cours du chapitre 4 le fait que l'une des forces d'UET est sa modularité, dans la mesure où il est envisageable d'appliquer l'approche sur différentes modélisations de données, pour peu que le processus de construction soit défini de manière cohérente. Nous avons vu jusqu'ici leur application à des données structurées, mais la méthode pourrait éventuellement être applicable à d'autres représentations des données. Nous nous sommes donc naturellement intéressés à son extension aux cas des graphes, nous offrant un cadre unifié de calcul de (dis)similarités sur des données structurées et sur des graphes, et montrerons dans ce chapitre que l'approche présentée au chapitre précédent est effectivement versatile.

L'intérêt de l'extension aux graphes potentiellement attribués est motivée par notre implication dans le projet FIGHT-HF. De plus, montrer la faisabilité de l'approche constitue une étape cruciale dans le projet, indiquant que les suppositions émises quant à la versatilité de l'approche à base d'arbres sont fondées. La validation de cette versatilité est un premier pas encourageant, ouvrant la voie à d'autres travaux sur l'extension du cadre de la méthode. D'autre part, sur le plan pratique, les graphes que l'on peut être amené à explorer et exploiter dans le domaine biomédical peuvent contenir de nombreuses informations, tant dans les relations entre objets que dans leurs attributs. Ces graphes contiennent des groupes d'objets biologiques (protéines, médicaments, etc.), formant des groupes pouvant aider à la compréhension des mécanismes d'une maladie, à la catégorisation des médicaments, etc. Ce sont ces raisons qui nous ont poussé à explorer cette voie.

Dans ce chapitre, nous commencerons dans la section 5.1 par décrire l'approche que nous proposons, basée de nouveau sur des arbres aléatoires. Nous montrerons qu'il est possible de l'étendre au cas des graphes attribués avec deux procédures de calcul, avant de présenter les résultats de l'évaluation de notre approche, au sein de la section 5.2. Nous présenterons et discuterons enfin dans la section 5.3 les résultats obtenus sur un graphe extrait de notre base de connaissances locale, la FIGHT-HF GK Box, avec de conclure ce chapitre section 5.4.

5.1 Calcul de dissimilarités entre noeuds d'un graphe par des arbres aléatoires

5.1.1 Principes de l'approche

Comme nous l'avons mentionné précédemment, des partitions aléatoires d'un espace permettent le calcul de (dis)similarités entre objets de cet espace. Ces partitions peuvent être réalisées par le biais d'arbres aléatoires, dont les feuilles, ou plus généralement les noeuds, sont considérées comme des partitions de l'espace initial. À notre connaissance, ce type d'approche n'a pas encore été proposé dans le cas des graphes.

L'intuition de la méthode que nous proposons ici est d'effectuer une partition de ce type sur les noeuds d'un graphe. Une question importante est alors : dans le cas d'un graphe, comment les arbres aléatoires peuvent-ils être construits ? Plus précisément, que représentent les partitions dans ces arbres ? Notre proposition consiste ici à considérer l'ensemble des noeuds d'un graphe

comme un ensemble d'objets. Une confusion terminologique est ici possible, dans la mesure où il peut y avoir une ambiguïté entre les noeuds d'un graphe et les noeuds d'un arbre aléatoire construit à partir de ce graphe. Nous utiliserons le terme d'objets lorsque nous parlerons des noeuds d'un graphe. Ce choix n'est par ailleurs pas tout à fait arbitraire, dans la mesure où les noeuds des graphes que nous considérons ici représentent en effet des objets du monde réel et leurs relations. Néanmoins, dans certains cas, nous utiliserons le terme noeud pour ces objets, afin d'éviter les répétitions. Afin de lever l'ambiguïté dans ce cas, nous utiliserons le terme *noeud d'un graphe*.

5.1.2 Construction des UET sur les graphes

Notre proposition d'adaptation d'UET est la suivante. Lors d'une partition au niveau d'un noeud d'un arbre aléatoire, un objet v_s est tiré sans remise de l'ensemble des objets du graphe présents dans ce noeud. Le voisinage de v_s, noté $\Gamma(v_s)$, est par la suite considéré. L'ensemble des objets $v \in \Gamma(v_s)$ sont dirigés vers un noeud enfant, alors que l'ensemble des objets n'étant pas dans cet ensemble sont dirigés dans l'autre noeud enfant. Plusieurs représentations de ces relations peuvent être utilisées, telles que les matrices d'adjacence, ou encore les listes d'adjacence. Nous utiliserons dans le cadre de cette proposition les matrices d'adjacence, mais toute représentation permettant de récupérer les relations entre les objets du graphe peut l'être.

La croissance de l'arbre peut-être stoppée selon plusieurs critères. Nous pouvons ici garder les mêmes que pour les arbres construits sur des données structuées, à savoir la taille minimale d'un noeud pour qu'il ne soit pas considéré comme une feuille, ainsi que l'arrêt des partitions pour un noeud si ce dernier est homogène. Dans le cadre du chapitre précédent, cela correspond au cas où l'ensemble des objets prend la même valeur pour l'attribut sélectionné. Ici, cela correspond au fait que l'ensemble des objets du noeud concerné sont voisins de l'objet sélectionné v_s. Cela se traduit par une ligne ne contenant que des valeurs de 1 dans la matrice d'adjacence en regard de l'objet v_s. Dans ce cas, aucune partition ne peut être réalisée sur ce noeud, il est donc logiquement considéré comme une feuille. L'algorithme 9 présente la construction d'un arbre aléatoire prenant un graphe en entrée selon l'approche que nous proposons. On remarquera que ce dernier est très proche de celui présenté au chapitre 4. Notons ici que nous avons volontairement écarté le terme de *Graph trees*, qui peut être retrouvé dans les publications liées à la méthode présentée ici, pour des raisons de clarté. En effet, UET est une méthode générique permettant de partitionner l'espace, et nous pensons qu'il n'est pas judicieux de multiplier les termes pour dénommer deux applications différentes d'une même approche.

5.1.3 Une forêt, mais plusieurs méthodes de calcul de dissimilarité

Une fois les M arbres construits, il est possible d'y appliquer une procédure permettant de calculer une similarité. Il est par exemple possible d'utiliser la même approche que nous avons décrite dans l'algorithme 7 au cours du chapitre précédent, dite *fréquentiste*. Pour rappel, il s'agit d'incrémenter les similarités entre deux objets à chaque fois que ces derniers se trouvent dans les mêmes feuilles, et normaliser par le nombre d'arbres M dans la forêt afin d'obtenir des valeurs de similarité dans l'intervalle $[0, 1]$.

Algorithme 9 : Algorithme des UET appliqués aux graphes

Entrées : Un graphe $G(V, E)$

Sortie : Les feuilles d'un arbre aléatoire t

$S \leftarrow \emptyset$ //On initialise une pile vide;
$feuilles \leftarrow \emptyset$ //On initialise une liste vide destinée à stocker les feuilles ;
$v_s \leftarrow$ un noeud tiré sans remise de V;
$V_{gauche} \leftarrow \Gamma(v_s) \cup \{v_s\}$;
$V_{droit} \leftarrow V \setminus V_{gauche}$;
$Empiler(V_{gauche}, S)$;
$Empiler(V_{droit}, S)$ //On empile les nouveaux noeuds enfants à S ;
tant que $S \neq \emptyset$ **faire**
    $V_{noeud} \leftarrow$ dépiler S;
    si $|V_{noeud}| < n_{min}$ **alors**
        Ajouter V_{noeud} à $feuilles$;
    fin
    sinon
        si $\Gamma(v_s) \cup \{v_s\} \neq V_{noeud}$ **alors**
            $v_s \leftarrow$ un noeud tiré sans remise de V_{noeud};
            $V_{gauche} \leftarrow \Gamma(v_s) \cup \{v_s\}$;
            $V_{droit} \leftarrow V_{noeud} \setminus V_{gauche}$;
            Empiler(V_{gauche}, S);
            Empiler(V_{droit}, S);
        fin
        sinon
            Ajouter V_{noeud} à $feuilles$;
        fin
    fin
fin
retourner *feuilles*;

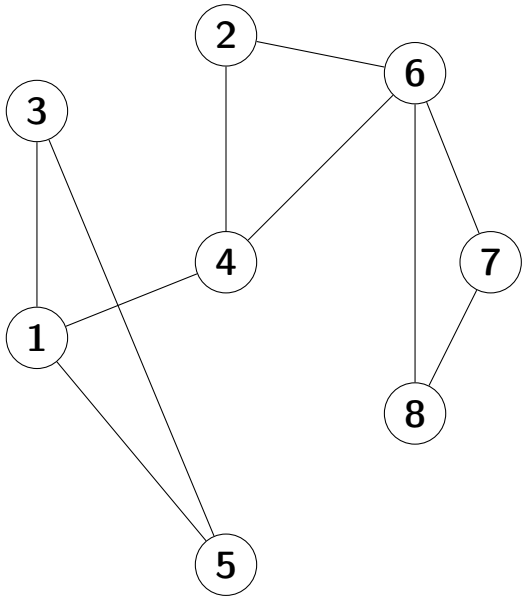


FIGURE 5.1 – Graphe d'exemple, contenant 8 noeuds.

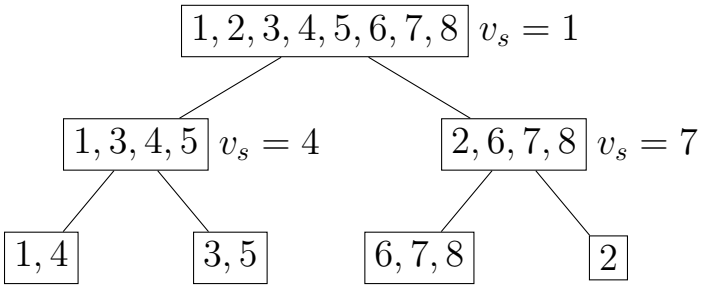


FIGURE 5.2 – Exemple d'UET construit sur la base du graphe d'exemple, avec $n_{min} = 4$. À chaque noeud, le noeud v_s du graphe sélectionné pour la partition est indiqué.

Soit le graphe d'exemple présenté à la figure 5.1. Il s'agit d'un graphe simple, contenant huit noeuds. Un exemple d'application d'UET sur ce graphe est présenté à la figure 5.2. Dans le cadre de cet exemple, le paramètre de lissage n_{min} est fixé à 4. Cet exemple permet de mettre en évidence un élément important relatif au fait qu'il s'agit d'une méthode d'ensemble. On note en effet que le noeud du graphe 2 se retrouve isolé dans une feuille. Selon l'approche fréquentiste décrite ci-dessus, sa similarité avec tout autre noeud du graphe ne sera pas incrémentée. Ainsi, si l'on ne devait se baser que sur cet arbre, ce noeud serait probablement dirigé vers un singleton, alors que l'on constate visuellement que ce n'est a priori pas le cas. Cette observation met en exergue l'importance de la forêt d'arbres : le processus aléatoire de sélection des noeuds v_s fait en sorte qu'en moyenne, le noeud 2 finira fréquemment dans les mêmes feuilles que les noeuds 4 et 6.

Au vu de l'aspect modulaire du calcul de (dis)similarités, d'autres méthodes peuvent être définies. Une alternative intéressante consiste par exemple à utiliser l'approche à base de masse proposée dans [TZC+16]. La propriété principale de cette mesure est que la dissimilarité, notée m_e, entre deux objets dans une région dense est plus grande que la dissimilarité entre ces mêmes

objets dans une région moins dense d'un même espace. Les auteurs partent d'une observation faite par une autre communauté, celle de la recherche en psychologie [Tve77][Kru78], qui est que la similarité entre deux objets est jugée plus grande si ces derniers sont dans une région moins dense que deux autres objets séparés par la même distance, dans une région plus dense. Un exemple spécifique est donné par les auteurs, permettant d'expliciter la notion. Soit une représentation des êtres humains par le biais de différents attributs plus ou moins arbitraires permettant de les différencier. Le postulat des auteurs est ici que la distance entre deux objets spécifiques, ici deux humains, ne devrait pas être la même partout, et devrait varier en fonction de la densité locale. Ainsi, en Europe, des individus de *type caucasien* seront jugés plus différents que ces deux mêmes individus en Asie, où ils seront jugés plus similaires.

Les auteurs de [TZC+16] parlent de *dissimilarité dépendante des données*. Ce terme est surprenant de prime abord, dans la mesure où l'on peut immédiatement se dire que toute mesure de dissimilarité dépend forcément des données. Le sens qu'ils donnent à cette phrase est ici un peu différent. En effet, les auteurs voient ici les dissimilarités ne dépendant pas des données comme donnant toujours les mêmes valeurs pour une paire d'objets, quel que soit leur voisinage. Ce qui n'est ici vraisemblablement pas le cas avec l'approche qu'ils proposent.

Cette mesure possède par ailleurs des propriétés surprenantes, parmi lesquelles le fait que la dissimilarité d'un objet avec lui même $m_e(x, x)$ n'est pas toujours égale à 0. En effet, celle-ci dépendant de la distribution des objets dans l'espace où cet objet se trouve, cette valeur de dissimilarité n'est pas constante. Bien que perturbante, d'autres propriétés nous permettent de retrouver des choses similaires à des mesures plus classiques. Tout d'abord, $\forall x \neq y, m_e(x, x) \leq m_e(x, y)$. Cette propriété est nécessaire, car bien que l'auto dissimilarité ne soit pas constante, il est important que cette dernière soit toujours inférieure à la dissimilarité de ce même point avec tout autre point. De plus, toutes les dissimilarités obtenues sont dans l'intervalle $[0, 1]$.

Voyons comment cette dissimilarité peut être obtenue. Soit $H \in \mathcal{H}(\mathcal{L})$ un partitionnement hiérarchique du jeu de données $\mathcal{L}$ en partitions ne se recouvrant pas, et non vides ; et $R(x, y|H)$ la plus petite région contenant x et y selon la partition H. La dissimilarité m_e estimée par un nombre fini de modèles – ici, des arbres aléatoires – est obtenue par la formule présentée équation 5.1, où $\tilde{P}(R) = \frac{1}{|\mathcal{L}|} \sum_{z \in \mathcal{L}} 1(z \in R)$. La figure 5.3 présente un exemple de partition hiérarchique d'un espace contenant les 8 noeuds du graphe 5.1.

$$m_e(x, y|D) = \frac{1}{M} \sum_{i=1}^{M} \tilde{P}(R(x, y|H_i)) \tag{5.1}$$

À titre d'exemple, calculons $m_e(1, 4)$ et $m_e(1, 8)$. Nous avons $m_e(1, 4) = \frac{1}{8}(2) = 0.25$, dans la mesure où la plus petite région où les objets 1 et 4 co-apparaissent contient 2 objets. Nous avons ensuite $m_e(1, 8) = \frac{1}{8}(8) = 1$, dans la mesure où la plus petite région contenant ces deux objets est le noeud racine, contenant 8 objets. Nous constatons bien ici que la distance entre les objets 1 et 4 est plus faible que celle entre les objets 1 et 8, ce qui est cohérent dans la mesure où la première paire d'objets reste groupée après deux partitions, alors que la deuxième ne l'est plus dès la première partition.

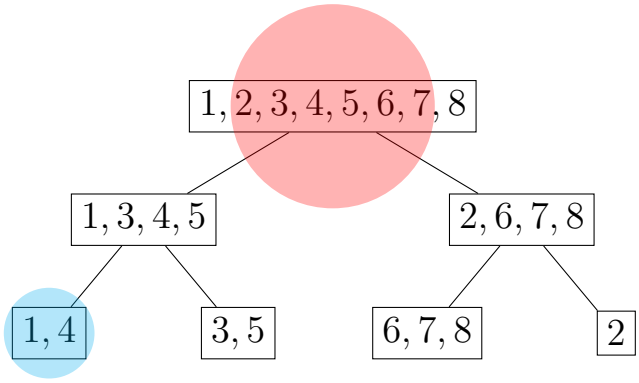


FIGURE 5.3 – Exemple de partition de 8 objets en régions en utilisant des arbres aléatoires. Les cercles bleu et rouge dénotent les noeuds (*i.e*, régions) les plus petits contenant les objets 1 et 4 et les objets 1 et 8, respectivement.

Au vu des propriétés intéressantes de cette approche de calcul, nous avons décidé de l'implémenter au sein de l'algorithme des UET appliqués aux graphes. Elle permet notamment l'obtention d'une matrice de dissimilarité directement en sortie, sans post-traitement. L'algorithme que nous proposons afin d'effectuer ce calcul est présenté dans l'algorithme 10. Dans la suite de ce document, lorsque nous mentionnerons les UET appliqués au graphe, nous nous référerons à cet algorithme de calcul de dissimilarité. Certains éléments de cet algorithme méritent d'être détaillés ici, dans la mesure où les mises à jour de la matrice de dissimilarité globale peuvent paraître au premier abord surprenantes. En effet, nous avons mentionné précédemment qu'UET représente la méthode permettant de construire les arbres, et que le *module* de calcul de (dis)similarité est à spécifier par l'utilisateur.

Cependant, on remarque que dans l'algorithme 10, les deux tâches sont effectuées de manière concomitante, et ceci pour des raisons d'optimisation. Résumons les grandes étapes de notre proposition, afin de la clarifier. Tout d'abord, une matrice de dissimilarité globale est initialisée. Cette dernière a pour taille $N \times N$ (pour rappel, N correspond au nombre d'objets, soit $|V|$ ici). Les arbres sont ensuite construits. Le principe de base de leur construction est le suivant : l'ensemble des objets du noeud courant est dépilé, et un objet v_s sur lequel sera effectué la partition est tiré sans remise dans une liste des objets propres à l'arbre. Si nous prenons pour exemple notre graphe de la figure 5.1, nous avons lors des premières étapes de construction de l'arbre $S = ([1, 2, 3, 4, 5, 6, 7, 8])$ et $V_{arbre} = [1, 2, 3, 4, 5, 6, 7, 8]$. En dépilant S, on récupère bien $[1, 2, 3, 4, 5, 6, 7, 8]$, correspondant bien au noeud racine de l'arbre, à savoir l'ensemble des noeuds du graphe. On tire sans remise $v_s = 1$ de V_{arbre} : une partition ne pourra plus être effectuée sur cet objet par la suite. En effectuant la partition, le noeud enfant de gauche correspond à $\Gamma(v_s) \cup \{v_s\} = [1, 3, 4, 5]$, et le noeud enfant de droite à $[2, 6, 7, 8]$. La valeur de n_{min} étant fixée à 4 dans le cadre de cet exemple, ces deux noeuds ne sont donc pas des feuilles, et sont empilés pour être traités ultérieurement. L'état de la pile après la ligne 30 est donc $S = ([1, 3, 4, 5], [2, 6, 7, 8])$.

La partie entre les lignes 30 et 32 mérite là aussi que l'on s'y attarde. En effet, on y voit que la matrice de dissimilarité est mise à jour, sans que l'on soit dans une feuille. Cela est lié à la définition même de l'approche à base de masse : la dissimilarité entre deux objets dépend de la taille du plus petit sous-espace les contenant. Ici, les objets qui ont été séparés entre les

deux noeuds enfants étaient donc au préalable dans le plus petit espace les contenant, on ajoute donc le nombre d'éléments présents avant partition pour les paires d'objets correspondants. On continue ensuite le processus, jusqu'à ce qu'il n'y ait plus de noeud à dépiler, ou plus d'objets à tirer sans remise. Dans ce dernier cas, s'il restait des noeuds à traiter dans l'arbre, on les aurait considérés comme des feuilles.

Ce calcul étant fait pour l'ensemble des M arbres, on incrémente ces valeurs dans la matrice globale, avant de diviser ces valeurs par M à la fin du calcul. Par ailleurs, rappelons que la taille des plus petites régions où les objets co-apparaissent doit être divisée par le nombre d'objets dans le jeu de données, soit n. Ainsi, une fois les valeurs de la matrice de dissimilarité D obtenues par incrémentation pour chaque arbre, ces dernières peuvent être divisées par $n \times M$ afin d'obtenir les dissimilarités finales.

Cette proposition d'algorithme permet d'intégrer le calcul des dissimilarités à l'intérieur de chaque boucle, en évitant le stockage et l'énumération explicite des objets présents dans chaque feuille. Au-delà d'un intérêt sur le plan de l'économie des ressources matérielles, on notera une relative facilité à le paralléliser, en veillant cependant à éviter les accès concurrents sur D.

Une implémentation de cet algorithme est disponible sur le dépôt Git suivant : `https://gitlab.inria.fr/kdalleau/graphtrees/`.

Le calcul de dissimilarités selon cette approche a une complexité temporelle de $\mathcal{O}(t\Psi log(\Psi) + n^2 t log(\Psi)$ [TZC+16], où t est le nombre d'arbres, Ψ la hauteur maximale des arbres, et n le nombre d'objets. Quand $\Psi << n$, cette complexité devient $\mathcal{O}(n^2)$.

5.1.4 Application à des graphes attribués

Nous avons vu qu'en effectuant des partitions successives de l'ensemble des noeuds d'un graphe sur la base de leur voisinage, il est possible de définir une mesure de (dis)similarité entre ceux-ci. Au vu du caractère stochastique de la méthode, n'utiliser les résultats obtenus sur la base d'un unique arbre ne serait pas pertinents, dans la mesure où leur variance serait très grande. Nous avons vu que la solution est de construire un ensemble de M arbres, constituant une forêt d'arbres.

Il est intéressant de noter qu'il est possible de construire des forêts contenant différents types d'arbres, spécialisés sur différentes modélisations des données : graphes, vecteurs d'attributs, etc. Là où nous avions jusqu'à présent des forêts à *essence unique*, nous proposons de construire des forêts *multiessences*, contenant des UET spécifiques aux données sous forme de vecteurs d'attributs extraites des attributs des noeuds du graphe, ainsi que des UET spécifiques au graphe. Cette approche permet le calcul de dissimilarités entre noeuds de graphes attribués, chaque type d'arbre étant spécifique à un aspect des données.

Algorithme 10 : Algorithme des UET, permettant le calcul de dissimilarités *via* l'approche basée sur la notion d'estimation de masse.

Entrées : Un graphe $G(V, E)$ de taille n, un nombre d'arbres M

Sortie : Une matrice de dissimilarités D

$D \leftarrow 0_{n,n}$ // Matrice nulle de taille $|V| \times |V|$;
pour *i in M* **faire**
  $S \leftarrow V$ //On initialise une pile avec l'ensemble des noeuds du graphe;
  $V_{arbre} \leftarrow V$ // On effectue une copie des indices des noeuds, locale à l'arbre.;
  tant que $S \neq \emptyset$ **faire**
    si $|V_{arbre}| == 0$ *// Il n'y a plus de noeuds à tirer au hasard;*
    alors
      pour $noeud \in S$ **faire**
        pour $u \in noeud$ **faire**
          pour $v \in noeud$ **faire**
            $D_{u,v} += |noeud|$;
      sortir;
    $V_{noeud} \leftarrow$ dépiler S;
    $v_s \leftarrow$ un noeud tiré sans remise de V_{arbre};
    si $\Gamma(v_s) \cup \{v_s\} \neq V_{noeud}$ *//v_s n'est pas voisin de tous les noeuds ;*
    alors
      $V_{gauche} \leftarrow \Gamma(v_s) \cup \{v_s\}$; $V_{droit} \leftarrow V_{noeud} \setminus V_{gauche}$;
      si $|V_{gauche}| < n_{min}$ *//Le noeud enfant de gauche est une feuille* **alors**
        pour $u \in V_{gauche}$ **faire**
          pour $v \in V_{gauche}$ **faire**
            $D_{u,v} += |V_{gauche}|$;
      sinon
        Empiler(V_{gauche}, S);
      si $|V_{droit}| < n_{min}$ **alors**
        pour $u \in V_{droit}$ **faire**
          pour $v \in V_{droit}$ **faire**
            $D_{u,v} += |V_{droit}|$;
      sinon
        Empiler(V_{droit}, S);
      pour $u \in V_{gauche}$ **faire**
        pour $u \in V_{droit}$ **faire**
          $D_{u,v} += |V_{gauche}| + |V_{droit}|$;
retourner $D/(n \times M)$;

Soient deux types de modèles de données τ_1 et τ_2. Notons T_{τ_1} l'ensemble des arbres spécifiques aux données de type τ_1, et T_{τ_2} l'ensemble des arbres spécifiques aux données de type τ_2. La forêt d'arbres *multiessences* correspond donc à $T = T_{\tau_1} \cup T_{\tau_2}$. Afin d'obtenir une mesure unifiée, il est nécessaire de spécifier en amont une méthode d'agrégation, potentiellement précédée d'une étape de prétraitement. En effet, dans le cas où les arbres de T_{τ_1} permettent d'obtenir une similarité – ce qui est le cas d'UET par exemple – alors que les arbres de T_{τ_2} permettent d'obtenir une dissimilarité, il sera nécessaire d'homogénéiser les valeurs obtenues avant agrégation. L'approche la plus directe afin d'agréger ces mesures est d'effectuer une somme pondérée, telle que présentée dans l'équation 3.9, que nous rappelons et adaptons ci-dessous.

$$d(u, v) = \alpha d_{\tau_1}(u, v) + (1 - \alpha) d_{\tau_2}(u, v) \tag{5.2}$$

La valeur du paramètre α permet de moduler l'importance d'un type par rapport à l'autre. Nous proposons, au moins dans un premier temps, de fixer cette valeur à $0,5$, signifiant que les informations présentes dans les relations du graphes et celles extraites des attributs sont de même importance. Nous verrons par la suite que cette hypothèse n'est pas toujours pertinente.

Nous pouvons par ailleurs envisager des forêts contenant plus de deux types d'arbres. Dans ce cas, la formule présentée peut être généralisée. Soit $Types = (\tau_1, \tau_2, \ldots, \tau_n)$ les différents types de données à considérer. L'équation 5.3 présente une méthode permettant d'agréger les mesures obtenues, avec $\sum_k \alpha_k = 1$.

$$d(u, v) = \alpha_1 d_{\tau_1}(u, v) + \alpha_2 d_{\tau_2}(u, v) + \cdots + \alpha_k d_{\tau_k}(u, v) \tag{5.3}$$

Une discussion de ce type d'agrégation de mesures a été effectuée précédemment, nous renvoyons donc le lecteur à la section 3.3.2. Le point important est ici que la valeur d'α est critique, et peut être complexe à déterminer. En effet, les différents types de données ne contenant a priori pas la même quantité d'information, leur pondération n'est pas triviale.

Dans les réflexions et expérimentations menées au cours de cette thèse, nous nous sommes concentrés sur le cas des graphes dont les noeuds sont attribués. Ce type de graphes ne représentent cependant qu'un sous-ensemble des graphes attribués existants : il serait intéressant notamment de pouvoir prendre en compte l'information contenue dans les arêtes lorsque ces dernières sont attribuées, qu'il s'agisse d'un poids ou encore d'un attribut qualitatif. Une extension à ce type de graphes peut-être envisagée, par l'ajout d'une mesure permettant de quantifier la dissimilarité en fonction des attributs portés par les arêtes. Cette extension n'a cependant pas été explorée dans cette thèse – faute de temps –, mais constitue une piste intéressante à poursuivre.

Dans l'espace des attributs, chaque noeud est ici considéré comme une instance, et les valeurs prises par ses attributs sont représentées comme un vecteur d'attributs (ou *tuple* d'attributs, en fonction du type de données). Ces derniers sont traités par les UET, afin d'en extraire une matrice de dissimilarité. Ce faisant, les avantages d'UET, notamment leur applicabilité sur des variables hétérogènes, sont transposés aux attributs des noeuds d'un graphe. Cela nous permet d'éviter l'utilisation de méthodes coûteuses et associées à une perte d'information, telle que la discrétisation de valeurs.

À notre connaissance, nos travaux sur le calcul de dissimilarités sur des graphes potentiellement attribués sont à ce jour les premiers à considérer ce calcul comme une tâche réalisable par l'utilisation d'arbres aléatoires. Cela apporte un éclairage nouveau sur la prise en compte du problème, et ouvre de nouvelles perspectives quant à un cadre unifié de calcul de dissimilarités sur différents types de données et différentes modélisations des données. Nous devons cependant effectuer une évaluation des approches présentées ici, afin de vérifier leur pertinence en pratique. Dans la section suivante, nous présentons les résultats que nous avons obtenus lors de l'évaluation d'UET dans les cadres des graphes non attribués et attribués.

5.2 Évaluation des UET appliqués aux graphes

Dans cette section, nous commencerons par comparer empiriquement les deux approches permettant d'obtenir des dissimilarités à partir des arbres, à savoir l'approche fréquentiste et l'approche à base de masse. Nous évaluerons ensuite notre algorithme selon deux approches différentes : dans un premier temps, par l'utilisation de mesures de qualité extrinsèques et en nous comparant à certains résultats d'autres travaux (section 5.2.2, et dans un second temps par l'utilisation de mesures de qualité intrinsèques (section 5.2.3). Nous évaluerons ici les deux cadres, à savoir attribué et non attribué.

5.2.1 Comparaison des deux procédures de calcul de dissimilarités

Nous avons vu que plusieurs approches de calcul de dissimilarités sont possibles, et en avons présentés deux : une approche dite fréquentiste, et une approche à base de masse. Nous nous proposons ici d'effectuer une comparaison entre les deux approches.

Nous proposons ici une évaluation sur des graphes réels et synthétiques, présentés dans la table 5.1. Deux graphes synthétiques sont générés par des modèles de blocs stochastiques, que nous avons définis dans le chapitre 3. Ces graphes sont nommés *SBM1* et *SBM2*, et diffèrent par leur nombre de noeuds et d'arêtes, ainsi que par les probabilités de création d'arêtes intra- et inter-communautés.

Le graphe *Email-Eu-Core* [YBLG17] [LKF07] représente des relations entre membres d'un institut de recherche où les arêtes représentent des communications entre ces membres. Le jeu de données contient par ailleurs des étiquettes correspondant aux groupes d'appartenance des membres. Chaque individu appartient à exactement un département parmi les 42 de l'institut.

Enfin, *RG* correspond à un graphe aléatoire d'Erdos-Renyi [ER60], où les arêtes sont créées avec une probabilité $p = 0.7$. Ce graphe ne présente aucune structure de communauté, et est utilisé en tant que contrôle négatif. En effet, si notre méthode met en évidence des groupes homogènes distincts sur ce graphe, cela serait le signe que l'approche que nous proposons ne fonctionne pas en pratique.

Nous avons appliqué UET avec deux approches. Dans un premier temps, nous avons utilisé l'approche vue dans le chapitre précédent avec UET, l'approche fréquentiste. Il s'agit d'incrémenter les similarités des objets qui co-apparaissent dans les mêmes feuilles, avant de normaliser et de transformer les similarités obtenues en dissimilarités par la relation $d = \sqrt{1-s}$. Dans

TABLE 5.1 – Graphes utilisés afin de comparer les deux approches de calcul de dissimilarités.

Graphe	# noeuds	# arêtes
SBM1	115	1226
SBM2	1005	25 571
Email-Eu-Core	2708	5429
RG	200	12928

un second temps, l'approche à base de masse est utilisée. Une fois les dissimilarités obtenues, nous avons appliqué t-SNE [MH08] afin d'obtenir une projection en deux dimensions des noeuds du graphe. Les visualisations de ces derniers, ainsi que les distributions des dissimilarités sont présentées dans les figures 5.4, 5.5, 5.6 et 5.7.

On constate que dans les deux modèles à base de blocs, l'approche à base de masse permet d'observer clairement trois communautés distinctes, alors que l'approche fréquentiste donne lieu à des groupes plus dispersés dans la projection. La différence entre les deux approches est encore plus grande dans le cas de *SBM2*, où la proportion d'arêtes entre différentes communautés est plus grande. Dans le cas de ce graphe, on constate même que le fait de ne pas utiliser l'approche à base de masse entraîne la perte d'une communauté dans la projection, celle-ci fusionnant avec un autre groupe. Dans le cas de *Email-Eu-Core*, la projection nous montre que l'approche à base de masse permet une meilleure distinction des communautés que l'approche fréquentiste, où l'ensemble des communautés semblent se regrouper. Enfin, l'application des deux approches sur le graphe aléatoire donne des résultats attendus, à savoir qu'aucune structure de communauté n'est observée dans les deux cas.

Dans les sections suivantes, nous allons donc effectuer l'évaluation d'UET sur des graphes attribués et non attribués, en calculant les dissimilarités avec l'approche à base de masse, dans la mesure où nous avons pu voir dans cette section que celle-ci semble plus intéressante en termes de qualité de discrimination, au delà de son intérêt en termes de temps de calcul.

5.2.2 Évaluation comparative en utilisant une vérité terrain

Commençons par évaluer la pertinence des résultats obtenus par les UET sur des graphes simples, non attribués. Cette évaluation est réalisée à la fois sur des graphes synthétiques et des graphes réels. Pour ce faire, nous comparons les NMI obtenues en utilisant UET avec celles obtenues par deux méthodes bien connues de clustering sur des graphes non attribués que nous avons présentées section 3.2, MCL [VD00] et Louvain [BGLL08].

Nous avons utilisé des jeux de données couramment choisis dans la littérature pour ce type de tâche d'évaluation, présentés dans la table 5.2. *Football* est un graphe modélisant les parties entre équipes de certaines universités étasuniennes en 2000, où chaque noeud représente une équipe et chaque arête représente le fait qu'une partie a été jouée entre les deux équipes correspondantes. Les étiquettes représentent l'appartenance de chaque équipe à une association sportive en particulier. *Email-eu-core*, décrit précédemment, représente les relations numérico-épistolaires entre membres de différents groupes de recherche. Le graphe *Polbooks* représente les relations entre

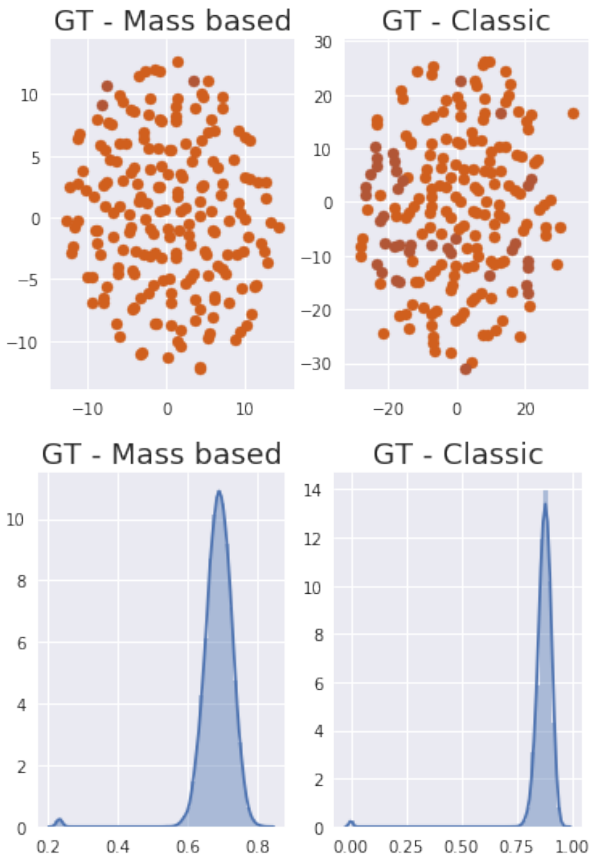


FIGURE 5.4 – Projection des noeuds d'un graphe aléatoire : les dissimilarités obtenues montrent une absence de groupe distinguable, ce qui était attendu. La figure du bas présente les distributions des dissimilarités entre noeuds.

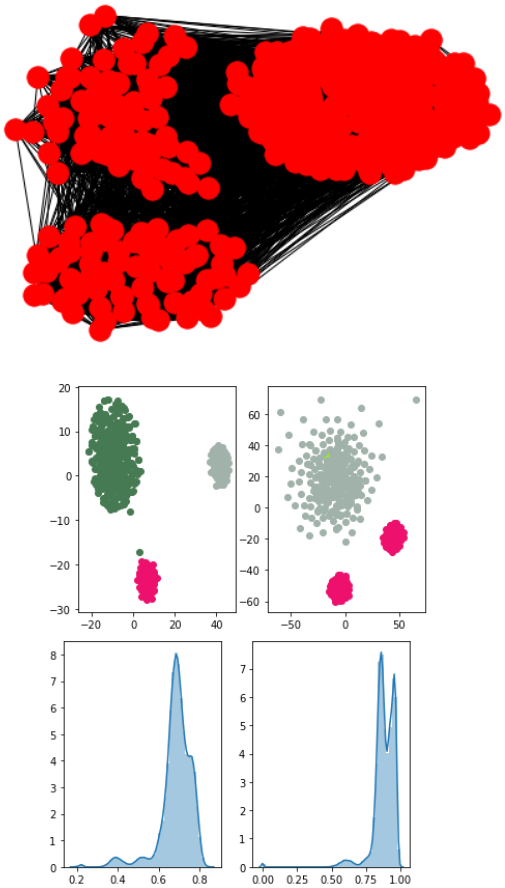


FIGURE 5.5 – SBM1 : les dissimilarités obtenues montrent que l'approche à base de densité permet l'observation de communautés clairement distinctes, avec la présence d'une communauté principale dense, alors que l'approche fréquentiste donne une communauté principale moins dense. Les figures de gauche correspondent aux dissimilarités obtenues utilisant l'approche à base de masse, et celles de droite à celles obtenues en utilisant l'approche fréquentiste.

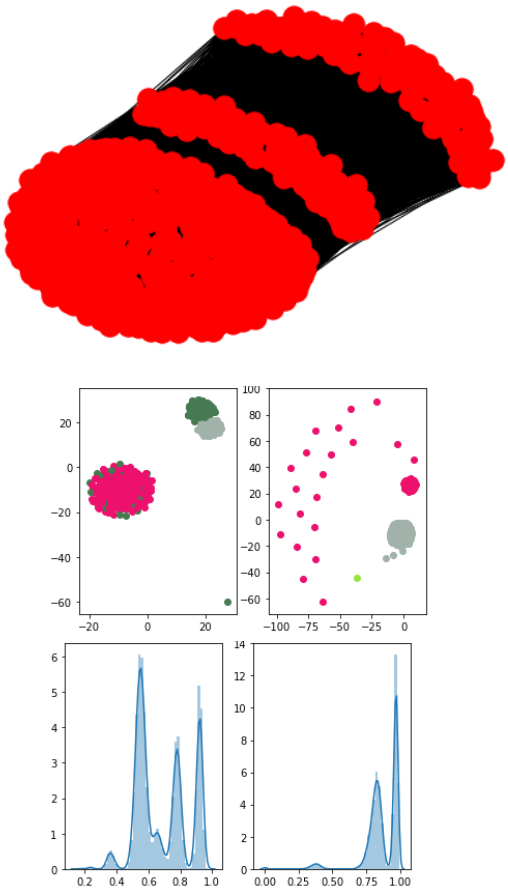


FIGURE 5.6 – SBM2 : les dissimilarités obtenues montrent que l'approche à base de masse donne des communautés clairement distinctes, contrairement à l'approche fréquentiste. De plus, on constate que la distribution de dissimilarités est plus étalée dans le premier cas. Les figures de gauche correspondent aux dissimilarités obtenues utilisant l'approche à base de masse, et celles de droite à celles obtenues en utilisant l'approche fréquentiste.

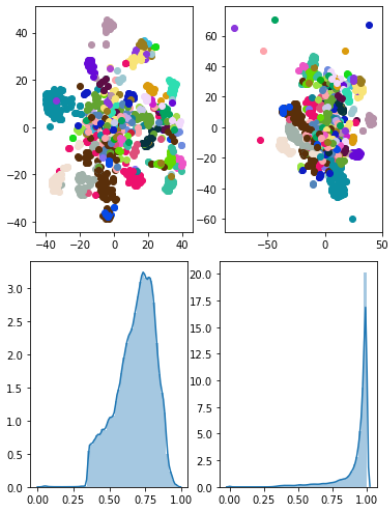


FIGURE 5.7 – Email-Eu-Core : l'approche à base de masse permet là encore une meilleure distinction des communautés que l'approche fréquentiste, où l'ensemble des groupes semblent moins distincts dans la projection. Les figures de gauche correspondent aux dissimilarités obtenues utilisant l'approche à base de masse, et celles de droite à celles obtenues en utilisant l'approche fréquentiste.

TABLE 5.2 – Graphes utilisés pour l'évaluation du clustering sur des graphes non attribués

Graphe	# Nœuds	# Arêtes	Degré moyen	# clusters
Football	115	1226	10,66	10
Email-Eu-Core	1005	25571	33,24	42
Polbooks	105	441	8,40	3
SBM3	450	65 994	293,307	3

livres traitant de politique, où les arêtes représentent des livres fréquemment achetés par les mêmes clients. Enfin, *SBM3* est un graphe synthétique, généré de la même manière que *SBM1* et *SBM2*.

La procédure est la suivante : après calcul des matrices de dissimilarités avec l'algorithme des UET, avec un nombre d'arbres $M = 200$, nous avons appliqué l'algorithme des k-moyennes sur les points obtenus après application de t-SNE sur les matrices de dissimilarités. Cette procédure est répétée 20 fois, et nous reportons les moyennes et écarts-types de NMI. Au cours de nos expériences, nous avons remarqué que le prétraitement des dissimilarités avant la tâche de clustering peut dans certains cas améliorer de manière significative la qualité de celui-ci. En particulier, l'application de la méthode *QuantileTransformer* de *Scikit-learn* [PVG+11] semble être très prometteuse. Cette transformation permet d'étaler les valeurs les plus fréquentes et de réduire l'impact des *outliers*. La figure 5.8 montre la distribution des dissimilarités entre noeuds

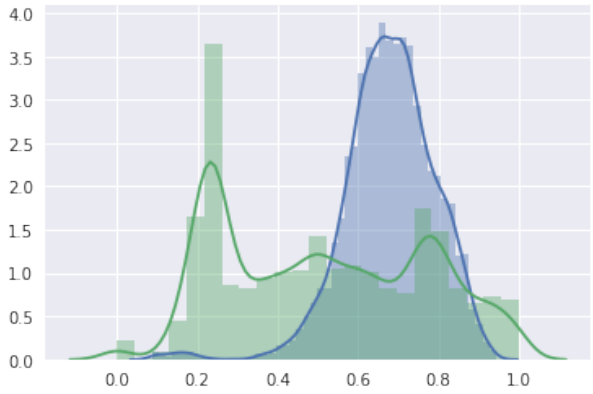


FIGURE 5.8 – Distribution des distances entre noeuds sans prétraitement (en bleu) et avec prétraitement (en vert).

TABLE 5.3 – Comparaison de NMI sur différents graphes. Les meilleurs résultats sont indiqués en gras.

Graphe	UET	Louvain	MCL
Football	**0.923 (0,007)**	**0.924 (0,000)**	0.879 (0,015)
Email-Eu-Core	**0.649 (0,008)**	0.428 (0,000)	0.589 (0,012)
Polbooks	0.524 (0,012)	0.521 (0,000)	**0.544 (0,02)**
SBM3	**0,998 (0,005)**	0,684 (0,000)	0,846 (0,000)

de *Polbooks* avec et sans prétraitement. On constate que la distribution de ces dernières est plus étalée, avec des pics plus visibles. Au cours des évaluations présentées dans ce chapitre, nous avons donc appliqué cette prétransformation à l'ensemble des matrices de dissimilarité obtenues, avec le nombre de quantiles $n_{quantile} = 10$.

Les résultats que nous avons obtenus en appliquant la procédure décrite précédemment sont présentés dans la table 5.3. Nous avons comparé les NMI moyennes par le biais du test t de Student, afin de vérifier que les différences entre valeurs obtenues sont statistiquement significatives.

Les résultats indiquent que notre méthode est compétitive vis-à-vis des deux autres méthodes utilisées ici. Dans le cas du graphe *SBM3*, on observe qu'UET donne de bien meilleurs résultats que les autres approches, et ce de manière significative. Cela semble confirmer ce que nous avions déjà pu constater précédemment, à savoir que cette approche semble particulièrement donner de bons résultats dans le cas de clusters de taille différente.

L'application d'UET sur ces quelques graphes semble donc montrer qu'en plus du réel intérêt d'utiliser des structures d'arbres aléatoires afin de calculer des dissimilarités dans un graphe, ces dernières peuvent effectivement être utilisées et, en particulier, être compétitives dans des tâches de clustering.

TABLE 5.4 – Graphes utilisés pour l'évaluation du clustering sur des graphes attribués par UET

Graphe	# noeuds	# arêtes	# attributs	# clusters
WebKB	877	1480	1703	4
Parliament	451	11646	108	7
HVR	307	6526	6	2
Lawyers	71	575	70	2

Au cours de la discussion faisant suite à la présentation d'UET, nous avons mentionné le fait qu'il serait possible d'envisager la construction de forêts à plusieurs essences afin d'effectuer un clustering sur des graphes attribués. Nous nous proposons ici d'effectuer une évaluation de la pertinence de cette idée. Nous allons ici utiliser quatre graphes, présentés dans la table 5.4.

WebKB représente les relations entre pages web de quatre universités où l'étiquette de chaque noeud correspond à l'une de ces universités, et où les attributs représentent les mots apparaissant sur ces pages. Le jeu de données *Lawyers* quant à lui représente des relations entre avocats, eux-mêmes décrits par des attributs, et le graphe *HVR* représente les relations entre gènes de la malaria. Enfin, le graphe *Parliament* représente des membres de l'Assemblée nationale française, reliés entre eux s'ils ont signé une proposition de loi ensemble. Leurs attributs représentent leur circonscription électorale, et leurs étiquettes correspondent à leur parti politique. On voit d'ailleurs ici les limitations de certains jeux de données. En effet, cette notion de circonscription est-elle pertinente dans l'exercice de recherche de groupes dans ce jeu de données ? Par ailleurs, la matrice contenant ces vecteurs d'attributs est creuse, chaque élu n'appartenant qu'à une circonscription. Il s'agit ici d'une digression, mais l'exemple de *Parliament* est particulièrement utile afin de mettre en évidence cette limitation.

Nous nous sommes ici comparés à des résultats de la littérature sur ces jeux de données. Nous avons utilisé l'ARI et la NMI en tant que mesure de qualité, en fonction de la mesure utilisée dans les travaux dont sont extraites les valeurs. La procédure est la suivante. Une forêt contenant des UET induits sur la base du graphe et des UET induits sur la base des attributs est tout d'abord construite. Les dissimilarités sont calculées dans chaque espace, avant d'être agrégées selon la relation de l'équation 3.9, avec $\alpha = 0.5$. Pour rappel, celle-ci consiste à calculer la somme pondérée de d_T et d_S, la pondération étant paramétrée par α. Enfin, la méthode des t-SNE est appliquée et l'algorithme des k-moyennes utilisé sur les points du nouvel espace. k est fixé au nombre de clusters, que nous connaissons dans la mesure où il s'agit de données étiquetées, où la *vérité terrain* est connue.

Cette procédure est répétée cinq fois, et nous reportons les moyennes et écarts-types des métriques de qualité considérées. La table 5.5 présente les NMI obtenues par notre approche ainsi que celles obtenues par les auteurs de [Boj18] sur les jeux de données *Parliament*, *Lawyers* et *HVR*. Elle contient par ailleurs les ARI obtenues par notre procédure ainsi que les valeurs obtenues par les auteurs de [MTO18] sur le jeu de données *WebKB*. Nous sommes ici bien conscients qu'il s'agit ici d'une évaluation limitée, dans la mesure où ne nous comparons qu'à deux travaux. Il ne s'agit donc pas ici d'une évaluation définitive et exhaustive, mais plutôt une

TABLE 5.5 – Comparaison de clusterings utilisant UET avec les meilleurs résultats de [Boj18] sur *Parliament*, *Lawyers* et *HVR*, et les résultats de [MTO18] sur le jeu de données *WebKB*. Les meilleurs résultats sont indiqués en gras.

Graphe	ARI/NMI UET	ARI/NMI Littérature
HVR	**1.00 (0,000)**	0.89
Parliament	0.65 (0,039)	**0.78**
Lawyers	0.12	**0.66**
WebKB	**0.999 (0,002)**	0.995 (0,002)

expérimentation nous permettant de nous situer vis-à-vis de quelques travaux que nous suivions à l'époque de ces tests.

On constate que la forêt multiessence d'UET donne de bons résultats dans le cas des graphes *WebKB* et *HVR*. En revanche, pour les autres jeux de données, les résultats de la littérature sont meilleurs que notre approche. Avant de discuter ces résultats, intéressons-nous à une autre manière de quantifier la qualité des clusters obtenus. En effet, nous nous sommes ici basés sur des vérités terrain fournies par les jeux de données afin d'effectuer une évaluation par le biais d'une mesure de qualité extrinsèque, la NMI. Ce n'est cependant pas la seule approche possible. Il est en effet possible de baser l'évaluation sur une mesure de qualité intrinsèque, telle que la modularité. Passer par ce type de mesure permet notamment de s'affranchir de la vérité terrain, qui n'est d'une part pas toujours disponible, et qui d'autre part peut être biaisée. Nous allons donc dans la section suivante présenter une méthode d'évaluation basée sur la modularité, ainsi que les résultats que nous avons obtenus.

5.2.3 Évaluation sur la base de la qualité intrinsèque des clusters obtenus

Commençons par rappeler que la modularité d'un graphe, vue section 3.2, est basée sur l'intuition qu'un bon clustering implique une forte proportion d'arêtes entre noeuds de mêmes clusters et une faible proportion d'arêtes entre noeuds de différents clusters. Nous commençons ici par effectuer cette évaluation sur les mêmes graphes de la section précédente, présentés dans la table 5.2. La procédure est la suivante : après avoir calculé les matrices de dissimilarité *via* UET, nous effectuons un plongement avec t-SNE et appliquons l'algorithme des k-moyennes sur les points obtenus. Nous calculons enfin les modularités en utilisant la fonction dédiée dans la bibliothèque *Networkx*. On calcule ici deux modularités : l'une sur la base des clusters obtenus par le biais de notre procédure, et l'autre sur la base des vérités terrain. Les résultats obtenus sont reportés dans la table 5.6. On peut ici noter plusieurs choses. Tout d'abord, les deux modularités obtenues pour chaque graphe sont, dans tous les cas, très proches. On remarque même une égalité pour le jeu de données synthétique, qui avait déjà donné de bons résultats dans la section précédente. Cela semble confirmer le fait que les dissimilarités obtenues par le biais d'UET sont pertinentes et permettent bien d'effectuer un *bon* clustering.

Un résultat est ici intéressant, sur le graphe *Email-Eu-Core*. On constate en effet une modularité supérieure dans le cas des clusters obtenus par notre procédure. Cela peut être expliqué de plusieurs manières. Il est d'une part possible que cela soit dû aux limites propres à la mesure

TABLE 5.6 – Modularités obtenues après clustering en utilisant les dissimilarités calculées par UET ou les vérités terrain. Les meilleurs résultats sont indiqués en gras.

Graphe	UET	Étiquette
Football	0.55	**0.60**
Email-Eu-Core	**0.31**	0.24
Polbooks	0.41	**0.50**
SBM3	**0.22**	**0.22**

TABLE 5.7 – Modularités obtenues après clustering en utilisant les dissimilarités calculées par UET appliqué uniquement sur le graphe, par une forêt d'UET multiessences (graphes et attributs), et enfin en utilisant les étiquettes. Les meilleurs résultats sont indiqués en gras.

Graphe	UET - Graphes	UET - Multiessences	Étiquette
WebKB	**0.74**	0.70	**0.74**
HVR	0.15	0.15	0.15
Parliament	**0.46**	0.15	0.20
Lawyers	**0.27**	0.23	**0.26**

de qualité. En effet, les clusters formés par les vérités terrain présentent des différences de tailles relativement conséquentes, le plus petit cluster ne contenant qu'un élément, et le plus grand plus de cent ; alors que les clusters obtenus par le biais de notre approche sont de taille plus homogène. La limite de résolution de la modularité dans les graphes où les clusters sont de taille différente est un problème connu, et nos observations sont probablement liées à cette limitation. Cependant, se pose aussi la question de la pertinence des étiquettes attribuées en tant que vérité terrain. En effet, ces dernières peuvent dépendre d'une tâche bien spécifique, ou encore représenter de manière forcée - et très humaine - une représentation du monde. Les résultats obtenus par le biais de méthodes non supervisées peuvent être éloignés de cette représentation du monde, sans pour autant être faux. Cela met en lumière la nécessité d'analyser les résultats obtenus afin de pouvoir statuer, et éventuellement en extraire des informations, puis des connaissances intéressantes.

Intéressons-nous maintenant au cas des graphes attribués. Nous utiliserons ici les mêmes graphes que ceux de la section précédente. Comme nous l'avions mentionné, ne se comparer qu'à quelques résultats de la littérature ne constitue qu'une étape préliminaire. Il est aussi intéressant d'effectuer une évaluation qui est indépendante des résultats de la littérature. Pour ce faire, nous avons calculé la modularité des clusterings obtenus dans trois cas : (i) en utilisant la dissimilarité obtenue par UET, uniquement sur la structure du graphe ; (ii) en utilisant les dissimilarités obtenues par UET sur la base des attributs et UET sur la base du graphe ; et (iii) en utilisant les étiquettes fournies avec les jeux de données. Les résultats obtenus sont présentés dans la table 5.7.

Comme pour les graphes non attribués, on constate que notre approche donne de bons résultats, parfois meilleurs que ceux obtenus en utilisant les étiquettes. Cette observation peut là

encore s'expliquer par le fait les *vérités terrain* ne constituent pas toujours une vérité absolue, notamment en termes de structure du graphe.

Nous avons constaté précédemment que dans deux graphes *Parliament* et *Lawyers*, les résultats de la littérature étaient meilleurs que notre approche sur la base de mesures de qualité extrinsèques. On remarque ici en comparant les modularités obtenues avec UET sur l'unique base du graphe et avec une forêt multiessences qu'ajouter l'information contenue dans l'espace des attributs peut donner un clustering de moins bonne qualité. C'est d'ailleurs le cas pour l'ensemble des quatre jeux de données. Les mauvais résultats obtenus précédemment peuvent donc être dûs à l'ajout de ces informations de l'espace des attributs, et ce pour plusieurs raisons potentielles. Premièrement, le choix de la méthode de calcul de dissimilarités dans cet espace – ici UET – peut ne pas être le meilleur et dégrader les performances globales. De plus, l'importance de chaque aspect des données peut varier d'un jeu de données à un autre. En effet, dans certains cas, la structure du graphe peut être plus discriminante et contenir plus d'informations que les attributs des noeuds. Le paramètre α étant fixé par défaut à 0.5, on considère ici que les deux espaces ont la même importance, ce qui n'est pas forcément le cas. Il s'agit ici des valeurs *standard* d'αÇ, dont une évaluation empirique pourrait montrer les limites. Enfin, les informations contenues dans les deux aspects des données peuvent ne pas être concordantes. Dans ce cas, par exemple, le fait de considérer les attributs peut n'ajouter que du bruit à la dissimilarité globale, et ceux-ci ne devraient pas être pris en compte.

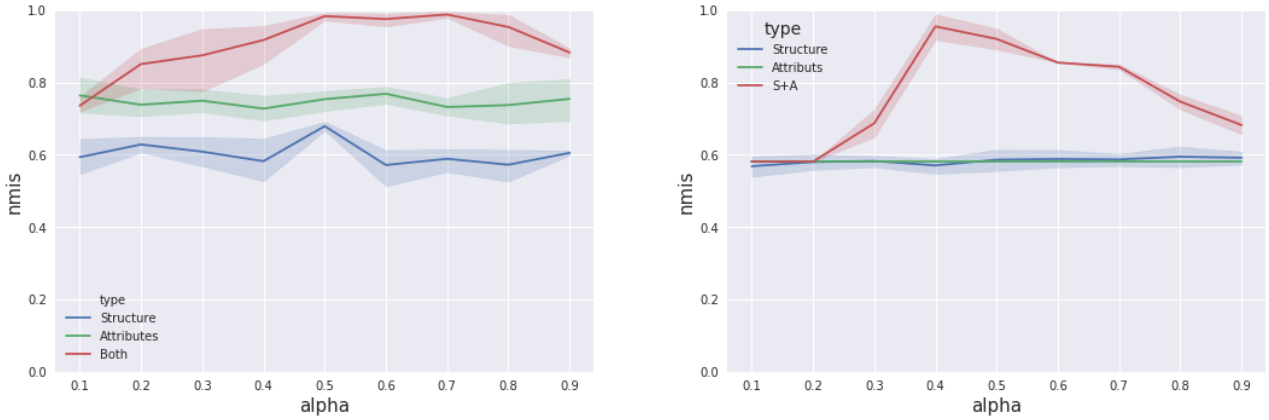
Afin d'avoir une idée générale de l'influence d'α sur les résultats, nous avons effectué un clustering pour différentes valeurs de ce paramètre sur les quatre graphes attribués utilisés ici. Les valeurs sont incrémentées par pas de $0,1$, et 20 clusterings sont effectués et évalués par le biais de la NMI, avant d'en effectuer la moyenne. Les courbes obtenues (figures 5.9 et 5.10) semblent confirmer nos hypothèses précédentes. On constate en effet que dans le cas du graphe de parlementaires, l'information contenue dans les attributs est nulle, et qu'il n'est par exemple pas nécessaire de la prendre en compte. On remarque par ailleurs que la valeur optimale d'α n'est pas toujours de $0,5$. Fait étonnant, pour *HVR*, on constate que cette valeur est à $0,4$, alors que les deux espaces pris individuellement semblent donner des clusterings de qualité semblable.

Globalement, ces résultats sont encourageants, et indiquent que l'approche des UET semble être une bonne candidate afin d'obtenir des dissimilarités pertinentes entre noeuds d'un graphe. L'extension au cadre des graphes attribués par le biais des forêts multiessences reste cependant une voie à poursuivre, afin d'attester de son intérêt pratique.

5.3 Application sur des graphes du projet FIGHT-HF

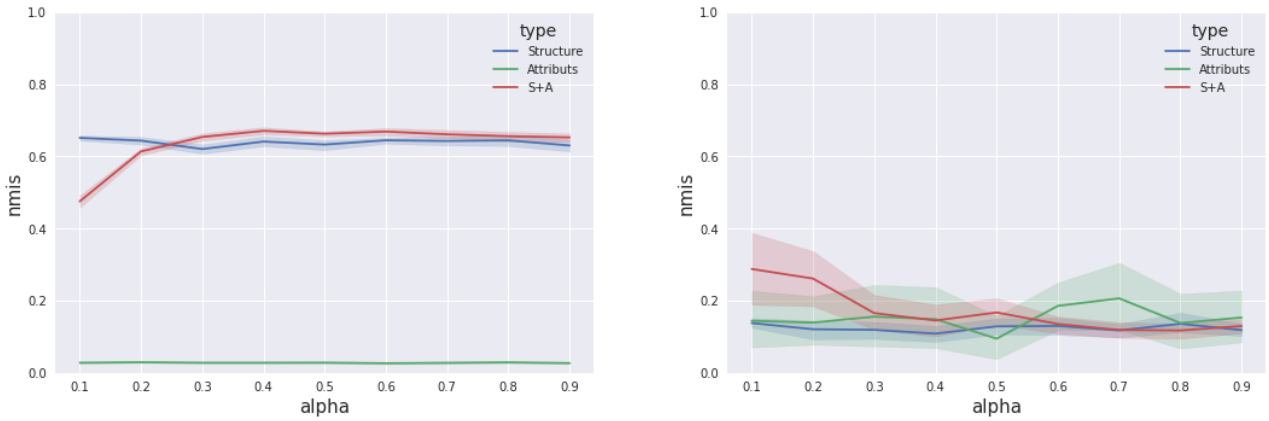
5.3.1 Description du cadre d'application

Nous avons déjà mentionné précédemment le déploiement d'une base de connaissances dans le cadre du projet FIGHT-HF, la Fight-HF GK Box (FHF-GKB), dont le travail d'intégration et de déploiement a été réalisé en partenariat avec la société EdgeLeap. Il s'agit d'une base de



(a) (b)

FIGURE 5.9 – Évolution de la NMI en fonction de α, considérant uniquement la structure, uniquement les attributs, ou les deux pour le graphe *WebKB* (a) et *HVR* (b).



(a) (b)

FIGURE 5.10 – Évolution de la NMI en fonction de α, considérant uniquement la structure, uniquement les attributs, ou les deux pour le graphe *Parliament* (a) et *Lawyers* (b).

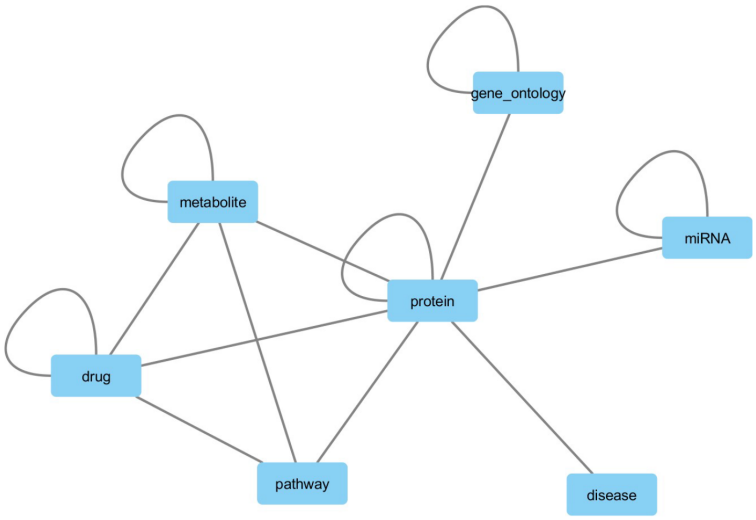


FIGURE 5.11 – Rappel du métagraphe de la FIGHT-HF GK Box.

connaissance contenant des relations provenant de ressources publiques telles qu'*Uniprot*, *Disease Ontology*, *Disgenet*, ou encore *WikiPathways*. Les informations contenues dans ces données publiquement accessibles sont agrégées sous forme d'un graphe. Le métagraphe de la FHF-GKB, présenté section 3.3, est rappelé à la figure 5.11. L'information importante de cette figure est que les noeuds peuvent être de sept types différents : *drug*, *pathway*, *metabolite*, *phenotype*, *tissue*, *gene*, ou *miRNA*. Le graphe stocké dans cette base de connaissances est cependant bien plus complexe que ce que ce métagraphe ne le laisse supposer. En effet, l'hétérogénéité des noeuds n'est pas le seul élément rendant difficile la tâche d'extraction de connaissances de ce graphe. En particulier, ces noeuds peuvent être attribués, de même que les relations entre eux.

Dans la mesure où plusieurs domaines sont couverts par ce graphe, il est nécessaire de délimiter un cadre d'étude au préalable. Nous nous sommes dans le cadre de ce document focalisés sur un type de noeud bien spécifique, les médicaments, de type *drug*. Notons que bien que nous éliminons ainsi la complexité liée à l'hétérogénéité des noeuds, celle liée aux différents types de liens possibles subsiste. En effet, un médicament peut agir sur un métabolite, peut participer à une voie métabolique, ou encore interagir avec une protéine associée à une maladie. On peut voir à la figure 5.12 un sous-graphe présentant les différents types de relations que peut avoir un noeud de type *drug* avec un noeud de type *protein*. En fonction de la tâche, le type d'une relation peut être important. On peut par exemple n'être à la recherche que de médicaments qui activent une protéine (relation *activates*), relation plus spécifique qu'une *simple* interaction (relation *interacts_with*). Dans le cadre de notre application, nous n'avons considéré que la présence d'une relation entre objets, et avons omis le type de celle-ci. Ce choix est motivé par le fait que la solution que nous proposons dans cette thèse ne permet pas, à l'heure actuelle, la prise en compte des attributs sur les arêtes.

Décrivons à présent l'expérimentation que nous avons effectuée sur cette base de connaissance, ainsi que les résultats obtenus.

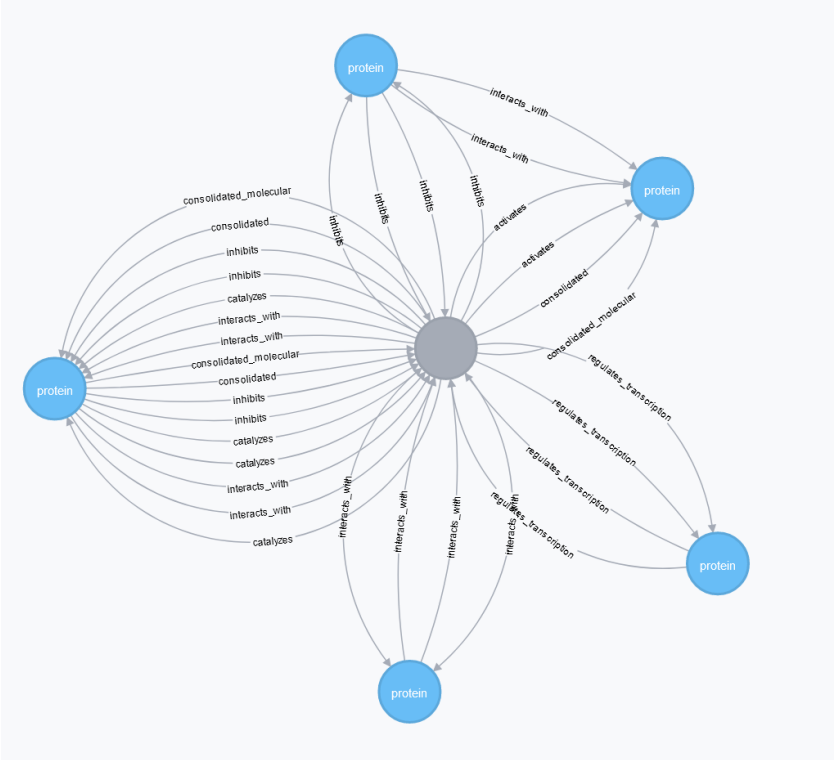


FIGURE 5.12 – Exemple de relations possibles avec un noeud de type *drug* dans la FIGHT-HF GK Box.

TABLE 5.8 – Caractéristiques du graphe extrait de la FHF-GKB

# Nœuds	# Arêtes	Degré moyen
1707	128191	154.09

5.3.2 Application d'UET sur des graphes extraits de la FIGHT-HF GK Box

Stratifier les patients soufrant d'insuffisance cardiaque (IC), première cause de décès dans le monde, et comprendre les mécanismes qui conduisent aux différentes formes de la maladie constituent les objectifs scientifiques du programme de RHU Fight-HF (ANR 15-RHUS-0004 [30]). Dans ce contexte, l'arsenal thérapeutique pour prendre en charge l'IC gagne a être enrichi car certaines catégories de patients ne répondent pas favorablement aux traitements actuels [ZKL+06]. Il existe principalement deux alternatives. La première est la conception de nouveaux médicaments grâce aux techniques de criblage virtuel de banques de molécules utilisées dans le protocole dit de *drug design*. La deuxième alternative est l'identification de molécules déjà sur le marché et susceptibles d'être repositionnées avec de nouvelles indications (c'est ce que l'on appelle le *drug repurposing*).

L'objectif de notre mini-étude est ainsi d'identifier des groupes de médicaments (ou plus généralement de substances) homogènes afin d'en dériver des informations potentiellement nouvelles en lien avec la fibrose, un mécanisme clé dans la survenue de l'insuffisance cardiaque. Pour ce faire, nous avons construit un graphe à partir des données présentes dans la FHF-GKB, selon la procédure suivante :

- L'ensemble des noeuds de type *protein* reliés aux noeuds de type *disease* dont l'étiquette comporte le terme *fibrosis* sont extraits.
- L'ensemble des noeuds de type *médicament* reliés aux protéines extraites à l'étape précédente sont extraits.
- Un nouveau graphe contenant ces noeuds de type *médicament* est créé, et des arêtes sont ajoutées entre paires de noeuds ayant une relation avec une protéine commune.

Nous obtenons donc un graphe homogène où les noeuds représentent des médicaments ayant une interaction - quel que soit son type - avec des protéines liées à l'insuffisance cardiaque, et où ces noeuds sont reliés entre eux sur la base d'interactions communes. Les caractéristiques de ce graphe sont détaillées dans la table 5.8.

Nous appliquons ensuite UET sur le graphe obtenu, avant d'appliquer un algorithme de clustering, les k-moyennes. Une projection des noeuds obtenue *via* t-SNE est présentée à la figure 5.13. Les étiquettes représentent les clusters attribués par l'algorithme.

On remarque ici une certaine partition en clusters mais qui ne semble pas naturelle, notamment au centre de la projection. Il s'agit ici de l'une des limitations de l'algorithme des k-moyennes que nous avions mentionnée section 2.3.3, à savoir que les clusters retournés par la méthode sont convexes. De plus, certains points sont attribués à un cluster alors qu'ils représentent plus probablement leur propre cluster. Enfin, le nombre de clusters devant être donné *a priori*, cette valeur a été déterminée par un processus d'essai-erreur et ne correspond pas forcément à la réalité du terrain. Il semblerait ici qu'une approche à base de densité, telle que décrite section 2.3.4, soit plus adaptée. Après application de DBSCAN sur les mêmes dissimilarités, nous obtenons le clustering présenté à la figure 5.14. Notons ici que les points semblent plus dispersés

30. `https://anr.fr/ProjetIA-15-RHUS-0004`

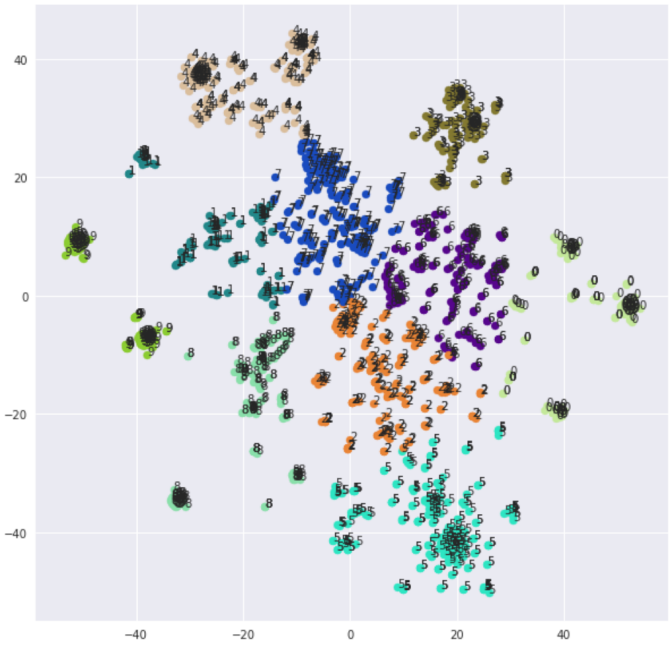


FIGURE 5.13 – Projection des noeuds de l'extrait de la FHF-GKB par t-SNE, après clustering avec l'algorithme des k-moyennes.

que sur la figure 5.13. Cela est dû à un changement d'échelle lors de la génération de la figure, afin de pouvoir plus aisément discerner les étiquettes de chaque cluster..

Nous allons ici nous concentrer sur quelques clusters. Commençons par le cluster 22 par exemple qui est un petit groupe de sept substances :

- Acide arotinoïde
- Isotrétinoïne
- Adapalène
- Tazarotène
- Étretinate
- Un dérivé d'acide benzoïque
- Un dihydronaphtalène, le BMS-493

Ce groupe est intéressant, dans la mesure où l'ensemble de ces substances appartiennent soit à la classe des rétinoïdes, soit sont de manière plus générale des agonistes/antagonistes aux récepteurs à l'acide rétinoïque. Il s'agit de substances utilisées notamment en dermatologie, pour le traitement de l'acné ou du psoriasis, pour leurs effets de régulation de la croissance et de la différenciation des cellules épithéliales. Ce premier cluster, représentant un groupe pertinent et homogène de substances, nous indique que la dissimilarité calculée *via* UET sur le sous-graphe est de bonne qualité, et permet bien de modéliser les distances entre les noeuds du graphe.

Attardons-nous maintenant sur un autre cluster, le 41, que l'on retrouve en haut sur la figure 5.14. Ce dernier contient 28 susbtances :

- Procaïnamide
- Cinchocaïne

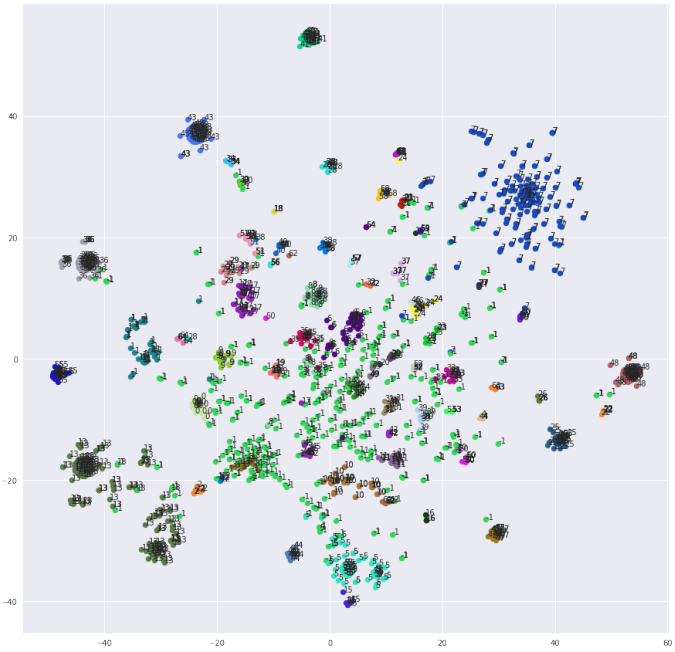
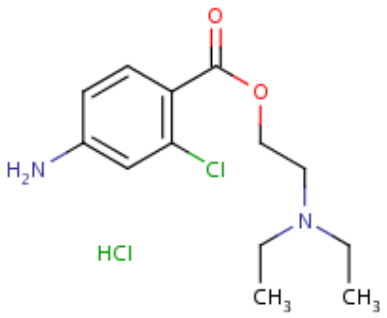


FIGURE 5.14 – Projection des noeuds de l'extrait de la EdgeBox par t-SNE, après clustering par l'algorithme DBSCAN.

— Procainamide
— Dyclonine
— Benzonatate
— Mexiletine
— Mepivacaïne
— Orphenadrine
— Disopyramide
— Oxybuprocaine
— Prilocaïne
— Alcool 2,4-dichlorobenzylique
— Encaïnide
— N-acetylprocainamide
— Lidocaïne
— Moricizine
— Eslicarbazepine
— Chloroprocaïne
— Ethotoïne
— Flecainid
— Tetracaïne
— Tocaïnide
— Zonisamide
— Quinidine

Pour des raisons de concision, nous avons ici volontairement éliminé les notations de sels dans la dénomination, et fusionné les différents sels de la même substance. On remarque ici

(a)



(b)

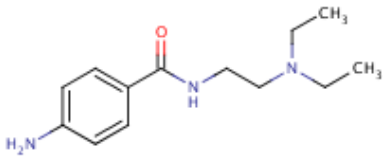


FIGURE 5.15 – Structure d'un anesthésique local, la chlorprocaïne (a), et d'un antiarythmique, la procaïnamide (b). On constate une forte similarité en termes de structure.

que ce cluster contient des substances de deux types : d'une part, des anesthésiques locaux (prilocaïne, lidocaïne, etc.), et d'autre part des antiarythmiques (procaïnamide, mexiletine, etc.). Bien que le fait que ces deux classes coexistent au sein d'un même cluster peut sembler étrange au premier abord, ce clustering est cohérent. En effet, de nombreuses substances ayant des effets antiarythmiques présentent aussi des effets anesthésiants [GH87]. Une forte similarité en termes de structure, laissant supposer ce chevauchement en terme d'activité pharmacologique, peut aussi être observée, comme on peut le voir à la figure 5.15. On retrouve enfin quelques substances liées à l'activité anticonvulsive, telles que l'eslicarbazépine ou le zonisamide. Le fait de retrouver ces substances, couramment utilisées dans le cadre des traitements de l'épilepsie, peut-être expliqué par le fait qu'ils bloquent les canaux dépendants au sodium, mécanisme partagé avec les antiarythmiques et anesthésiques locaux. L'unique substance dont la présence dénote du reste du cluster est l'alcool 2,4-dichlorobenzylique. Ce dernier est un antiseptique local, couramment retrouvé dans les pastilles utilisées dans le traitement des maux de gorge. Cependant, certaines études semblent en effet montrer un effet anesthésiant de ce type de substance – en particulier de l'alcool benzylique, proche de la substance retrouvée ici – [WM99, BLA+09].

Commentons un dernier cluster contenant les substances suivantes (les sels, ainsi que les métabolites des substances présentées ci-dessus ont été omis pour des raisons de concision) :

— Périndopril

— Benazépril
— Quinapril
— Trandolapril
— Val-Pro-Pro
— Spirapril
— Fosinopril
— Rescinnamine

Ici, on constate qu'il s'agit d'un cluster dont les substances sont bien connues dans le cas du traitement des pathologies cardiovasculaires : il s'agit en effet pour la majorité d'inhibiteurs de l'enzyme de conversion de l'angiotensine, substances à l'activité antihypertensive. La substance sortant de l'ordinaire ici est un tripeptide, Val-Pro-Pro. Au vu du groupe auquel il appartient, on peut supposer que ce tripeptide présente une action antihypertensive. Il semblerait en effet que cela soit le cas, au vu de certaines études [MNON17, TEK$^+$11].

Ces résultats nous montrent que d'une part, l'application d'UET sur un cas d'application pratique donne des résultats pertinents, cohérents avec les connaissances du domaine. D'autre part, l'analyse des clusters obtenus ici peut mener à la découverte de nouveaux traitements intéressants à explorer pour l'insuffisance cardiaque, en ciblant les molécules jusque là non utilisées à cet effet et appartenant à un cluster de traitements connus.

Nous avons effectué cette exploration sur un graphe que nous avons rendu homogène, *i.e* constitué d'un seul type de noeuds, pour les besoins de l'expérience. Ce type d'approche peut être retrouvé dans d'autres travaux, tels que [ZMKS15]. En pratique, ceci met en lumière une possibilité d'amélioration de notre méthode, que nous n'avons jusqu'alors pas mentionnée, à savoir l'application aux graphes hétérogènes. Bien qu'il serait possible d'appliquer UET sur un graphe hétérogène sans prétraitement, les résultats obtenus seraient au mieux incomplets, puisque qu'ignorant une partie de l'information disponible (à savoir l'information portée par le type de noeud), au pire inexploitables. Une piste d'amélioration serait la prise en compte de graphes hétérogènes, potentiellement par le biais de l'ajout d'attributs aux noeuds ou aux arêtes.

5.4 Bilan du chapitre

Nous avons présenté ici une approche permettant de calculer des dissimilarités entre noeuds d'un graphe, basée sur des arbres aléatoires. L'intuition est très proche de celle de notre précédente contribution, où ces arbres aléatoires sont utilisés afin de séparer l'espace initial des attributs en de multiples sous-espaces, permettant *in fine* d'obtenir des (dis)similarités entre objets.

La méthode présentée ici divise de l'espace des noeuds d'un graphe en sélectionnant de manière aléatoire l'un des noeuds de cet espace, avant d'effectuer la partition. Cette partition est réalisée grâce à la notion de voisinage : l'ensemble des noeuds voisins du noeud sélectionné est associé à un nouveau sous-espace, alors que tous les autres sont associés à un autre sous-espace. Cette partition est réalisée de manière récursive jusqu'à ce qu'une condition d'arrêt soit satisfaite.

Une fois les forêts d'UET construites, il est possible d'obtenir une mesure de (dis)similarité, soit en utilisant une approche similaire à celle utilisée dans UET, soit en calculant une dissimilarité basée sur l'estimation de masse. Nous montrons que cette dernière méthode tend à donner de meilleurs résultats, notamment sur les graphes dont les densités d'arêtes entre communautés sont fortement différentes.

Nous avons montré que l'approche que nous proposons donne des résultats prometteurs, notamment en les comparant à des résultats obtenus par des méthodes couramment utilisées dans la recherche de communautés, à savoir *Louvain* et *MCL*. Nous avons aussi montré qu'il est théoriquement possible d'obtenir des dissimilarités entre noeuds de graphes attribués, en construisant des forêts mixtes contenant à la fois des UET spécialisés sur les graphes et des UET spécialisés sur les attributs des noeuds. En revanche, les résultats obtenus par cette approche sont mitigés. En effet, notre évaluation empirique – en utilisant UET comme méthode de calcul de dissimilarité dans l'espace des attributs – montre que la qualité des clusterings sur la base des dissimilarités calculées varie fortement d'un graphe à l'autre.

Cette observation peut être due à différents facteurs. Premièrement, l'importance des attributs des noeuds est dépendante du graphe considéré, et dans certains cas, considérer ces attributs peut ajouter du bruit plutôt que de l'information. De plus, le choix de la méthode d'agrégation entre les différentes dissimilarités joue un rôle essentiel. Au cours de nos évaluations, cette agrégation est effectuée par le calcul de la somme pondérée $d_{v_i,v_j} = \alpha d_T(v_i, v_j) + (1 - \alpha) d_S(v_i, v_j)$, avec $\alpha = 0,5$. Le même poids est donc attribué aux deux espaces, ce qui ne correspond pas toujours à la réalité du terrain. Enfin, le choix de l'algorithme utilisé afin de calculer les dissimilarités dans l'espace des attributs n'est peut-être pas le plus pertinent. Les résultats mitigés que nous observons dans certains cas peuvent être dus à des limitations d'UET.

Afin de confirmer ou infirmer certaines de ces hypothèses, nous avons comparé les modularités des partitions obtenues en utilisant des forêts mononessences, multiessences, et enfin en utilisant les étiquettes fournies avec les jeux de données. Les résultats obtenus semblent indiquer qu'effectivement, les forêts monoessences donnent de meilleurs résultats que les forêts multiessences. Fait intéressant, les résultats obtenus en utilisant UET uniquement sur les graphes sont parfois meilleurs qu'en utilisant directement la «vérité terrain» contenue dans les étiquettes. Cela peut être dû au fait que dans certains cas, même cette *vérité* ne correspond pas toujours à l'organisation réelle des noeuds dans le graphe.

Enfin, une application de l'approche sur des graphes disponibles dans le cadre du projet nous ont permis de montrer que l'algorithme développé ainsi que son implémentation sont prêts à être utilisés dans le cadre pratique. Nous avons en effet montré qu'un clustering basé sur les dissimilarités obtenues permettent de mettre en évidence des clusters de médicaments intéressants, sur la base des relations présentes dans notre base de connaissances.

Il est à noter que les résultats que nous avons obtenus dépendent du choix d'un algorithme de clustering spécifique. En effet, UET n'est pas un algorithme de clustering, mais une méthode permettant de quantifier une dissimilarité entre noeuds. Il est ainsi nécessaire d'y appliquer un algorithme de clustering une fois les dissimilarités obtenues. Ceci constitue un atout de l'approche présentée dans cette contribution. En effet, la tâche de recherche de communautés peut être réalisée par plusieurs algorithmes, en tirant profit de leurs forces et faiblesses respectives.

6

Conclusions et perspectives

Sommaire

6.1 Conclusion et perspectives

Nous avons pu voir que l'extraction de connaissances à partir de données est une tâche complexe selon plusieurs facteurs. L'un des éléments rendant cette tâche complexe est la multiplicité des formes que les données peuvent prendre en entrée. Les sources étant de plus en plus diverses, ces données peuvent prendre la forme de graphes, de tableaux, voire de texte libre. Pour complexifier l'ensemble, ces représentations peuvent encore être sous différentes formes. On retrouve par exemple des graphes attribués, des données structurées où les types ne sont pas homogènes, etc.

De nombreuses méthodes de fouille –celles principalement utilisées– ne sont applicables qu'à un ensemble restreint de types de données. Il n'est d'ailleurs pas rare que cet ensemble ne contienne qu'un unique élément. Il s'agit de là de l'un des (nombreux) problèmes auxquels sont confrontées les personnes chargées d'appliquer des méthodes de fouille de données, afin d'en extraire des connaissances.

Bien sûr, il serait utopique que de chercher une méthode unique permettant de surpasser l'ensemble des autres méthodes dans tous les cas. Cela n'est d'ailleurs pas possible, comme nous avons pu le voir avec le théorème du *no free lunch*. La question que nous nous sommes posée est donc plutôt de savoir s'il était possible de développer une approche modulaire permettant l'application d'un certain nombre d'algorithmes déjà existants et éprouvés sur différents types de données en entrée. Ces derniers types ne sont pas choisis au hasard. Il s'agit ici de données représentées sous forme de vecteurs d'attributs et sous forme de graphes, dans la mesure où il s'agit des types de données pouvant être rencontrées dans le cas de la recherche biomédicale, et plus particulièrement dans le cadre du projet FIGHT-HF.

Nous avons fait le choix d'explorer au cours de cette thèse des approches basées sur les arbres aléatoires afin de mener à bien cette tâche. Ces structures, bien connues et étudiées depuis

plusieurs décennies, ont dévoilé étude après étude des performances particulièrement intéressantes – voire étonnantes – dans le cadre de la fouille de données. Bien qu'initialement utilisées à des fins d'apprentissage supervisé, notamment dans le très célèbre algorithme des forêts d'arbres aléatoires, les arbres aléatoires peuvent aussi l'être de manière plus surprenante dans le cadre non supervisé. L'approche est aussi retrouvée dans le cadre, peut-être plus confidentiel, de la recherche d'anomalies. Il s'agit donc bien là d'une approche ubiquitaire, pouvant être utilisée *à façon* en fonction du cadre de l'étude. Il n'est d'ailleurs pas étonnant de les trouver dans de nombreuses études, tant leurs propriétés sont remarquables. Leur caractère stochastique, bien que pouvant poser problème dans certains cas, leur permet notamment d'être appliquées sur de grands jeux de données. De plus, l'interprétabilité des modèles obtenus est rendue très simple. Enfin, les arbres aléatoires peuvent être appliqués à des données de différents types – numériques et qualitatives.

C'est en particulier cette propriété qui nous a poussés à explorer les approches à base d'arbres aléatoires, et à en proposer de nouvelles. En effet, l'application de méthodes de clustering à des données dites hétérogènes peut s'avérer complexe en pratique, pour des raisons liées aux algorithmes de clustering eux-mêmes, ou liées aux mesures de dissimilarité utilisées dans ces derniers. Le travail de Tao Shi et Steven Horvath, publié en 2006, a ouvert la voie à l'utilisation d'arbres aléatoires dans le cadre non supervisé. Leur approche constitue cependant les balbutiements du domaine, tant les limitations – notamment en termes de temps de calcul – sont fortes. De plus, ces derniers se sont principalement focalisés sur le cadre des données continues, ce qui constituait le coeur de travaux.

C'est pourquoi nous nous sommes penchés sur le sujet, et avons proposé dans [DCST18], puis dans [DCST20] une nouvelle méthode, basée sur les arbres extrêmement aléatoires, en suivant la voie ouverte par les auteurs cités précédemment. En utilisant les arbres extrêmement aléatoires en tant que structure de base à la forêt, nous proposons une approche plus performante que les forêts d'arbres aléatoires initiales. Nous avons de plus proposé une approche permettant la prise en compte des données hétérogènes. Nos évaluations ont permis d'affirmer la caractère prometteur de l'approche, au vu des bons résultats que nous avons pu obtenir. En effet, ces évaluations nous ont montré d'une part que les UET permettent d'obtenir des dissimilarités pertinentes, permettant le clustering de données, et d'autre part que ces clusterings sont de bonne qualité, que ce soit sur des données synthétiques ou réelles. De plus, nous avons pu montrer que notre proposition donne de meilleurs résultats que la méthode initiale des URF. Enfin, une application pratique dans [PDD$^+$21] sur des données cliniques a permis la mise en évidence de clusters spécifiques et différents de ceux étant obtenus par d'autres mesures plus classiques. Ces clusters peuvent s'expliquer cliniquement, et sont donc en ce sens intéressants. Ce travail, réalisé en collaborations avec d'autres chercheurs du projet, a permis aussi d'effectuer une évaluation tierce d'UET en les comparant à d'autres approches de clustering et d'autres mesures. Ces résultats sont prometteurs et nous ont montré que notre approche donne de bons résultats, notamment face à la distance de Gower, souvent retrouvée dans la littérature lorsqu'il s'agit de données hétérogènes. Bien sûr, certains points restent à éclaircir. En particulier, nous savons que les hyperparamètres de la méthode influencent drastiquement les résultats. C'est pour cette raison que nous avons mis en oeuvre une procédure d'optimisation automatique, permettant d'évaluer quelles valeurs de ces paramètres permettent d'obtenir les meilleurs résultats. Une piste de recherche est cependant ouverte ici, dans la mesure où, bien que tout à fait possible, l'application de notre approche d'optimisation sur de gros jeux de données peut être potentiellement chronophage. Des approches

plus poussées, telles que l'optimisation bayésienne, pourraient être envisagées ici, mais n'ont pas été explorées au cours de ce travail.

Nous nous sommes par la suite concentrés sur l'application de cette approche aux graphes, toujours dans l'objectif de développer une approche applicable à de multiples types de données. La méthode que nous proposons dans [DCST19] consiste à construire des UET sur des graphes, en considérant les noeuds de ces derniers comme des objets à séparer. La difficulté ici a été de définir une fonction permettant de couper l'espace des noeuds, afin de pouvoir construire les arbres. Notre proposition a été ici de considérer la notion voisinage comme primitive à cette partition. Il s'agit là d'une vision sortant des sentiers battus dans les arbres aléatoires, fonctionnant généralement sur des données numériques. Cependant, nos évaluations nous ont montré que cette approche n'est pas si farfelue, les dissimilarités obtenues permettant effectivement d'appliquer avec succès des algorithmes de clustering classiques sur les graphes. Nous avons aussi montré qu'il est possible de construire d'UET de différente nature, permettant par exemple de calculer des dissimilarités entre noeuds d'un graphe attribué. Nous avons pu montrer que cette proposition est là encore très prometteuse, avec des résultats très intéressants, à la fois sur des jeux de données d'évaluation que sur des jeux de données *réels* extraits d'une base de connaissance locale. Sur cette dernière, nous avons mis en évidence des groupes de substances pouvant être utilisées en tant que médicament, ayant des similarités mécanistiques. Bien que présenté comme *mini-étude*, ces résultats montrent le caractère pratique de notre approche ainsi que sa capacité à donner des groupes homogènes et pertinents, notamment dans le cadre de graphes de données biomédicales.

Bien sûr, de nombreuses études complémentaires restent à être effectuées. Nous avons ici ouvert la voie et montré la faisabilité de cette approche. Des études plus poussées sont nécessaires, tant sur le plan théorique que pratique. En particulier, bien que nous avons publié les implémentations qui sont donc maintenant disponibles publiquement, certaines améliorations restent nécessaires. Nous avons pu voir en particulier que les résultats obtenus dans le cas des graphes attribués, et bien que prometteurs et rassurants sur le bien-fondé de la méthode, sont dans quelques cas de faible qualité. Cela peut être dû à de nombreux facteurs, tels que la pertinence de l'information apportée par les attributs, comme nous avons pu le constater. Nous pourrions ici envisager une méthode de quantification préalable de l'information apportée par cahque facette des données, passant par exemple par une analyse des dissimilarités obtenues, ou encore de la distribution des valeurs. Ces résultats peuvent aussi être dûs au mode de calcul des dissimilarités sur l'une ou l'autre des représentations de données. Rappelons ici que nous avons utilisé la méthode de calcul fréquentiste avec les arbres spécifiques aux attributs des noeuds. D'autres études, notamment utilisant l'approche à base de masse sur les données numériques pourraient mettre en évidence une limitation de l'approche fréquentiste. Cela pourrait permettre par ailleurs le calcul des dissimilarités *en une fois*, la même approche de calcul de dissimilarité étant utilisée sur l'ensemble des arbres de la forêt.

Pour résumer en quelques mots les contributions de cette thèse, nous avons ici montré que les arbres aléatoires sont des structures intéressantes dont les applications peuvent être variées, et peuvent aller au-delà des usages classiques sur des données numériques. En montrant qu'une simple modification de la fonction de séparation au niveau des feuilles peut permettre leur application sur des données non numériques, nous espérons qu'il s'agit ici que des pas dans leur utilisation dans des cadres moins courants. Nous espérons par ailleurs que ce travail permettra

de réhabiliter l'utilisation des arbres aléatoires dans le cadre non supervisé.

À ce propos, nous avons pu montrer au fil de ce document que différentes méthodes de calcul de dissimilarité à partir des arbres peuvent être définies. Nous avons ici présenté deux approches, dites à base de densité et fréquentiste, mais d'autres méthodes pourraient éventuellement être proposées. Nous espérons que les résultats présentés dans ces travaux ouvriront la voie à de nombreuses autres approches exploratoires basées sur les arbres aléatoires.

Bibliographie

[AK10] Hanan G Ayad and Mohamed S Kamel. On voting-based consensus of cluster ensembles. *Pattern Recognition*, 43(5) :1943–1953, 2010.

[AW19] Tahani Alqurashi and Wenjia Wang. Clustering ensemble method. *International Journal of Machine Learning and Cybernetics*, 10(6) :1227–1246, 2019.

[BA96] Ronald J Brachman and Tej Anand. The process of knowledge discovery in databases. In *Advances in knowledge discovery and data mining*, pages 37–57. 1996.

[BA99] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439) :509–512, 1999.

[BAK18] Danilo Bzdok, Naomi Altman, and Martin Krzywinski. Points of significance : statistics versus machine learning, 2018.

[BCMM15] Cécile Bothorel, Juan David Cruz, Matteo Magnani, and Barbora Micenkova. Clustering attributed graphs : models, measures and methods. *arXiv preprint arXiv :1501.01676*, 2015.

[BCN+17] Eric Breck, Shanqing Cai, Eric Nielsen, Michael Salib, and D Sculley. The ml test score : A rubric for ml production readiness and technical debt reduction. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1123–1132. IEEE, 2017.

[BDGS05] Arindam Banerjee, Inderjit S Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6(Sep) :1345–1382, 2005.

[Ber44] Joseph Berkson. Application of the logistic function to bio-assay. *Journal of the American statistical association*, 39(227) :357–365, 1944.

[BFSO84] Leo Breiman, J Friedman, CJ Stone, and RA Olshen. Classification algorithms and regression trees. *Classification and regression trees*, 15(2) :246, 1984.

[BGLL08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics : theory and experiment*, 2008(10) :P10008, 2008.

[BGV92] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[BLA+09] Vanessa Buchholz, Martin Leuwer, Jörg Ahrens, Nilufar Foadi, Klaus Krampfl, and Gertrud Haeseler. Topical antiseptics for the treatment of sore

throat block voltage-gated neuronal sodium channels in a local anaesthetic-like manner. *Naunyn-Schmiedeberg's archives of pharmacology*, 380(2) :161–168, 2009.

[Boj18] Stephan Bojchevski, Aleksandar and Günnemann. Bayesian Robust Attributed Graph Clustering : Joint Learning of Partial Anomalies and Group Structure. 2018.

[Bre01] Leo Breiman. Random Forests. *Machine Learning*, 45(1) :5–32, October 2001.

[Cas19] Casimir. Illustrating the random forest algorithm in tikz. Tex Stack Exchange, 2019. URL :https ://tex.stackexchange.com/questions/503883/illustrating-the-random-forest-algorithm-in-tikz (version : 2021-06-21).

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection : A survey. *ACM computing surveys (CSUR)*, 41(3) :1–58, 2009.

[CBP13] Juan David Cruz, Cécile Bothorel, and François Poulet. Integrating heterogeneous information within a social network for detecting communities. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pages 1453–1454. IEEE, 2013.

[Chi11] Chire. Illustration de dbscan, 2011.

[Chu19] Petr Chunaev. Community detection in node-attributed social networks : a survey. *arXiv preprint arXiv :1912.09816*, 2019.

[CK01] Anne Condon and Richard M Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures & Algorithms*, 18(2) :116–140, 2001.

[CL14] Michael R Chernick and Robert A LaBudde. *An introduction to bootstrap methods with applications to R*. John Wiley & Sons, 2014.

[CLEZG12] David Combe, Christine Largeron, Elöd Egyed-Zsigmond, and Mathias Géry. Combining relations and text in scientific network clustering. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 1248–1253. IEEE, 2012.

[CLGEZ15] David Combe, Christine Largeron, Mathias Géry, and Előd Egyed-Zsigmond. I-louvain : An attributed graph clustering method. In *International Symposium on Intelligent Data Analysis*, pages 181–192. Springer, 2015.

[CMS13] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[CNM04] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6) :066111, 2004.

[CNP06] Gianfranco Chicco, Roberto Napoli, and Federico Piglione. Comparisons among clustering techniques for electricity customer classification. *IEEE Transactions on Power Systems*, 21(2) :933–940, 2006.

[CS14] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1) :16–28, 2014.

[CS15] Vinay Chaorasiya and A Shrivastava. A survey on big data : Techniques and technologies. *International Journal of Research and Development in Applied Science and Engineering*, 8(1) :1–4, 2015.

[CY06] Jingchun Chen and Bo Yuan. Detecting functional modules in the yeast protein–protein interaction network. *Bioinformatics*, 22(18) :2283–2290, 2006.

[Cyc08] Cyc. svm max sep hyperplane with margin. `https://commons.wikimedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png`, 2008.

[CZY11] Hong Cheng, Yang Zhou, and Jeffrey Xu Yu. Clustering large attributed graphs : A balance between structural and attribute similarities. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(2) :1–33, 2011.

[DB79] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2) :224–227, 1979.

[DCST18] Kevin Dalleau, Miguel Couceiro, and Malika Smaïl-Tabbone. Unsupervised extremely randomized trees. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 478–489. Springer, 2018.

[DCST19] Kevin Dalleau, Miguel Couceiro, and Malika Smail-Tabbone. Computing vertex-vertex dissimilarities using random trees : Application to clustering in graphs. In *International Symposium on Intelligent Data Analysis*, pages 132–144. Springer, 2019.

[DCST20] Kevin Dalleau, Miguel Couceiro, and Malika Smaïl-Tabbone. Unsupervised extra trees : a stochastic approach to compute similarities in heterogeneous data. *Int. J. Data Sci. Anal.*, 9(4) :447–459, 2020.

[DD14] Michel Marie Deza and Elena Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[Die00] Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees : Bagging, boosting, and randomization. *Machine learning*, 40(2) :139–157, 2000.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society : Series B (Methodological)*, 39(1) :1–22, 1977.

[Dom12] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10) :78–87, 2012.

[DSH20] Alexis Derumigny and Johannes Schmidt-Hieber. On lower bounds for the bias-variance trade-off. *arXiv preprint arXiv :2006.00278*, 2020.

[dSPFR21] Rafael Gaspar de Sousa, Sarajane Marques Peres, Marcelo Fantinato, and Hajo Alexander Reijers. Concept drift detection and localization in process mining : an integrated and efficient approach enabled by trace clustering. In Chih-Cheng Hung, Jiman Hong, Alessio Bechini, and Eunjee Song, editors, *SAC '21 : The 36th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, Republic of Korea, March 22-26, 2021*, pages 364–373. ACM, 2021.

[DV12] TA Dang and Emmanuel Viennet. Community detection based on structural and attribute similarities. In *International conference on digital society (icds)*, pages 7–12, 2012.

[EA10] Haytham Elghazel and Alex Aussem. Feature selection for unsupervised learning using random cluster ensembles. In *2010 IEEE International Conference on Data Mining*, pages 168–175. IEEE, 2010.

[EB94] Martin G Everett and Stephen P Borgatti. Regular equivalence : General theory. *Journal of mathematical sociology*, 19(1) :29–52, 1994.

[EKS+96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[EMSK97] Floriana Esposito, Donato Malerba, Giovanni Semeraro, and J Kay. A comparative analysis of methods for pruning decision trees. *IEEE transactions on pattern analysis and machine intelligence*, 19(5) :476–491, 1997.

[ER60] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1) :17–60, 1960.

[FB04] Xiaoli Zhang Fern and Carla E Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of the twenty-first international conference on Machine learning*, page 36, 2004.

[FDM+18] Ferreira J.P., Duarte K., McMurray J.J.V., Pitt B., van Veldhuisen D.J., Vincent J., Ahmad T., Tromp J., Rossignol P., and Zannad F. Data-Driven Approach to Identify Subgroups of Heart Failure With Reduced Ejection Fraction Patients With Different Prognoses and Aldosterone Antagonist Response Patterns. *Circulation : Heart Failure*, 11(7) :e004926, July 2018.

[FGKB18] Issam Falih, Nistor Grozavu, Rushed Kanawati, and Younès Bennani. Community detection in attributed network. In *Companion Proceedings of the The Web Conference 2018*, pages 1299–1306, 2018.

[Fla12] Peter Flach. *Machine learning : the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

[For10] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5) :75–174, 2010.

[FPSS96] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3) :37–37, 1996.

[Fre77] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[Fri98] Jerome H Friedman. Data mining and statistics : What's the connection ? *Computing science and statistics*, 29(1) :3–9, 1998.

[Fri06] Jerome H Friedman. Recent advances in predictive (machine) learning. *Journal of classification*, 23(2) :175–197, 2006.

[Gen10] Robin Genuer. *Random Forests : elements of theory, variable selection and applications*. Theses, Université Paris Sud - Paris XI, November 2010.

[GEW06] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1) :3–42, 2006.

[GFBS10] Stephan Gunnemann, Ines Farber, Brigitte Boden, and Thomas Seidl. Subspace clustering meets dense subgraph mining : A synthesis of two paradigms. In *2010 IEEE International Conference on Data Mining*, pages 845–850. IEEE, 2010.

[GGCM08] Mike Gashler, Christophe Giraud-Carrier, and Tony Martinez. Decision tree ensemble : Small heterogeneous is better than large homogeneous. In *2008 Seventh International Conference on Machine Learning and Applications*, pages 900–905. IEEE, 2008.

[GH87] GA Gintant and BF Hoffman. The role of local anesthetic effects in the actions of antiarrhythmic drugs. *Local Anesthetics*, pages 213–251, 1987.

[GJ03] K. Grąbczewski and N. Jankowski. Transformations of symbolic data for continuous data oriented models. In *Artificial Neural Networks and Neural Information Processing—ICANN/ICONIP 2003*, pages 359–366. Springer, 2003.

[GJ+10] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http ://eigen.tuxfamily.org, 2010.

[GL16] Aditya Grover and Jure Leskovec. node2vec : Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[Gow71] J.C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, pages 857–871, 1971.

[GP17] Robin Genuer and Jean-Michel Poggi. Arbres cart et forêts aléatoires, importance et sélection de variables. 2017. hal-01387654v2.

[HA85] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1) :193–218, 1985.

[HBMB05] Graeme S Halford, Rosemary Baker, Julie E McCredden, and John D Bain. How many variables can humans process ? *Psychological science*, 16(1) :70–76, 2005.

[HDMB21] Melanie Hackl, Suparno Datta, Riccardo Miotto, and Erwin Bottinger. Unsupervised learning to subphenotype heart failure patients from electronic health records. In *International Conference on Artificial Intelligence in Medicine*, pages 219–228. Springer, 2021.

[HK65] Yu-Chi Ho and RL Kashyap. An algorithm for linear inequalities and its applications. *IEEE Transactions on Electronic Computers*, (5) :683–688, 1965.

[HKK05] Julia Handl, Joshua Knowles, and Douglas B Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15) :3201–3212, 2005.

[HLL83] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels : First steps. *Social networks*, 5(2) :109–137, 1983.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning : data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[Hua98] Z. Huang. Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Mining and Knowledge Discovery*, 2(3) :283–304, September 1998.

[JHCL18] S. Jian, L. Hu, L. Cao, and K. Lu. Metric-based auto-instructor for learning mixed data representation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[JL13] George H John and Pat Langley. Estimating continuous distributions in bayesian classifiers. *arXiv preprint arXiv :1302.4964*, 2013.

[JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 6. Springer, 2013.

[JZZW19] Yuting Jia, Qinqin Zhang, Weinan Zhang, and Xinbing Wang. Community-gan : Community detection with generative adversarial nets. In *The World Wide Web Conference*, pages 784–794, 2019.

[Kin19] Mital Kinderkhedia. Learning representations of graph data–a survey. *arXiv preprint arXiv :1906.02989*, 2019.

[KL02] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*, volume 2002, pages 315–22, 2002.

[Kos19] Sven Kosub. A note on the triangle inequality for the jaccard distance. *Pattern Recognition Letters*, 120 :36–38, 2019.

[KR09] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data : an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[Kru78] C. Krumhansl. Concerning the applicability of geometric models to similarity data : The interrelationship between similarity and spatial density. *Psychological Review*, 85 :450–463, 1978.

[Kva87] Tarald O Kvalseth. Entropy and correlation : Some comments. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3) :517–519, 1987.

[KW52] William H Kruskal and W Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260) :583–621, 1952.

[KW16] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv :1611.07308*, 2016.

[Les97] Michael Lesk. How much information is there in the world ?, 1997. http ://www.lesk.com/mlesk/ksg97/ksg.html.

[LFR08] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4) :046110, 2008.

[LGT08] Zhenqiu Liu, Zhongmin Guo, and Ming Tan. Constructing tumor progression pathways and biomarker discovery with fuzzy kernel kmeans and dna methylation data. *Cancer informatics*, 6 :117693510800600007, 2008.

[LKF07] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution : Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1) :2–es, 2007.

[LLZG20] Chang Liu, Christine Largeron, Osmar R Zaïane, and Shiva Zamani Gharaghooshi. A late-fusion approach to community detection in attributed networks. In *International Symposium on Intelligent Data Analysis*, pages 300–312. Springer, 2020.

[LMBZ17] Christine Largeron, Pierre-Nicolas Mougel, Oualid Benyahia, and Osmar R Zaïane. Dancer : dynamic attributed networks with community structure generation. *Knowledge and Information Systems*, 53(1) :109–151, 2017.

[LMRZ15] Christine Largeron, Pierre-Nicolas Mougel, Reihaneh Rabbany, and Osmar R Zaïane. Generating attributed networks with communities. *PloS one*, 10(4) :e0122777, 2015.

[Lou14] Gilles Louppe. *Understanding random forests : From theory to practice*. PhD thesis, 2014.

[LP49] R Duncan Luce and Albert D Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2) :95–116, 1949.

[LTZ08] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[LW71] Francois Lorrain and Harrison C White. Structural equivalence of individuals in social networks. *The Journal of mathematical sociology*, 1(1) :49–80, 1971.

[M+67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[Mah36] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. volume 2, pages 49–55, 1936.

[MBC16] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM computing surveys (CSUR)*, 49(4) :1–33, 2016.

[MGM16] Saskia Metzler, Stephan Günnemann, and Pauli Miettinen. Hyperbolae are no hyperbole : Modelling communities that are not cliques. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 330–339. IEEE, 2016.

[MH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov) :2579–2605, 2008.

[MNON17] Hidetoshi Miyazaki, Teppei Nakamura, Kohji Ohki, and Katsuya Nagai. Effects of the bioactive peptides ile-pro-pro and val-pro-pro upon autonomic neurotransmission and blood pressure in spontaneously hypertensive rats. *Autonomic Neuroscience*, 208 :88–92, 2017.

[Moč75] Jonas Močkus. On bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pages 400–404. Springer, 1975.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[MSLC01] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather : Homophily in social networks. *Annual review of sociology*, 27(1) :415–444, 2001.

[MTO18] Seiji Maekawa, Koh Takeuch, and Makoto Onizuka. Non-linear Attributed Graph Clustering by Symmetric NMF with PU Learning. *arXiv preprint arXiv :1810.00946*, 2018.

[MW17] Marco Maisenbacher and Matthias Weidlich. Handling concept drift in predictive process monitoring. In Xiaoqing (Frank) Liu and Umesh Bellur, editors, *2017 IEEE International Conference on Services Computing, SCC 2017,*

Honolulu, HI, USA, June 25-30, 2017, pages 1–8. IEEE Computer Society, 2017.

[NAJ03] Jennifer Neville, Micah Adler, and David Jensen. Clustering relational data using attribute and link information. In *Proceedings of the text mining and link analysis workshop, 18th international joint conference on artificial intelligence*, pages 9–15, 2003.

[New06] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23) :8577–8582, 2006.

[NG04] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2) :026113, 2004.

[NRG17] Haiqiang Niu, Emma Reeves, and Peter Gerstoft. Source localization in an ocean waveguide using supervised machine learning. *The Journal of the Acoustical Society of America*, 142(3) :1176–1188, 2017.

[NS01] Krzysztof Nowicki and Tom A B Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American statistical association*, 96(455) :1077–1087, 2001.

[Ope08] OpenMP Architecture Review Board. OpenMP application program interface version 3.0, May 2008.

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk : Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[PBR+08] Bertram Pitt, George Bakris, Luis M Ruilope, Lorenzo DiCarlo, and Robin Mukherjee. Serum potassium and clinical outcomes in the eplerenone post–acute myocardial infarction heart failure efficacy and survival study (ephesus). *Circulation*, 118(16) :1643–1650, 2008.

[PDD+21] Gregoire Preud'homme, Kevin Duarte, Kevin Dalleau, Claire Lacomblez, Emmanuel Bresso, Malika Smaïl-Tabbone, Miguel Couceiro, Marie-Dominique Devignes, Masatake Kobayashi, Olivier Huttin, et al. Head-to-head comparison of clustering methods for heterogeneous data : a simulation-driven benchmark. *Scientific reports*, 11(1) :1–14, 2021.

[PLL99] José Manuel Pena, Jose Antonio Lozano, and Pedro Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern recognition letters*, 20(10) :1027–1040, 1999.

[PMB13] Marcus B Perry, Gregory V Michaelson, and M Alan Ballard. On the statistical detection of clusters in undirected networks. *Computational statistics & data analysis*, 68 :170–189, 2013.

[Pod99] János Podani. Extending gower's general coefficient of similarity to ordinal characters. *Taxon*, 48(2) :331–340, 1999.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn : Machine learning in python. *the Journal of machine Learning research*, 12 :2825–2830, 2011.

[PYC+15] Ruilin Pan, Tingsheng Yang, Jianhua Cao, Ke Lu, and Zhanchao Zhang. Missing data imputation by k nearest neighbours based on grey relational structure and mutual information. *Applied Intelligence*, 43(3) :614–632, 2015.

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1) :81–106, 1986.

[Qui87] J Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3) :221–234, 1987.

[Qui93] J Ross Quinlan. C4. 5 : Programming for machine learning. *Morgan Kauffmann*, 38 :48, 1993.

[Ran71] W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336) :846–850, 1971.

[RKSR02] P Krishna Reddy, Masaru Kitsuregawa, P Sreekanth, and S Srinivasa Rao. A graph based approach to extract a neighborhood customer community for collaborative filtering. In *International Workshop on Databases in Networked Information Systems*, pages 188–200. Springer, 2002.

[Ros61] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[Rou87] Peter J Rousseeuw. Silhouettes : a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20 :53–65, 1987.

[Sam03] Rajan Sambandam. Cluster analysis gets complicated. *Marketing Research*, 15(1) :16–21, 2003.

[SC08] Karsten Steinhaeuser and Nitesh V Chawla. Community detection in a large real-world social network. In *Social computing, behavioral modeling, and prediction*, pages 168–175. Springer, 2008.

[Sch07] Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1) :27–64, 2007.

[Seg04] Mark R Segal. Machine learning benchmarks and random forest regression. 2004.

[SG02] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec) :583–617, 2002.

[SH06] T. Shi and S. Horvath. Unsupervised learning with random forest predictors. *Journal of Computational and Graphical Statistics*, 15(1) :118–138, 2006.

[She00] Colin Shearer. The crisp-dm model : the new blueprint for data mining. *Journal of data warehousing*, 5(4) :13–22, 2000.

[SKD+14] János Szüle, Dániel Kondor, László Dobos, István Csabai, and Gábor Vattay. Lost in the city : Revisiting milgram's experiment in the age of social networks. *PloS one*, 9 :e111973, 11 2014.

[SKS+15] Sanjiv J Shah, Daniel H Katz, Senthil Selvaraj, Michael A Burke, Clyde W Yancy, Mihai Gheorghiade, Robert O Bonow, Chiang-Ching Huang, and Rahul C Deo. Phenomapping for novel classification of heart failure with preserved ejection fraction. *Circulation*, 131(3) :269–279, 2015.

[SRA19] Bishnu Sarker, David W Ritchie, and Sabeur Aridhi. Functional annotation of proteins using domain embedding based sequence classification. In *KDIR 2019-11th International Conference on Knowledge Discovery and Information Retrieval*, 2019.

[SRPMLCCdlV11] Esther-Lydia Silva-Ramírez, Rafael Pino-Mejías, Manuel López-Coello, and María-Dolores Cubiles-de-la Vega. Missing value imputation on missing completely at random data using multilayer perceptrons. *Neural Networks*, 24(1) :121–129, 2011.

[SSB+05] Tao Shi, David Seligson, Arie S Belldegrun, Aarno Palotie, and Steve Horvath. Tumor classification by tissue microarray profiling : random forest clustering applied to renal cell carcinoma. *Modern Pathology*, 18(4) :547–557, 2005.

[Sur04] James Surowiecki. *The Wisdom of Crowds : Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*. Doubleday, May 2004.

[TEK+11] Anu M Turpeinen, Pauliina I Ehlers, Anne S Kivimäki, Salme Järvenpää, Iris Filler, Erol Wiegert, Eberhard Jähnchen, Heikki Vapaatalo, Riitta Korpela, and Frank Wagner. Ile-pro-pro and val-pro-pro tripeptide-containing milk product has acute blood pressure lowering effects in mildly hypertensive subjects. *Clinical and Experimental Hypertension*, 33(6) :388–396, 2011.

[Tve77] Amos Tversky. Features of similarity. *Psychological review*, 84(4) :327, 1977.

[Twa09] Bhekisipho Twala. An empirical comparison of techniques for handling incomplete data using decision trees. *Applied Artificial Intelligence*, 23(5) :373–405, 2009.

[TWvE19] Vincent A Traag, Ludo Waltman, and Nees Jan van Eck. From louvain to leiden : guaranteeing well-connected communities. *Scientific reports*, 9(1) :1–12, 2019.

[TZC+16] Kai Ming Ting, Ye Zhu, Mark Carman, Yue Zhu, and Zhi-Hua Zhou. Overcoming key weaknesses of distance-based neighbourhood methods using a data dependent dissimilarity measure. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1205–1214, 2016.

[VCL+09] Ravishankar R Vallabhajosyula, Deboki Chakravarti, Samina Lutfeali, Animesh Ray, and Alpan Raval. Identifying hubs in protein interaction networks. *PloS one*, 4(4) :e5344, 2009.

[VD00] Stijn Marinus Van Dongen. *Graph clustering by flow simulation*. PhD thesis, 2000.

[vdH15] J. van den Hoven. Clustering with optimised weights for Gower's metric. *Netherlands : University of Amsterdam*, 2015.

[VEB10] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison : Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11 :2837–2854, 2010.

[VPRS11] Sandro Vega-Pons and José Ruiz-Shulcloper. A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(03) :337–372, 2011.

[VVOCA13] Nathalie Villa-Vialaneix, Madalina Olteanu, and Christine Cierco-Ayrolles. Carte auto-organisatrice pour graphes étiquetés. In *Atelier Fouilles de Grands Graphes (FGG)-EGC'2013*, pages Article–numéro, 2013.

[Wat04] Duncan J Watts. *Small worlds : the dynamics of networks between order and randomness*, volume 9. Princeton university press, 2004.

[WH00] Rüdiger Wirth and Jochen Hipp. Crisp-dm : Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, pages 29–39. Springer-Verlag London, UK, 2000.

[Wil02] Rudolf Wille. Why can concept lattices support knowledge discovery in databases ? *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3) :81–92, 2002.

[WM97] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1) :67–82, 1997.

[WM99] Lance Wilson and Steven Martin. Benzyl alcohol as an alternative local anesthetic. *Annals of emergency medicine*, 33(5) :495–499, 1999.

[WPH+19] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. Attributed graph clustering : A deep attentional embedding approach. *arXiv preprint arXiv :1906.06532*, 2019.

[WT07] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 1275–1276, 2007.

[WVO12] Marco Wiering and Martijn Van Otterlo. Reinforcement learning. *Adaptation, learning, and optimization*, 12 :3, 2012.

[Yao03] YY Yao. Information-theoretic measures for knowledge discovery and data mining. In *Entropy measures, maximum entropy principle and emerging applications*, pages 115–136. Springer, 2003.

[YBLG17] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 555–564, 2017.

[YML13] Jaewon Yang, Julian McAuley, and Jure Leskovec. Community detection in networks with node attributes. In *2013 IEEE 13th International Conference on Data Mining*, pages 1151–1156. IEEE, 2013.

[ZCY09] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, 2(1) :718–729, 2009.

[Zho12] Zhi-Hua Zhou. *Ensemble methods : foundations and algorithms*. CRC press, 2012.

[Zhu05] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.

[ZKL+06] F Zannad, M Kessler, P Lehert, JP Grünfeld, C Thuilliez, A Leizorovicz, P Lechat, Fosidial Investigators, et al. Prevention of cardiovascular events in end-stage renal disease : results of a randomized trial of fosinopril and implications for future studies. *Kidney international*, 70(7) :1318–1324, 2006.

[ZMD+18] Zhongxing Zhang, Geert Mayer, Yves Dauvilliers, Giuseppe Plazzi, Fabio Pizza, Rolf Fronczek, Joan Santamaria, Markku Partinen, Sebastiaan Overeem, Rosa Peraita-Adrados, et al. Exploring the clinical features of narcolepsy type 1 versus narcolepsy type 2 from european narcolepsy network database with machine learning. *Scientific reports*, 8(1) :1–11, 2018.

[ZMKS15] Jonas Zierer, Cristina Menni, Gabi Kastenmüller, and Tim D Spector. Integration of 'omics' data in aging research : from biomarkers to systems biology. *Aging cell*, 14(6) :933–944, 2015.

[ZXZ+18] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 95–103, 2018.

[ZYZZ18] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Network representation learning : A survey. *IEEE transactions on Big Data*, 2018.

Résumé

La notion de distance, et plus généralement de dissimilarité, est une notion importante en fouille de données, tout particulièrement dans les approches non supervisées. Les algorithmes de cette classe de méthodes visant à regrouper de manière homogène des objets, nombre d'entre eux s'appuient sur une notion de dissimilarité, afin de quantifier la proximité entre objets. Le choix des algorithmes ainsi que celui des dissimilarités n'est cependant pas trivial. Plusieurs éléments peuvent motiver ces choix, tels que le type de données – données homogènes ou non –, leur représentation –vecteurs d'attributs, graphes –, ou encore certaines de leurs caractéristiques –fortement corrélées, bruitées, etc. –. Bien que de nombreuses mesures existent, leur choix peut devenir complexe dans certains cadres spécifiques. Ceci entraîne une complexité supplémentaire dans les tâches d'exploration et de fouille des données. Nous présentons dans cette thèse une nouvelle approche permettant le calcul de dissimilarités, basée sur des arbres aléatoires. Il s'agit d'une approche originale dont nous montrons plusieurs avantages, parmi lesquels l'on retrouve une grande versatilité. En effet, par le biais de différents modules de calcul de dissimilarités que nous accolons à la méthode, il devient possible de l'appliquer dans divers cadres. Nous présentons notamment dans ce document deux modules, permettant le calcul de dissimilarités — et, *in fine*, le clustering — sur des données structurées sous forme de vecteur d'attributs, et sur des données sous forme de graphes. Nous discutons des résultats très prometteurs obtenus par cette approche, ainsi que des nombreuses perspectives ouvertes par cette dernière, telle que le calcul de dissimilarité dans le cadre des graphes attribués, par le biais d'une approche unifiée.

Mots-clés: Distance, Dissimilarité, Arbres aléatoires, Méthodes stochastiques, Clustering, Clustering de graphes.

Abstract

The notion of distance, and more generally of dissimilarity, is an important one in data mining, especially in unsupervised approaches. The algorithms belonging to this class of methods aim at grouping objects in an homogeneous way, and many of them rely on a notion of dissimilarity, in order to quantify the proximity between objects. The choice of algorithms as well as that of dissimilarities is not trivial. Several elements can motivate these choices, such as the type of data – homogeneous data or not –, their representation – feature vectors, graphs –, or some of their characteristics – highly correlated, noisy, etc. –. Although many measures exist, their choice can become complex in some specific settings. This leads to additional complexity in data mining tasks. In this thesis, we present a new approach for computing dissimilarities based on random trees. It is an original approach, which has several advantages such as a great versatility. Indeed, using different dissimilarity calculation modules that we can plug to the method, it becomes possible to apply it in various settings. In particular, we present in this document two modules, enabling the computation of dissimilarities - and, *in fine*, clustering - on data structured as feature vectors, and on data in the form of graphs. We discuss the very promising results obtained by this approach, as well as the numerous perspectives that it opens, such as the computation of dissimilarity in the framework of attributed graphs, through a unified approach.

Keywords: Distance, Dissimilarity, Random trees, Stochastic approach, Clustering, Graph clustering