Thesis

Méthode pour le pré-traitement et l'extraction des biomarqueurs en spéctrométrie de masse

PRADOS, Julien

Abstract

La spectrométrie de masse offre des outils pour analyser la composition moléculaire d'un échantillons. En protéomique clinique, les technologies SELDI-TOF et MALDI-TOF ont récemment reçus une attention particulière pour leur facultés à analyser des échantillons biologiques de patients (serum sanguin, urine, tissus,...). Malgré leur résolution faible, ces technologies offrent en effet la possibilité de dresser rapidement le profil d'expression de centaines de protéines pour des centaines d'échantillons, ce qui permet d'envisager leur utilisation pour le diagnostique de patients et la recherche de biomarqueurs de certaines maladies. Ces applications soulèvent néanmoins des interrogations quant à la fiabilité des résultats que l'on peut tirer de ces technologies. Cette thèse aborde différents aspects pour améliorer la fiabilité des résultats. Nous étudions principalement le problème de l'analyse bio-informatique des données mais aussi l'amélioration des protocoles de préparation des échantillons biologiques. Concernant l'analyse bio-informatique, les difficultés majeures résident dans la [...]

Reference

PRADOS, Julien. *Méthode pour le pré-traitement et l'extraction des biomarqueurs en spéctrométrie de masse*. Thèse de doctorat : Univ. Genève, 2009, no. Sc. 4154

URN : urn:nbn:ch:unige-51517
DOI : 10.13097/archive-ouverte/unige:5151

UNIVERSITÉ DE GENÈVE

FACULTÉ DES SCIENCES

Département d'informatique

Professeur Christian Pellegrini
Docteur Alexandros Kalousis

Méthode pour le pré-traitement et l'extraction des biomarqueurs en spectrométrie de masse

THÈSE

présentée à la Faculté des sciences de l'Université de Genève
pour obtenir le grade de Docteur ès sciences, mention informatique

par

Julien Prados

de

Aouste sur Sye (France)

These N^o 4154

GENÈVE
ReproMail
2009

UNIVERSITÉ DE GENÈVE

FACULTÉ DES SCIENCES

Doctorat ès sciences
mention informatique

Thèse de *Monsieur Julien PRADOS*

intitulée :

" Méthode pour le pré-traitement et l'extraction des biomarqueurs en spectrométrie de masse "

La Faculté des sciences, sur le préavis de Ch. PELLEGRINI, professeur ordinaire et directeur de thèse (Département d'informatique), A. KALOUSIS, docteur et codirecteur de thèse (Département d'informatique), M. MÜLLER, docteur (Proteom Informatics Group, Swiss Institute of Bioinformatics, Geneva, Switzerland), I. XENARIOS, professeur (Vital-IT and SwissProt Groups, Swiss Institute of Bioinformatics, Geneva, Switzerland) et de Madame E. MARCHIORI, professeur (Machine learning group, Intelligent systems section, Institute for computing and information science, Radboud University, Nijmegen, The Netherlands), autorise l'impression de la présente thèse, sans exprimer d'opinion sur les propositions qui y sont énoncées.

Genève, le 23 novembre 2009

Thèse - 4154 -

Le Doyen Jean-Marc TRISCONE

N.B.- La thèse doit porter la déclaration précédente et remplir les conditions énumérées dans les "Informations relatives aux thèses de doctorat à l'Université de Genève".

Nombre d'exemplaires à livrer par colis séparé à la Faculté : - 4 -

Les savants des temps passés et des nations révolues n'ont cessé de composer des livres, ils l'ont fait pour léguer leur savoir à ceux qui les suivent. Ainsi demeurera vive la quête de la vérité. Et ne sera pas vaine la peine qui fut la leur en découvrant les secrets de la science et en en éclairant la part obscure. Tel homme découvre une nouveauté (jusqu'à lui inconnue) et la lègue à ceux qui viennent après lui. Tel autre ouvre ce qui est resté clos chez les Anciens : il jette une lumière sur le chemin, il facilite l'accés. La prise est proche. Tel autre encore trouve des erreurs en quelque livre : il cherche à réparer, à réctifier, sans accabler l'auteur, ni tirer gloire de sa rectification.

al-Khwārizmī[1]

1. Mathématicien arabe du IX^{e} siècle, exerçant à l'institut *Beit al Hikma* (Maison de la Sagesse) de *Bagdad* (anciennement nommée *Madinat al Salam* : ville de la paix).

Le travail qui suit, a été rendu possible grâce aux financements de l'Office Fédéral de l'Éducation et de la Science (OFES) dans le cadre du projet européen *Fun Prot* (Knowledge Discovery in Functional Proteomics), COST Action 282 : Knowledge Exploration in Science and Technology KnowIEST[2].

2. www.mpa-garching.mpg.de/~opmolsrv/COST282/

Remerciements

Avant de commencer, il me faut remercier ceux est celles qui m'ont accompagné durant ce travail. Je pense avant tout à Mélanie Hilario qui est à l'initiative du projet. Elle a obtenu les fonds qui m'ont permis de commencer ce travail, elle a initié les contacts avec moi, et elle a choisie de miser sur moi. Donc un grand MERCI à Mélanie. Je pense aussi à Christian Pellegrini, une personne formidable pour qui j'ai la plus grand estime, qui sait toujours répondre présent dans les moments difficiles. Je pense enfin à Alexandros Kalousis qui a suivi de près mon travail et qui a su en orienter les aspects scientifiques par ses conseils avisés. Nous avons passé beaucoup de temps à discuter du travail qui suit, et il a passé beaucoup de temps à étudier ce document.

Il est également évident que tous les membres du laboratoire d'intelligence artificielle ont eut un rôle important pour l'ambiance sympathique et à la fois studieuse qui régnait. Avec les personnes de ma sphère privée, ils ont joué un rôle de soutient important, ils m'ont apporté le réconfort et les moments de détente nécessaires. J'ai eu la chance de vivre une aventure importante de ma vie avec vous, j'y ai pris beaucoup de plaisir, j'ai appris énormément, et je le dois en grande partie à vous. Merci à tous.

Résumé

La spectrométrie de masse offre des outils pour analyser la composition moléculaire d'un échantillons. En protéomique clinique, les technologies SELDI-TOF et MALDI-TOF ont récemment reçus une attention particulière pour leur facultés à analyser des échantillons biologiques de patients (serum sanguin, urine, tissus, ...). Malgré leur résolution faible, ces technologies offrent en effet la possibilité de dresser rapidement le profil d'expression de centaines de protéines pour des centaines d'échantillons, ce qui permet d'envisager leur utilisation pour le diagnostique de patients et la recherche de biomarqueurs de certaines maladies. Ces applications soulèvent néanmoins des interrogations quant à la fiabilité des résultats que l'on peut tirer de ces technologies. Cette thèse aborde différents aspects pour améliorer la fiabilité des résultats. Nous étudions principalement le problème de l'analyse bio-informatique des données mais aussi l'amélioration des protocoles de préparation des échantillons biologiques.

Concernant l'analyse bio-informatique, les difficultés majeurs résident dans la structuration des données, et dans leur forte dimension qui perturbent de nombreux algorithmes d'extraction de connaissances. Pour répondre à ces difficultés, la thèse propose d'une part des algorithmes pour le pré-traitement des spectres de masses (élimination de ligne de base, détection des pics, alignement des pics, ...), et d'autre part, une méthode de sélection d'attributs adaptée à la recherche de biomarqueurs. En effet, les problématiques soulevées par la recherche de biomarqueurs dans les données de spectrométrie de masse, soulèvent en réalité des questions plus général sur sélection d'attributs.

Pour aborder ces questions dans notre travail, nous optons pour une approche de sélection d'attributs qui s'appuie sur les machines à vecteurs supports (SVM) car elle permettent de traiter efficacement la forte dimension des données. Ensuite trois aspects nous guident dans le design de notre méthode : 1) la pertinence des résultats ; 2) la stabilité des résultats qui garantit leur robustesse et leur reproductibilité ; 3) l'interprétation des modèles avec une attention particulière portée à la compréhension les interactions entre les variables. Les deux premiers points permettent de définir un cadre pour comparer la qualité de notre approche face à d'autres méthodes. Le troisième point cherche à répondre à la demande particulièrement forte des experts pour comprendre les interactions protéine-protéine qui entre dans le mécanisme biologique de la maladie. A ce sujet nous montrons notamment l'importance de la normalisation des données dans l'interprétation des modèles.

Table des matières

Chapitre 1

Introduction

L'augmentation "relativement récente" des capacités de stockage numérique rend possible l'accumulation de données volumineuses dans lesquelles sont stockées de nouveaux types d'informations. On voit en particulier se multiplier le stockage de nombreux signaux numériques en provenance de périphériques de capture divers. Les exemples sont nombreux et touchent tous les domaines : photos, vidéos, sons, données sismiques, données météorologiques, données financières, données médicales (cardiologie, radiologiques, etc.), données biologiques (spectrométrie de masse, imagerie RMN, images de bio-puces, etc.), données astronomique, etc. Cette thèse s'intéresse plus particulièrement aux signaux issus de l'analyse par spectrométrie de masse d'échantillons biologiques, toutefois, les problématiques que l'on rencontre dans ce domaine sont similaires à celles auxquelles on est confronté dans les autres domaine, et on peut espérer que les méthodes développées pour l'analyse de données de spectrométrie de masse soient, dans une certaine mesure, généralisables à d'autres domaines. Tous ces domaines expriment de plus un besoin grandissant pour des outils capables d'analyser et d'extraire de la connaissance des signaux qu'ils génèrent. Malheureusement, les données des signaux sont encore assez difficiles à traiter en apprentissage automatique car 1) elles sont peu structurées et nécessite un effort important pour en extraire l'information qui nous intéresse, 2) elles sont de très grande dimension, 3) elles contiennent souvent beaucoup de redondance, 4) elles sont généralement bruitées, 5) il peut y a des dépendances temporelles entre les exemples d'apprentissage. De plus, le confort apporté par les nouvelles capacités de stockage a modifié les comportements des utilisateurs qui ont tendance à accumuler des informations même si elles sont de mauvaise qualité, périmées, ou peu pertinentes pour leur problème étudié dans l'espoir qu'elles serviront un jour. Dans de telles conditions, plusieurs challenges se posent au *data-minier* pour extraire des connaissances de ces données.

Le premier challenge auquel est confronté le *data-miner* est de structurer l'information que renferment les signaux par des étapes de **pré-traitement** des données. Cela consiste à identifier et à extraire l'information qui nous intéresse dans les signaux afin d'exprimer les données sous la forme d'un problème d'ap-

prentissage propositionnel pour lequel on dispose de nombreux outils d'extraction de connaissances. Ces étapes de pré-traitement servent aussi à nettoyer, à réduire, et à uniformiser l'information en faisant appel aux connaissances du domaine pour se focaliser sur les données pertinentes (par exemple les pics dans les signaux de spectrométrie de masse). Elles s'avèrent souvent coûteuses à mettre en place, et difficile à paramétrer, même si en spectrométrie de masse les corrections et les étapes de pré-traitement à apporter aux données sont relativement bien identifiées [90, 48]. La première partie de cette thèse est donc dédiée à la problématique du pré-traitement des signaux issus de spectrométrie de masse, et à l'ajustement des paramètres qui le contrôlent. Dans cette partie, nous définissons et nous proposons des méthodes de pré-traitement des spectres de masse en mettant l'accent sur leur simplicité conceptuelle de manière à ce que leur utilisation paraisse naturelle et que le nombre de paramètres à ajuster par l'utilisateur soit limité. De plus, notre contribution ne se limite pas à la définition de méthodes de pré-traitement à suivre, mais nous tâchons également d'évaluer leur efficacité.

Le deuxième challenge dans l'extraction de connaissances à partir de signaux vient de la forte dimension de ce type de données et de la redondance d'information qu'ils contiennent. Même si pendant le pré-traitement on cherche à réduire la quantité d'information et à éliminer de la redondance, le jeu de données produit peut contenir des centaines, voir des milliers de variables qui sont pour la plupart inintéressantes pour la connaissance que l'on cherche à extraire. Or, de telles conditions perturbent de nombreux algorithmes d'apprentissage automatique, qui ont tendance à produire des modèles complexes et peu performants [45]. Une manière d'éviter ces problèmes de dimension consiste à employer des méthodes de sélection d'attributs pour focaliser l'apprentissage sur un sous-ensemble de variables pertinentes. La sélection d'attribut a également l'avantage de simplifier les modèles décisionnels et de faciliter leur interprétation. Ce dernier point est un aspect essentiel pour l'utilisateur final, le biologiste, qui cherche à comprendre les résultats de l'expérience qu'il à mis en place. Une alternative à la sélection d'attributs consiste à utiliser des algorithmes d'apprentissage adaptés à la grande dimension des données, comme les Machines à Vecteur Support (SVM). Celles-ci offrent de plus des facilités pour introduire des connaissances du domaine dans l'apprentissage par l'intermédiaire de fonctions noyaux. La deuxième partie de cette thèse aborde ces problématiques : nous nous intéressons effectivement aux méthodes de sélection d'attributs basées sur SVM, et nous proposons d'utiliser la fonction noyau logRatio pour répondre 1) aux exigences de la biologie (besoin d'interpréter des modèles, besoin de comprendre les interactions entre les variables), 2) aux problèmes de dimension des données, 3) aux difficultés liées au pré-traitement des données et notamment vis à vis de la normalisation des données.

L'objet de cette thèse est donc l'extraction de connaissances à partir de signaux de spectrométrie de masse. Voyons plus précisément de quoi il retourne, quels sont les objectifs visés dans ce domaine, et les challenges auxquels cela nous confronte.

1.1 Apprentissage automatique avec des signaux de spectrométrie de masse

Un spectre de masse est un signal résultant d'une analyse moléculaire sur un échantillon, et qui permet d'obtenir des informations sur sa composition chimique. Plusieurs technologies peuvent être utilisées pour générer des spectres de masse, mais celles qui nous intéressent plus particulièrement ici sont les technologies SELDI-TOF et MALDI-TOF. Ces technologies sont appréciés des laboratoires biomédicaux pour leur simplicité d'utilisation et leur rapidité à analyser le protéome des patients, parfois au détriment de la fiabilité de l'analyse. Effectivement, chaque échantillon prélevé sur un patient (par exemple du sang ou des urines) donne lieu à un ou plusieurs spectres de masse qui reflètent les protéines présentent dans son corps. En comparant les spectres de masse issus de patients souffrants d'une pathologie (les malades) à ceux issus de patients qui n'en sont pas atteint (les contrôles), les médecins espèrent mettre en évidence des protéines qui jouent un rôle dans la maladie étudiée. Ces protéines actives que l'on appel *biomarqueurs* sont très précieuses pour les chercheurs car elles aident à comprendre les mécanismes de la maladie, elles peuvent être utiliser pour produire des diagnostiques plus précoces et/ou plus performants de la maladie, et elles peuvent parfois être la cible de nouveaux traitements. L'objet de cette thèse est de proposer des méthodes de traitement et d'analyse de données de spectrométrie de masse pour le diagnostique de patients à partir de données de spectrométrie de masse, et surtout pour l'extraction des biomarqueurs potentielles.

Du point de vu de l'apprentissage automatique, extraire les biomarqueurs à partir de spectres de masse n'est cependant pas trivial. D'ailleurs, les premiers travaux de Petricoin et al. [86, 87] dans ce domaine ont était remis en question [111, 5, 15]. Effectivement, en plus des problèmes mentionnés plus haut (relatifs aux pré-traitement des signaux et à la dimension des données) des questions se posent sur la reproductibilité des technologies MALDI-TOF/SELDI-TOF et des méthodes employées. Ces problèmes poussent médecins et biologistes à formuler d'avantage d'exigences sur les résultats qui leurs sont fournis. Par exemple, pour compenser le fait que les technologies employées ne sont pas fiable à 100%, il faut fournir des garanties que les algorithmes utilisés ne sont pas sensibles à ces variations de données. Il s'agit donc de montrer la bonne stabilité des algorithmes de pré-traitement, et des modèles décisionnels que nous construisons par rapport aux variations des conditions expérimentales. Les utilisateurs ont également d'autres exigences. Une d'elle est particulièrement importante à leur yeux, c'est le besoin de comprendre les modèles décisionnels qui sont construits. Effectivement, il ne suffit pas de construire un modèle décisionnel qui soit performant pour distinguer les groupes de patients sain/malade, mais il faut en plus pouvoir expliquer quels sont les relations entre les variables qui permettent à ce modèle de prendre ses décisions. Car, gardons à l'esprit que ce que recherche le biologiste c'est comprendre les **interactions** entre les protéines qui peuvent être impliquées dans la maladie. Il faut donc veillez à construire des modèles

facile à interpréter et qui fournissent, si possible, des informations sur les interactions entre les variables. Dans cette thèse nous prendrons soin d'apporter une contribution à l'ensemble de ces points.

1.2 Organisation du document

Cette thèse propose donc une méthode complète pour l'extraction des biomarqueurs à partir de données de spectrométrie de masse. Elle s'organise en deux grandes parties. La première aborde le pré-traitement des spectres de masse. Elle propose des algorithmes de pré-traitement qui permettent de corriger les spectres de masse, d'en extraire l'information pertinente et de la structurer. A l'issue de cette partie, les algorithmes proposés sont utilisés pour mener une étude de la reproductibilité des protocoles de préparation des échantillons biologiques (chapitre 6). Nous menons également une étude approfondie du comportement de ces algorithmes de pré-traitement (chapitre 7) dans laquelle on cherche d'une part à ajuster les paramètres qui contrôle le pré-traitement de manière à extraire une information de la meilleure qualité possible, et d'autre part à montrer les qualités de notre approche pour fournir des résultats robustes (stable) face aux variations expérimentales.

La seconde partie du document propose quand à elle une méthode d'apprentissage qui essaie de répondre aux exigences rencontrées pour l'extraction des biomarqueurs en spectrométrie de masse. Cette méthode s'articule autour d'une machine à vecteur support (SVM) et la définition d'une fonction noyau, que nous appelons logRatio. Cette méthode regroupe plusieurs avantages qui permettent de répondre aux exigences du domaine : elle focalise l'apprentissage sur les interactions entre les variables (c'est à dire les protéines) ; elle permet de tirer des conclusions sur la redondance entre les variables ; elle s'intègre avantageusement au SVM pour traiter les données de grande dimension ; elle produit des modèles facilement interprétable ; elle permet de définir un algorithme de sélection d'attribut pour l'extraction des biomarqueurs ; elle permet de s'abstraire des problèmes de pré-traitement liées à la normalisation des données. Cependant, même si notre méthode a été construite pour répondre aux exigences de ce domaine, elle ne se limite pas à celui-ci, et nous montrons aussi son utilité pour traiter d'autres types de données.

Mais, avant d'entrer dans le vif du sujet avec ces deux parties, le chapitre 2 introduit plus en détail le domaine de la spectrométrie de masse et de la protéomique, et situe plus précisément notre travail dans ce contexte.

Chapitre 2

Spectrométrie de masse & Protéomique

Ce chapitre introduit la spectrométrie de masse et la protéomique qui sont les principaux domaines d'application des traitements informatiques proposés dans cette thèse. Il a pour objectifs de présenter les buts à atteindre et les exigences à respecter dans ce domaine, de montrer les principes de fonctionnement des instruments qui produisent les données (les spectres de masses), leurs limitations, et les problèmes rencontrés durant l'analyse de ces données. Ces informations constitueront autant de connaissances du domaine qui serviront par la suite pour nous guider dans le traitement de ces données.

2.1 Spéctrométrie de masse

La spectrométrie de masse est une technique d'analyse moléculaire qui permet de séparer les molécules d'un échantillon en fonction de leur masse. Elle est utilisée dans de nombreux domaines scientifiques comme la physique, la chimie, la biologie, la médecine, pour déterminer la composition moléculaire des échantillons récoltés dans ces domaines. Une grande variété d'instruments existent, certains capables de distinguer des molécules dont les masses différent de quelques dixièmes de Daltons – 1 Dalton est égale par définition à un douzième de la masse d'un atome de carbone-12, soit environ 1.67×10^{-24} grammes –, d'autres de fragmenter les molécules et de mesurer la masse de ces fragments pour obtenir des informations structurales, d'autres encore de quantifier le nombre de molécules dans les échantillons mêmes si elles sont présentes en très faibles quantités.

Malgré cette diversité, tous les instruments adoptent le même schéma de fonctionnement : la substance à analyser est d'abord introduite dans l'appareil par un *système d'introduction* ; un dispositif appelé *source d'ion* s'occupe alors de charger électriquement les molécules (ionisation) ; les ions produits sont triés par un *analyseur* en fonction de leur rapport masse/charge ; enfin, les ions sont

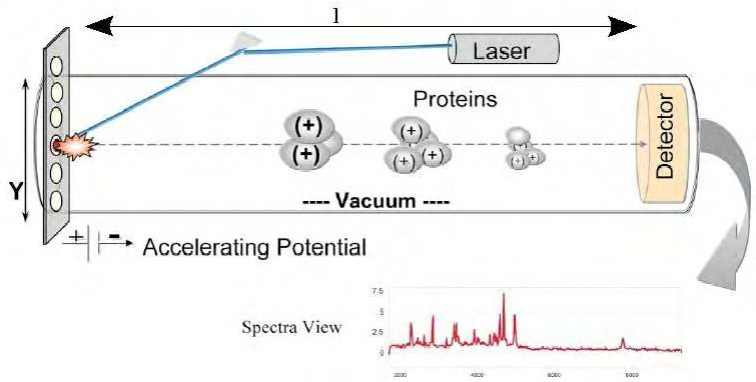


FIGURE 2.1 – Illustration du fonctionnement des spectromètres MALDI-TOF et SELDI-TOF

détectés par *un capteur*, et les signaux résultants sont collectés et échantillonnés sur des dizaines de milliers de points pour produire le spectre de masse [2, 10].

Dans cette thèse, nous nous intéressons plus spécifiquement aux données générées par les instruments MALDI-TOF et SELDI-TOF. Ces instruments fonctionnent de manière identique, mais le second possède un dispositif additionnel qui facilite l'analyse des échantillons complexes. La figure 2.1 schématise leur fonctionnement, que l'on décrit brièvement dans les deux sous-sections suivantes.

2.1.1 MALDI-TOF

Dans un MALDI-TOF, la source d'ion est de type MALDI (Matrix Assisted Laser Desorption Ionization), et l'analyseur de type TOF (Time Of Flight). La source MALDI ionise les molécules de l'échantillon au moyen d'impulsions lasers envoyées en direction de l'échantillon, que l'on a au préalable fait prisonnier dans une substance appelée matrice. La matrice absorbe l'énergie du laser et par un phénomène pas encore totalement connu, elle "explose" en projetant les molécules de l'échantillon chargées électriquement. La majorité des ions produits par ce processus ont une seule charge.

L'analyseur TOF, repose quant à lui sur des principes fondamentaux de mécanique. Rappelons que sa tâche est de trier les ions en fonction de leur masse m. Pour cela, l'analyseur TOF génère un champ électrique de sorte que les ions qui viennent de la source reçoivent une énergie cinétique E_c proportionnelle à leur charge z. Les ions volent alors en direction du détecteur à travers un tube de longueur l (sous vide), mais tous ne volent pas à la même vitesse v. En effet, comme l'énergie cinétique E_c est liée à la masse m de l'ion et à sa vitesse v par l'équation $E_c = \frac{1}{2}mv^2$, les ions qui ont les rapports m/z les plus petits volent plus rapidement que les ions dont le rapport m/z est élevé. Ainsi un tri

s'opère dans l'analyseur, et lorsque qu'un ion percute le détecteur au temps t, son temps de vol permet de déduire sa vitesse de vol $v = \frac{l}{t-t_0}$, et surtout son rapport masse/charge $\frac{m}{z} = \frac{2E_c}{l^2}(t-t_0)^2$ – Notez que dans le cas d'une ionisation MALDI la charge z vaut généralement 1, et donc $m/z = m$.

Le détecteur situé en bout de tube compte les ions qui arrivent et le temps qu'ils ont mis pour parcourir la distance l. Ce périphérique produit le signal que l'on appel spectre de masse. L'amplitude $f(t)$ du spectre (appelée aussi intensité) indique la quantité d'ion qui est arrivée au temps t donné en abscisse, et une mise à l'échelle adéquate de l'axe des abscisses selon l'équation discutée précédemment, permet de convertir le temps de vol en rapport masse/charge. Les nombreux pics que l'on distingue dans le spectre de masse correspondent donc aux principaux constituants de l'échantillon. Leurs positions en abscisse reflètent la masse des molécules qui ont produits ces pics, et leurs intensités sont en relation avec l'abondance des molécules dans l'échantillon analysé. Cependant, comme nous en rediscuterons dans les sections 2.2.2 et 2.3, l'intensité est également liée à d'autres facteurs comme la capacité des molécules à être ionisées par le laser.

Soulignons qu'il est préférable que les échantillons aient une composition simple lorsqu'ils sont soumis à la technologie MALDI-TOF [2]. C'est pourquoi, lorsque les échantillons sont complexes, ils sont généralement fractionnés en plusieurs sous-échantillons, chacun contenant des molécules qui partagent une même propriété physico-chimique. Par exemple, il est courant d'utiliser des gels d'électrophorèse pour séparer les protéines selon leur poids isoéléctrique avant de les soumettre à spectrométrie de masse[2]. La technologie SELDI-TOF présentée dans la section 2.1.2 qui suit est une extension du MALDI-TOF qui offre un autre moyen, très simple, de fractionner des échantillons.

2.1.2 SELDI-TOF

Le spectromètre SELDI-TOF (Surface Enhanced Laser Desorption Ionization), commercialisé par la société *Ciphergen Biosystems*, fonctionne de la même manière qu'un instrument MALDI-TOF. Le SELDI offre cependant des facilités supplémentaires pour fractionner les échantillons complexes avant de procéder à leur analyse, ce qui le rend attractif pour les analyses bio-médicales. Le dispositif se présente sous la forme d'une surfaces chimique sur laquelle l'échantillon à analyser est déposé. Les molécules qui ont des affinités avec la surface s'y collent et sont les seules à être soumises à la spectrométrie MALDI, les autres sont éliminés lors d'un rinçage. Au final, le protocole expérimental de préparation de l'échantillon inclus donc les étapes suivantes : a) une période d'incubation où l'échantillon est déposé sur la surface et agit pendant 30 minutes, le temps que les molécules qui ont des affinités avec la surface se fixent ; b) un rinçage qui élimine les molécules qui n'ont pas réagi avec la surface ; c) le dépôt de la matrice qui autorise l'ionisation des molécules par le laser du MALDI-TOF. Selon la nature de la surface chimique choisie, différentes molécules s'agrippent à la surface et sont soumises à spectrométrie MALDI-TOF. Ainsi, en variant les

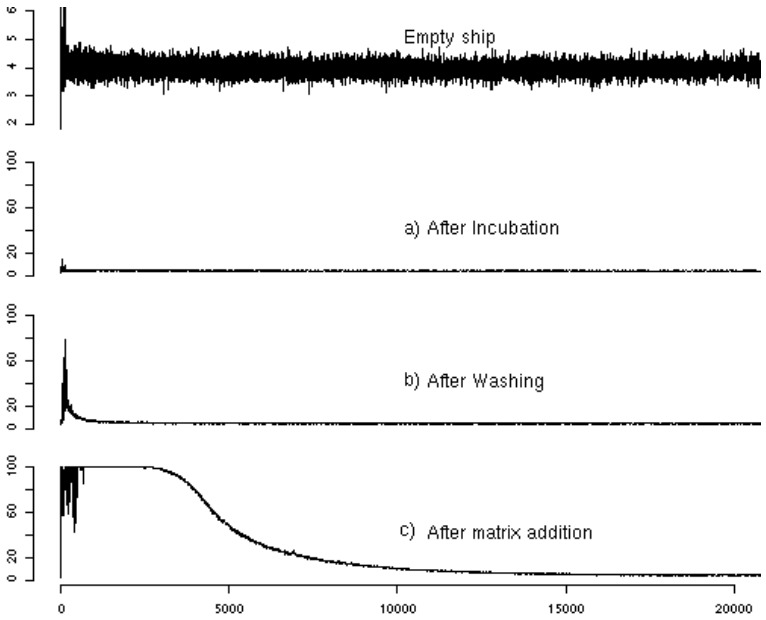


FIGURE 2.2 – Spectres obtenus à différentes étapes du protocole expérimental pour un échantillon vide. Notez la différence d'échelle sur le spectre du haut qui permet de mieux observer le bruit des spectres. *(Pour ces données, je tiens à remercier l'équipe BPRG de l'hôpital cantonal de Genève, et M. Nicolas Zaganidis, un ex-membre de notre équipe qui a mis en place l'expérience.)*

surfaces, on peut obtenir des spectres représentants des fractions différentes de l'échantillon initial.

A titre instructif, la figure 2.2 montre les spectres de masse SELDI-TOF obtenus après les différentes étapes du protocole expérimental mentionnées ci-dessus, lorsque l'on analyse un échantillon "vide" censé ne produire aucun pic sur le spectres de masse. Remarquez notamment sur cette figure l'apparition d'une ligne de base après l'ajout de la matrice, et aussi le bruit dans le signal de départ qui gène l'interprétation des données. La prise en considération de ces deux éléments dans le traitement des données sera discuté dans le chapitre 3.

2.2 MS, un outil pour la protéomique clinique

La protéomique est un domaine des sciences de la vie qui étudie les protéines (leur rôle, leur structure, leur localisation, leurs interactions) dans un organisme, un tissu ou une cellule. La protéomique clinique s'intéresse plus spécifiquement à l'homme et au rôle des protéines dans les maladies. Dans ce domaine, on cherche à comprendre les mécanismes biologiques qui régissent certaines maladies à par-

tir d'observations sur les protéines contenues dans le corps des patients. L'un des buts est de mettre au point des tests diagnostiques plus performants et plus précoces pour ces maladies, et de développer des traitements.

Les chercheurs ont notamment à leur disposition pour leurs études des outils de "criblage" du protéome qui leurs permettent de sonder l'expression des protéines dans le corps des patients, tel que les gels d'électrophorèses ou la spectrométrie de masse comme nous allons le voir. Avec ces outils, ils cherchent à mettre en évidence des changements d'activité entre les protéines des patients atteints par une maladie spécifique, et d'autres qui ne l'on pas (les patients contrôles, ou témoins). Si ces changements sont liés à la maladie, les protéines ainsi découvertes sont susceptibles d'être utilisés pour améliorer l'efficacité des diagnostiques, voir même d'être des cibles pour de nouveaux traitements. En tant que signature protéique de la maladie, on les appelle *biomarqueurs*.

Notez que pour certaines maladies, comme par exemple les cancers, certains experts estiment que la plupart des biomarqueurs singuliers (dont l'activité prise individuellement des autres protéines permet d'expliquer la maladie) ont déjà été identifiés [26]. Ils se tournent donc maintenant vers la recherche de biomarqueurs moins spécifiques, mais qui une fois combinés en panel peuvent améliorer les performances des diagnostiques. Notez aussi qu'il est important d'inclure dans les patients contrôles des personnes qui sont atteintes par des maladies différentes de celle étudiée, et pas seulement des personnes en bonne santé. En effet, comme cela on évite de faire ressortir dans les données des biomarqueurs qui seraient représentatifs d'un état de fatigue du corps, alors que l'on s'intéresse aux biomarqueurs spécifiques à la maladie étudiée. Enfin, signalons aussi que pour des raisons pratique, on préfère souvent rechercher des biomarqueurs dans des échantillons faciles à collecter comme l'urine et le sérum des patients car cela présente de multiples avantages : les risques d'infection neusocomiale sont limités ; on peut avoir de nombreux échantillons pour nos études car ils sont faciles à collecter ; il n'y a pas besoin d'infrastructure complexe et de personne qualifiée pour les collecter ; et tout cela réduit les coûts. Le lecteur intéressé par une liste de précautions à prendre tout au long d'une étude par spectrométrie de masse peut consulter l'article de Hortin, [51].

2.2.1 Rôle de la spectrométrie de masse en protéomique :

La spectrométrie de masse se révèle être un outil incontournable en protéomique car elle offre des moyens pour identifier les protéines, les séquencer, y détecter des modifications, obtenir des informations sur leurs structures, les quantifier, etc. – voir Aebersold et Mann [2] pour plus de détails à ce sujet –. Les instruments MALDI-TOF et SELDI-TOF ont de plus trouvé un intérêt particulier comme outils de "criblage" du protéome pour le diagnostique de patients et l'extraction de biomarqueurs [86]. Effectivement, ces technologies permettent, comme les gels d'électrophorèse, d'obtenir un profil protéique pour un patient à partir d'échantillons prélevés sur lui, par exemple du sérum, des urines, un tissu, et elle est relativement simple à mettre en oeuvre. En plus, cette technologie élargie le champ d'investigation jusqu'alors possible car elle permet d'étudier des

protéines de petite taille, invisibles sur les gels d'électrophorèses (typiquement de masse inférieure à 50000 Dalton).

Effectivement, les intensités des pics contenus dans les spectres de masse générés selon ces technologies reflètent l'abondance d'une partie des protéines contenues dans l'échantillon analysé. Ainsi, en trouvant des différences systématiques entre les spectres qui proviennent de patients sains et ceux qui viennent de patients malades, on peut espérer mettre en évidence des biomarqueurs de la maladie. Malheureusement, la comparaison des spectres est difficile à réaliser manuellement à cause du grand nombre d'informations qu'ils contiennent et du bruit. Deux exemples de spectres de masse SELDI-TOF réalisés à partir du sérum d'un patient ayant contracté un accident vasculaire cérébral est un autre à partir d'un patient sain sont donnés figure 2.3. On voit bien déjà sur ces figures les difficultés qu'il y a à comparer ces données, et l'on imagine le problème orsqu'il faut comparer des centaines de spectres. C'est pourquoi il est nécessaire de développer des outils informatiques capables de détecter et d'extraire automatiquement les "motifs" contenus dans ces données, et de les présenter clairement aux utilisateurs pour qu'ils puissent étudier les protéines impliquées et leurs interactions. Le travail que nous présentons dans cette thèse propose des outils pour réaliser cela.

2.2.2 Limitations technologiques

Les travaux pionniers dans l'utilisation des spectres de masse SELDI-TOF pour la recherche de biomarqueurs en protéomique clinique sont ceux de Petricoin et al. [86]. En 2002, ils analysèrent des spectres de masse pour identifier des bio-marqueurs du cancer des ovaires. Depuis, les études se multiplient et portent majoritairement sur les cancers : cancer des ovaires [86], cancer de la prostate [87, 1, 128, 96, 97, 105], cancer du rein [102], cancer du sein (voir [26, 49] pour une revue). Dans la majorité des cas, les performances reportées sont excellentes (sensitivité et spécificité de l'ordre de 80% à 100%), mais de nombreuses critiques ont été formulées quant à la qualité de certaines données [111, 5], et sur les limitations de la technologie SELDI-TOF pour cette tâche[26]. Il est très peu probable que l'on atteigne de tels performances dans la réalité.

Diamandis [26] reproche en particulier aux instruments SELDI-TOF d'être trop peu sensible, et selon lui, il est peu probable que des tumeurs cancéreuses puissent altérer le protéome d'un patient assez significativement pour être détecté par les instruments SELDI-TOF à partir de leur sérum. En plus de cela, il souligne que les protéines de faible abondance ont peu de chance d'être prise en compte par les technologies SELDI car elles sont en compétition avec d'autres protéines des milliers de fois plus abondantes dans les échantillons (e.g. albumine) pour s'attacher à la surface. Cette compétitivité entrave également les facultés quantitatives des instruments SELDI-TOF. Effectivement, si le nombre de protéines qui s'attache à la surface SELDI dépend du nombre de ses compétiteurs, alors une protéine qui a la même abondance dans deux échantillons distinct ne s'attachera pas de la même manière à la surface chimique car ses compétiteurs sont plus ou moins nombreux selon les échantillons analysés.

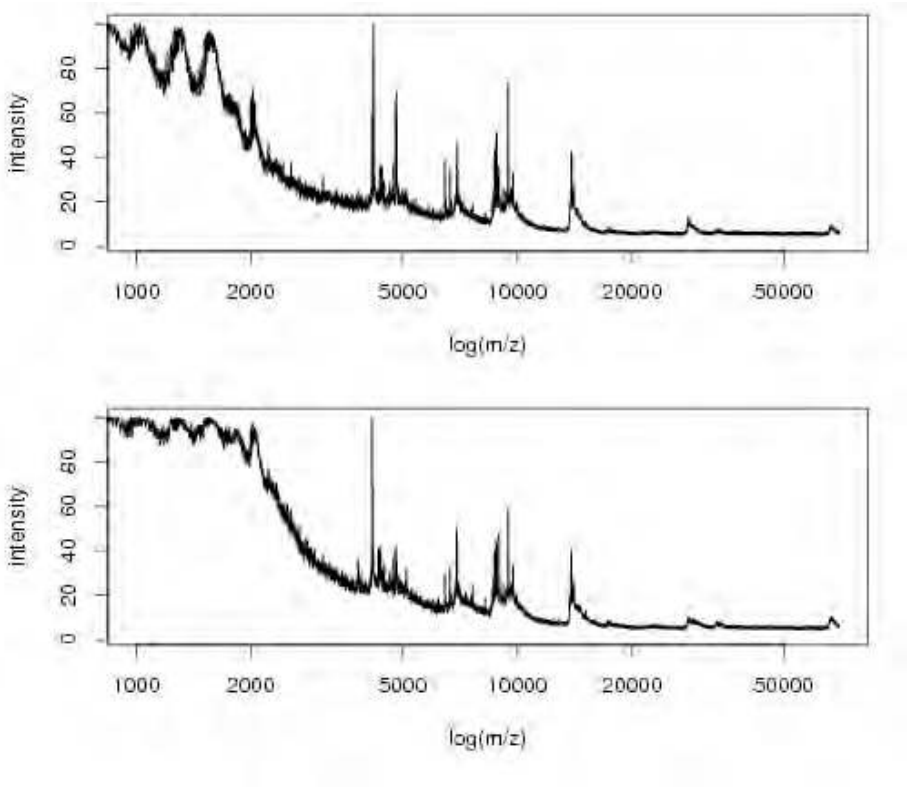


FIGURE 2.3 – Exemples de spectres de masse SELDI-TOF réalisés à partir du sérum d'un patient qui a contracté un accident vasculaire cérébral (en haut), et d'un patient sain (en bas).

Diamandis souligne aussi les incohérences entre les différentes études menées sur la recherche des biomarqueurs pour les cancers. Il note en particulier qu'aucune n'a fait ressortir les biomarqueurs que l'on connaissait déjà comme la PSA, dont la masse est pourtant située dans l'intervalle de couverture des SELDI-TOF. Encore une fois, cela viendrait, selon lui, de la faible sensibilité des instruments. Il remarque aussi que les différentes études n'ont pas fait ressortir les mêmes biomarqueurs, et plus grave encore, que des études bioinformatiques différentes portant sur les mêmes données n'ont pas extrait les mêmes biomarqueurs. Cela soulève des questions sur la reproductibilité de l'ensemble du processus expérimental qui sert à rechercher les biomarqueurs. Cela inclut non seulement les problèmes technologiques, mais aussi les méthodes bioinformatiques utilisées pour l'extraction des biomarqueurs. Nous abordons ces sujets dans la section qui suit.

Malgré ces limitations, et malgré le développement rapide d'autres technologies de spectrométrie de masse plus fiables et plus sensibles pour le criblage du protéome (typiquement à base de LC-MSMS : chromatographie liquide couplée à des instruments de MS/MS[2]), la spectrométrie de masse MALDI-TOF et SELDI-TOF reste aujourd'hui encore l'une des rares technologies qui permettent d'analyser un grand nombre d'échantillons (jusqu'à plusieurs centaines) rapidement et simplement. C'est une condition nécessaire si l'on souhaite mener des études à grande échelle, et si l'on souhaite appliquer des méthodes d'apprentissage automatique qui repèrent les biomarqueurs avec une certaine significativité statistique. C'est l'une des raisons pour laquelle dans cette thèse nous nous intéressons aux données MALDI-TOF et SELDI-TOF pour étudier le problème de l'extraction des biomarqueurs à partir des spectres de masse. Compte tenu des limitations de la technologie, nous veillerons cependant dans notre travail à proposer des méthodes suffisamment générales pour être applicables à d'autres technologies. Par exemple, dans le chapitre 4 nous proposons une méthode de détection de pics dans les spectres SELDI-TOF et MALDI-TOF qui se généralise aux données LCMS.

2.3 Reproductibilité des expériences MS

La reproductibilité est un problème récurrent en spectrométrie de masse que l'on retrouve sur plusieurs technologies. Une étude récente du laboratoire LSMV (Laboratoire de Spectrométrie de Masse du Vivant) de Genève montre par exemple qu'en répétant une expérience de LC-MSMS trois fois, il est possible d'identifier 207 protéines différentes en les considérant indépendemment les unes des autres, mais seulement 48 d'entre elles (environ 23%) sont identifiées dans les trois expériences (SOURCE : présentation de Emmanuel Vareisio le 15 mars 2006). L'origine de ce problème n'est pas clairement identifié car plusieurs étapes d'analyses sont nécessaires pour aboutir au résultat, et chacune est susceptible d'introduire des erreurs. Cependant, on suppose qu'une cause majeure de la défaillance est l'heuristique utilisée pour la sélection des pics précurseurs soumis

à la fragmentation MS/MS. Cette heuristique est basée sur le pic d'intensité maximale, mais les variations d'intensité d'une expérience à l'autre (le coefficient de variance des intensités est de l'ordre de 15% dans ce type d'expérience, [69]) empêchent de reproduire le comportement des expériences précédentes. Sans doute est-il possible d'améliorer l'heuristique pour avoir des résultats d'avantage reproductibles.

En ce qui concerne la technologie SELDI-TOF, la reproductibilité des expériences est le sujet de bien des débats qui ont commencé avec les articles de Sorase [111] et Baggerly [5]. Ces articles remettent en question les résultats obtenus lors de trois études réalisés au sujet du cancer des ovaires. Baggerly [5] s'attache à montrer les difficultés rencontrées pour généraliser les résultats obtenus sur un des jeux de données aux deux autres, il met en évidence des problèmes de pré-traitement des données (suppression de la ligne de base, normalisation des intensités des spectres), des problèmes d'alignement des spectres (calibration), des problèmes de paramétrage des instruments (saturation du signal), et il identifie également des structures dans le bruit de fond du signal qui ont peu de sens biologique. Tous ces problèmes l'empêchent de reproduire les résultats publiées initialement par l'équipe qui a produit les données. Au final, il conclut que les résultats ne sont pas généralisables d'une expérience à l'autre, et insiste sur la nécessité d'améliorer l'ensemble du processus, en allant de l'acquisition des échantillons biologiques à l'analyse bioinformatique, pour que les expériences SELDI-MS deviennent reproductibles et que les biomarqueurs extraits à partir d'une expérience puissent être validés et exploités par d'autres laboratoires situés en des endroits différents de la planète. Au niveau de l'analyse bioinformatique, il insiste aussi sur l'importance du pré-traitement, et l'avantage qu'il y a de réduire l'information en détectant les pics dans les spectres de masse plutôt que de considérer l'ensemble du signal car cela permet de s'abstraire du bruit et des éventuelles structures qu'il peut contenir.

Dans un article suivant, Hu et les mêmes auteurs [52] en viennent à souligner l'importance du design des expériences de spectrométrie de masse, et montrent sur plusieurs exemples que celui-ci peut compromettre l'analyse bioinformatique des données. Ils notent en particulier que l'un des paramètres qui influe le plus les données est la date d'acquisition des spectres, un phénomène aussi observé par [102, 53] et lié en partie à l'usure de l'instrument dont certains composants doivent être régulièrement entretenus (laser, détecteur), et aussi beaucoup à la manière dont sont stockés les échantillons biologiques [123]. A la suite de leurs expériences, les auteurs énoncent plusieurs conseils pour réaliser des expériences SELDI-TOF de qualité, dont notamment la nécessité d'examiner les échantillons dans un ordre aléatoire, et des directives pour la conservation des échantillons par congélation. Finalement, en suivant ces recommandations lors d'une étude utilisant un instrument SELDI-TOF pour examiner l'activité des protéines dans le plasma de patientes atteintes d'un cancer du sein, Hu et al. [52] prétendent arriver à des résultats reproductibles dans le temps, et cela même sur une période de plusieurs mois. Ces résultats renforcent donc la crédibilité de la technologie SELDI-TOF pour les applications où l'on cherche à examiner le profil protéique des patients pour y détecter des biomarqueurs potentiels.

Une étude de plus grande envergure a également été lancée par le "Early Detection Research Network" (EDRN) pour évaluer la reproductibilité de la technologie SELDI-TOF entre plusieurs laboratoires [40]. Le rapport préliminaire de cette étude [105] est assez encourageant, car il conclut qu'en adoptant un protocole de préparation rigoureux, et une politique stricte de contrôle de qualité, il est possible d'obtenir une certaine reproductibilité des résultats. Deux éléments sont à retenir de cette étude. D'une part les estimations de la variation de la position (coefficient de variance, CV : 0.03-0.12%), de l'intensité (CV : 15-36%), et de la résolution des pics (CV : 6-23%) qui permettent de conclure que la reproductibilité inter-laboratoire est aussi bonne que la reproductibilité intra-laboratoire. Ces valeurs ont été obtenus en analysant 3 des pics des 96 spectres de masse produits par les six laboratoires à partir du même échantillon, et sont en accord avec des résultats précédents de Rogers et al.[102] et de Hong et al. [50]. Et d'autre part le succès obtenus par l'auteur pour classifier 28 spectres de masse produits par chacun des six laboratoires à l'aide d'un modèle construit 2 ans auparavant, mais ces bons résultats pourrais être biaisé par le fait que ces mêmes échantillons ont servi à l'apprentissage du modèle, d'ou peut-être un effet de sur-apprentissage.

Mentionnons enfin les travaux de Pelikan et al. [85] dont les résultats sont également encourageants vis à vis de la reproductibilité de la technologie SELDI-TOF et de son application pour l'extraction de biomarqueurs potentiels. Lors d'une étude sur le cancer du poumon, 46 échantillons de patients ont été analysés par spectrométrie de masse SELDI-TOF à quatre reprises, sur une période de 18 mois. Les résultats de ce travail montrent que les signaux des spectres de masse sont statistiquement reproductibles sur cette période.

2.3.1 Améliorations de la reproductibilité :

Pour améliorer la reproductibilité des expériences de spectrométrie de masse SELDI-TOF, nous pouvons agir à plusieurs niveaux. Effectivement, une expérience inclut plusieurs étapes, et nous pouvons agir sur chacune d'elles pour améliorer sa reproductibilité. Par exemple, au niveau de la préparation biologique des échantillons plusieurs protocoles sont possibles et l'on peut se demander lequel est le plus approprié ; lors de l'acquisition des spectres de masse, on peut aussi se demander comment ajuster les paramètres de l'instrument de manière optimale (voltage, temps de latence, nombre de tirs laser, etc...) ; enfin on peut également agir sur l'analyse bioinformatique des données et proposer des méthodes robustes aux variations couramment observées dans les spectres de masse.

Dans le travail qui suit, c'est principalement l'amélioration des méthodes bioinformatiques qui nous intéresse. Dans ces méthodes, nous distinguons entre le pré-traitement des données dont le but est de structurer et d'extraire l'information contenue dans les spectres de masse (traité dans la Partie I) et les méthodes d'apprentissage dont le but est d'extraire les biomarqueurs potentielles et de construire des modèles de classification pour la prédiction des maladies à partir des spectres de masse (Partie II). Cependant, en collaboration avec la fondation pour la recherche biomédicale de l'académie d'Athènes (qui s'est occupée de la

génération des données) nous avons également mené une des rares études sur l'amélioration de la reproductibilité des protocoles expérimentaux pour l'analyse d'échantillons d'urine [131] qui vient compléter le travail de Rogers et al. [102] également à ce sujet. Nous présentons ce travail dans le chapitre 6, et le lecteur peut également se référer à l'article [131].

Première partie

Prétraitement des Spectres de Masse

De manière générale en apprentissage automatique, l'objectif du pré-traitement des données est de produire une structure cohérente de l'information pour qu'elle puisse être exploiter par les méthodes d'apprentissage (qui font l'objet de la deuxième partie de la thèse). En ce qui concerne la spectrométrie de masse et la problématique de l'extraction des biomarqueurs, l'idéal est que les méthodes d'apprentissage aient en entrée une matrice d'expression de l'activité des protéines dans le corps des patients. Cette matrice $\mathbf{X}$, à l lignes et n colonnes, contient la concentrations de n protéines mesurées chez l patients distincts. Nous utilisons aussi dans la suite le terme *jeu de données* pour faire référence à cette matrice d'expression qui constitue notre ensemble d'apprentissage.

Il est possible de construire cette matrice d'expression (ou du moins de s'en approcher) grâce à des spectres de masse réalisés à partir de l échantillons prélevées sur les patients que l'on souhaite étudier. En effet, les variations d'intensité observées dans ces spectres de masse sont liées à la concentration des molécules contenues dans les échantillons. On peut donc envisager réaliser l'apprentissage avec les spectres de masse, cependant utiliser les données telle quelle pose plusieurs problèmes. D'abord cela suppose que les spectres sont tous acquis dans des conditions identiques : même fréquence d'échantillonage, même plage de variation en m/z, même calibration des instruments. Toutes ces contraintes ne peuvent être respectés que si les spectres sont acquis dans un intervalle de temps restreint. C'est souvent le cas des ensembles d'apprentissage que l'on rencontre dans la littérature [85], mais dans la pratique ces contraintes sont trop fortes car on aimerait pouvoir traiter des spectres acquis à des moments différents et en des endroits différents (par exemple, pour faire du diagnostique de patients à partir de leurs spectre de masse). Un autre problème lorsque l'on utilise les données brutes telle quelle, c'est que l'on introduit beaucoup de bruit et de redondance dans les matrices d'expression et que cela perturbe de nombreux algorithmes d'apprentissage qui ont des difficultés avec les données de grande dimension.

A l'inverse, en construisant une matrice d'expression qui se limite aux informations pertinentes contenues dans les spectres, c'est à dire les pics, on limite ces problèmes. En effet, les pics concentrent l'information pertinente des spectres de masse puisque leur intensité (et plus exactement leur aire) est censée refléter la concentration de la/des molécule(s) dissimulée(s) derrière. Ainsi la taille de la matrice d'expression est considérablement réduite, passant de plusieurs dizaines de milliers de colonnes à quelques centaines. Par contre, l'effort de pré-traitement nécessaire pour construire la matrice d'expression est beaucoup plus important. Cette partie de la thèse est justement dédiée à ce sujet.

On distingue plusieurs étapes de pré-traitement pour extraire les pics des spectres de masse et construire la matrice d'expression. Certaines d'entre elles ont pour but de corriger les défauts des spectres de masse par un traitement du signal : élimination de la ligne de base, élimination du bruit, normalisation des amplitudes des spectres (chapitre 3) ; une étape est en charge de la détection des pics et du calcul de leurs caractéristiques tel que intensité, largeur, aire (chapitre 4) ; enfin une dernière étape est responsable de l'alignement des pics détectés dans les différents spectres de masse pour identifier ceux qui corres-

pondent à la même molécule sous-jascente (chapitre 5). Le diagramme de la figure 2.4 montre l'enchaînement de ces étapes dans notre approche, et il peut s'avérer utile de s'y référer pour se situer tout au long des chapitres. Signalons que dans notre approches du pré-traitement des spectres de masse, nous avons cherché une certaine simplicité, en limitant le nombre d'étapes et en se focalisant sur des procédures :

1. simples dans leur conception et qui paraissent naturelles à utiliser pour résoudre les problèmes posés.
2. rapides dans leur exécution pour pouvoir les utiliser à grande échelle.
3. qui ne détruisent pas le signal des spectres de masse.
4. qui nécessitent peu de paramètres à ajuster pour l'utilisateur.
5. ne sont pas spécifiques au traitement des spectres de masse MALDI-TOF & SELDI-TOF, mais peuvent se généraliser à d'autres types de données.

L'essentiel des méthodes ont fait l'objet de publications [90, 89, 61, 131], et une bonne revue des méthodes de pré-traitement pour les spectres de masse est disponible dans [48].

Enfin, en plus de proposer une succession d'étapes pour le pré-traitement des spectres de masse MALDI-TOF & SELDI-TOF, les chapitres 7 et 6 s'attachent à montrer leur pertinence et leur utilité. Dans le premier, on s'intéressent à l'évaluation de la qualité du pipeline de pré-traitement proposé, et à l'étude des paramètres qui le contrôle. Dans le second, nous utilisons les algorithmes de pré-traitement comme des outils pour évaluer la qualité des protocoles de préparation des échantillons.

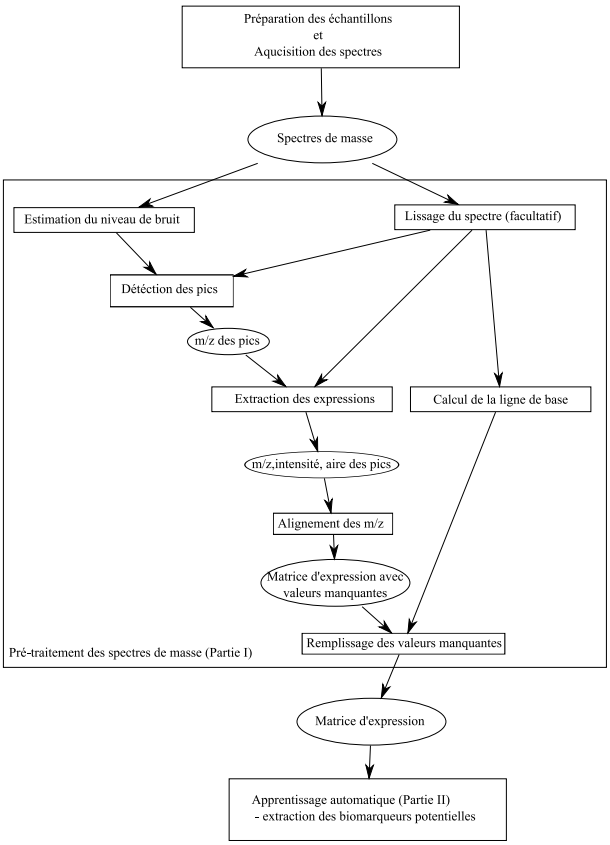


FIGURE 2.4 – Diagramme montrant l'enchaînement des étapes de pré-traitement. Les boites rectangulaires représentent une étape de traitement, les boites ovales décrivent le type de donnée obtenu lorsqu'il change.

Chapitre 3

Traitement du signal MS

Ce chapitre présente des méthodes de traitements des signaux que l'on rencontre couramment pour corriger les imperfections des spectres de masse, les uniformiser, et les rendre comparables les uns aux autres. Ce nettoyage des signaux et ces corrections ont pour but de faciliter l'interprétation des données par la suite, et notamment de faciliter la détection des pics et le calcule de l'abondance des protéines. Effectivement, à l'issue des étapes de pré-traitement présentées dans ce chapitre, nous nous attendons à ce que les signaux générés à partir d'un même échantillon biologique soient le plus superposables possible les uns sur les autres. Les corrections nécessaires à cela incluent : une élimination de la ligne de base (section 3.2), une estimation et l'élimination du bruit (section 3.3), et l'uniformisation/normalisation des intensités (section 3.4). Pour chacun de ces thèmes, un survol des méthodes existantes est fait, nous proposons une manière de les réaliser, et nous tachons d'analyser leur comportements. Rappelons aussi que nous avons mis l'accent sur la simplicité des méthodes, et que nous avons cherché à définir des procédures rapides, avec peu de paramètres, et qui ne détruisent pas le signal des spectres de masse.

3.1 Modélisation des spectres de masse

Pour débuter avec le traitement du signal MS, penchons nous sur la modélisation d'un spectre de masse. Un spectre de masse peut être vu comme une fonction du temps $f(t)$ dans laquelle on distingue trois composantes qui s'additionnent : le signal théorique, $S(t)$, produit par les molécules de l'échantillon si l'on se place dans des conditions d'acquisition idéales ; la ligne de base, $B(t)$, qui surélève le signal et le déforme ; et le bruit, $\epsilon(t)$, qui y introduit de petites oscillations de hautes fréquences. Il est également de rigueur d'introduire dans le modèle un facteur de normalisation N sur $S(t)$ afin de capturer la concentration de l'échantillon analysé. Au final donc, un spectre de masse se décompose selon l'équation 3.1 [21, 79, 48] :

$$f(t) = B(t) + N \times S(t) + \epsilon(t) \tag{3.1}$$

Le signal théorique $S(t)$ peut se modéliser à son tour comme une somme de pics indépendant les uns des autres, définis par leur position en m/z, leur hauteur, leur largeur, leur état de charge, ... On peut faire l'hypothèse que la forme des pics suit une distribution de probabilité normale même si d'autres distributions peuvent être plus appropriées [77]. Dans le cas des spectres de masse à haute résolution, on peut également adapter ces distributions aux profils isotopiques des pics[82, 113]. Dans notre approche, nous avons choisi de ne pas modéliser le signal $S(t)$ en raison des difficultés liées à la forme des pics dans les signaux de faible résolution SELDI-TOF, néanmoins notre objectif final reste le même, c'est à dire que l'on cherche à extraire les caractéristiques des pics contenus dans le signal $S(t)$. Dans un premier temps ce que nous cherchons donc à faire par un traitement du signal MS c'est d'extraire un signal $S(t)$ propre, puis viendra ensuite le détection des pics et l'extraction de leurs caractéristiques.

Concernant le facteur de normalisation N, celui-ci est très important pour corriger les intensités des pics de $S(t)$. Effectivement, en principe, ces intensités sont seulement liées à la quantité de molécules dans l'échantillon analysé, mais cette dernière est soumise à des aléas dans la préparation des échantillons qu'il est difficile de contrôler en raison de l'intervention humaine. Par exemple, les intensités des pics de $S(t)$ varient sous l'effet des erreurs de dilutions de l'expérimentateur, ou encore sous l'effet d'erreurs de dosage sur les quantités déposées dans l'instrument. Comme ces erreurs influencent l'intensité de tous les pics du spectre de masse, le facteur de normalisation N à pour rôle de capturer la concentration de l'échantillon et de corriger l'intensités des pics pour les rendre comparables d'un spectre à l'autre.

Enfin, la décomposition des spectres de masse selon l'équation 3.1 nous suggère les étapes de traitement du signal à effectuer sur $f(t)$, et nous fournit une indication sur l'ordre dans lequel les réaliser. Effectivement, si on associe une tâche de traitement du signal à chacune des composantes $B(t)$, $\epsilon(t)$, N, la décomposition de l'équation 3.1 nous suggère qu'il faut : 1) commencer par supprimer de $f(t)$ la ligne de base, $B(t)$, et enlever le bruit $\epsilon(t)$; 2) puis normaliser le signal obtenu en déterminant le facteur N. Les méthodes pour réaliser chacune de ces opérations de correction sont discutées dans les section de ce chapitre. Le chapitre qui suit (chapitre 4) sera quant à lui consacré à la détection des pics dans le signal corrigé, $S(t)$ et à l'extraction de leurs caractéristiques (intensité, largeur, aire).

3.2 Ligne de base

Comme nous l'avons vu dans la section 2.1.2, l'analyse d'un échantillon "vide" par spectrométrie de masse ne produit pas un spectre plat d'intensité zéro comme on pourrais s'y attendre. Au contraire une large bosse en début de spectre apparaît en raison de l'ajout de la matrice dans le protocol expérimental. Le problème est que si un pic se trouve dans cette région du spectre, son intensité est biaisée par cette déformation. Le calcul de la ligne de base (qui correspond à la fonction $B(t)$ dans le modèle de l'équation 3.1) à pour but de corriger ce

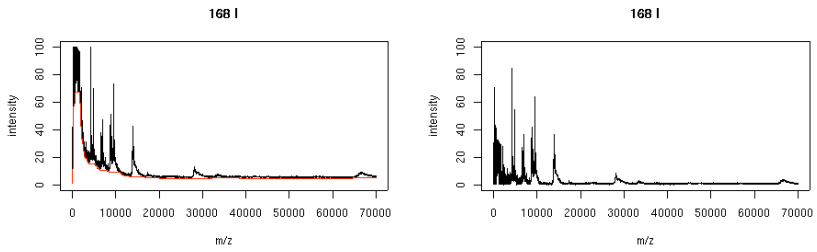


FIGURE 3.1 – A gauche un spectre de masse $f(t)$ et sa ligne de base $B(t)$, à droite le même spectre après soustraction de sa ligne de base $(f(t) - B(t))$.

phénomène par une estimation du zéro théorique en chaque point du spectre. La figure 3.1 montre un exemple de l'effet que doit produire la suppression de la ligne de base sur un signal $f(t)$. On voit que après la suppression, le signal est ajusté sur ligne d'intensité 0 et que l'on peut lire l'intensité des pics directement sur l'axe des ordonnées.

Les raisons de l'apparition de la ligne de base proviennent de l'électronique du spéctro-mètre de masse qui arrive a saturation lorsque qu'un trop grand nombre d'ions affluent sur le détecteur qui les comptes. Un effet "d'éblouissement" de l'appareil se produit alors est provoque l'apparition de la ligne de base [82]. Il faut effectivement savoir que l'électronique du détecteur a besoin d'un petit temps de latence pour revenir à son état neutre lorsqu'il reçoit un ion. Malheureusement lorsque le flux d'ions est trop important, les ions se succèdent trop rapidement pour que le capteur ait le temps de revenir à son état neutre. Les ions arrivés au temps t influencent alors la mesure effectuée au temps $t + 1$, et les valeurs s'accumulent petit à petit. Ce sont ces observation qui poussent Malyarenko et al. [74] à modéliser le signal $f(t)$ récursivement par l'équation 3.2.

$$f(t) = S(t) + c + a \sum_{k=0}^{t-1} e^{-k/\tau} S(t-k) \tag{3.2}$$

Cette formulation modélise l'intensité du spectre de masse observé à un temps t $(f(t))$ comme étant l'intensité du signal théorique au même instant $(S(t))$ auquel s'ajoute une partie des intensités observées aux temps précédents. Ces dernières sont pondérées par une fonction à décroissance exponentielle selon l'ancienneté de la mesure. Les paramètres a et τ contrôlent la rapidité de la décroissance exponentielle, et c est un paramètre d'offset du signal. A partir du modèle de signal définie par l'équation 3.2, Malyarenko et al. déduisent un modèle de ligne de base donnée sur l'équation 3.3 :

$$B(t) = c + a \sum_{k=0}^{t-1} (1-a)^k e^{-k/\tau} (f(t-k) - c) \tag{3.3}$$

L'ennui est que la formule est gouvernée par trois paramètres (a, c et τ) qu'il faut ajuster manuellement à chaque changement de configuration de la machine.

Avant ces travaux, la ligne de base restait difficile à caractériser. On notait simplement deux choses : 1) elle montre une distorsion plus importante dans les régions de faible m/z car les ions majoritaires sont les ions de petites masses issues des molécules de la matrice [32, 48] ; 2) sa fréquence est bien inférieure à celle des pics et du bruit [106, 39]. Le premier point est considérer par [79] lorsque qu'il modélise la ligne de base par une fonction à décroissance exponentielle dans son spectromètre virtuel de manière à ce que son intensité dans les régions de faible m/z soit plus importante. Sinon, les autres approches proposées pour supprimer la ligne de base reposent sur la deuxième caractéristique, et consistent à appliquer un filtre dit *passe haut* pour supprimer les basses fréquences du signal. On distingue ensuite parmi elles les approches fréquentielles des approches temporelles.

Les *approches fréquentielles* projettent le spectre de masse vers un espace fréquentiel par transformé de Fourier ou ondelettes pour y supprimer les basses fréquences, puis reviennent à l'espace initial par une transformée inverse. Cependant ce type de filtre sont plutôt adapté au lissage du signal, et ils doivent être utilisé avec prudence dans le cadre de la ligne de base [106, 6]. Par exemple, en ce qui concerne les ondelettes, il faut préférer les méthodes continues aux méthodes discrètes [29, 100].

Les *approches temporelles* quand à elles, agissent directement sur les données : elles évaluent le niveau de la ligne de base en un point t du spectre en considérant ses n points voisins. Plus n est grand et plus on capture de basses fréquences. Par exemple [4] prend pour ligne de base en chaque point du signal la valeur minimum des n points l'entourant (voir figure 3.5). Ce filtre à également la propriété de construire une ligne de base située en dessous du signal original ($B(t) \leq f(t)$), or c'est à cet endroit que l'on s'attend à la trouver [35, 82], car le bruit qui s'ajoute aux spectres est essentiellement positif ($\epsilon(t) \geq 0$). Le logiciel *Ciphergen ProteinChip Software* extrait lui aussi une ligne de base en dessous du signal original, qu'il trouve en découpant le signal et en recherchant l'enveloppe convexe du signal dans chaque région [32]. Remarquez aussi que le calcul de la ligne base selon l'équation 3.3 de [74] peut être considéré comme une approche temporelle car le calcul de $B(t)$ se fait en considérant le signal $f(t)$ situé dans le voisinage en amont du point t, et l'étendu du voisinage est contrôlé par le paramètre τ.

La difficulté avec ce type d'algorithme est de fixer le paramètre n qui définit le nombre de voisins à considérer pour le calcul de la ligne de base [48]. Il doit être assez grand pour que la ligne de base ne soit pas influencée par les pics du spectre, et suffisamment petit pour capturer la forme de la ligne de base. De plus, ce paramètre est d'autant plus difficile à régler que, dans les spectres MALDI-TOF et SELDI-TOF, la largeur des pics varie selon leur position dans le spectre : les pics sont plus large en fin de spectre qu'en début. C'est pourquoi le logiciel *Ciphergen ProteinChip Software* varie dynamiquement le nombre de voisin à considérer pour le calcul de la ligne de base en fonction de la position dans le spectre.

3.2.1 Ligne de base par approximation linéaire en k morceaux

Pour calculer la ligne de base d'un spectre de masse en évitant les problèmes listées ci-dessus, nous avons dans un premier temps pensé à mettre en oeuvre une méthode pour rechercher la meilleure approximation du signal – au sens des moindres carrés – par une fonction linéaire en k morceaux, avec des morceaux de taille inconnue et pas forcement de longueur égale. L'un des avantage de cette approche est qu'elle ne possède qu'un seul paramètre k, assez intuitif et donc facile à régler par les utilisateurs (sans-doute, est-il même possible de le choisir automatiquement en étudiant l'évolution de l'erreur d'approximation lorsque k varie). Son autre avantage est qu'elle devrait être mieux adaptée à des signaux dont la résolution varie. Effectivement, comme l'on cherche à approcher au mieux le signal de départ par k segments de longueurs variable, on évite les difficultés rencontrés pour fixer la taille des fenêtres glissantes dans les approches temporelles. L'ennui, c'est que ce problème de rechercher les k segments de longueurs variables qui s'ajuste au mieux sur le signal de départ – au sens des moindres carrés – est difficile à résoudre. Il existe cependant plusieurs méthodes pour approcher la solution [73], comme par exemple au moyen d'une optimisation par un algorithme génétique [88], pourtant ces solutions sont inutilement complexe pour rechercher la ligne de base des spectres de masse. De plus, elles ont un autre inconvénient qui est qu'elle n'extraient pas nécessairement une ligne de base située en dessous du signal original, comme on l'aimerait, et comme le font le filtre minimum de [4] et celui du logiciel Ciphergen ProteinChip [32].

Nous proposons cependant une alternative simple qui combine tous les avantages, c'est à dire qui 1) génère une ligne de base sous le signal, 2) est de complexité faible 3) génère une ligne de base linéaire en k segments de taille différentes. Cet algorithme commence en considérant que la ligne de base $B(t)$ est identique au signal original $f(t)$, puis il élimine tour à tour de $B(t)$ la valeur d'échantillonage tel que l'approximation linéaire qui en résulte soit la plus proche possible du signal original – au sens des moindres carrés – tout en restant au dessous de lui. L'itération s'arrête lorsque on ne peut plus enlever de points ou lorsque $B(t)$ contient les k morceaux demandés. Au final, on obtient une enveloppe du signal de départ située au dessous de lui.

La figure 3.2 illustre une étape dans le processus itératif que nous venons de décrire et qui abouti à l'estimation de la ligne de base. A cette étape, la ligne de base est constituée de 11 segments, et il nous faut décider du point à éliminer pour produire la ligne de base à 10 segments. Parmi les 10 points candidats à l'élimination, certains ne peuvent pas être supprimer car sinon la ligne de base générées passerait au dessus du signal. Par exemple, le point C ne peut être supprimé, car le segment de ligne de base généré joindrait les points B et D en coupant le signal. Par contre, il nous reste le choix entre l'élimination des points B, D et G, qui ne violent pas cette contrainte. A l'évidence, pour que la ligne de base s'écarte le moins possible du signal original, il faut éliminer le point tel que l'aire du triangle dont il est le sommet soit la plus faible possible. C'est à dire B sur l'exemple de la figure car son triangle à la plus petite des trois aires.

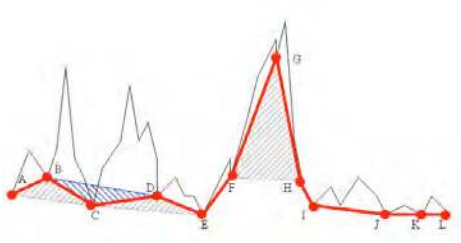


FIGURE 3.2 – Illustration d'une étape dans le processus itératif qui permet d'estimer la ligne de base par une fonction linéaire en k morceaux. La ligne rouge représente l'estimation actuelle de la ligne de base en 11 morceaux. Les triangles hachurés représentent l'erreur d'approximation qui s'ajoute si on supprime le point située en leur centre. Le point C ne peut être supprimé car la ligne de base obtenue couperait le signal original.

L'algorithme 2 implémente cette idée avec une complexité n^2 (où n est le nombre de points dans le spectre), mais en introduisant une file d'attente avec priorité, on peut ramener sa complexité à un niveau plus acceptable de $\mathcal{O}(n \log n)$. L'algorithme initialise la ligne de base avec le signal original, et utilise l'aire du triangle formée par trois points consécutifs pour déterminer celui dont l'élimination à le moins d'impact sur la détérioration. Il utilise également la fonction is.ccw (algorithme 1) pour s'assurer que la suppression du point génére une ligne de base située au dessous l'approximation précédente.

Algorithm 1 is.ccw(A,B,C) : est vrai si les trois points (A,B,C) du plan sont disposés selon le sens de rotation des aiguilles d'une montre

1: **return** $(B.x - A.x)(C.y - B.y) - (B.y - A.y)(C.x - B.x) > 0$

Algorithm 2 LLE(f[],k) : Calcule l'enveloppe linéaire inférieure du signal f[] (Linear Lower Envelop) qui s'ajuste le mieux possible au signal, et qui contient k points.

1: initialisation : B[] = f[]
2: **while** card$(B) > k$ **do**
3: Trouver l'indice i qui minimise l'aire du triangle $(B[i-1], B[i], B[i+1])$ et tel que *is.ccw*(B[i-1],B[i],B[i+1])
4: **Stop** si il n'existe pas de tel i
5: Supprimer B[i]
6: **end while**
7: **return** B[]

Les quatre graphiques de gauche dans la figure 3.3 montrent des exemples de lignes de base calculées avec l'algorithme 2 lorsque le paramètre k, qui contrôle

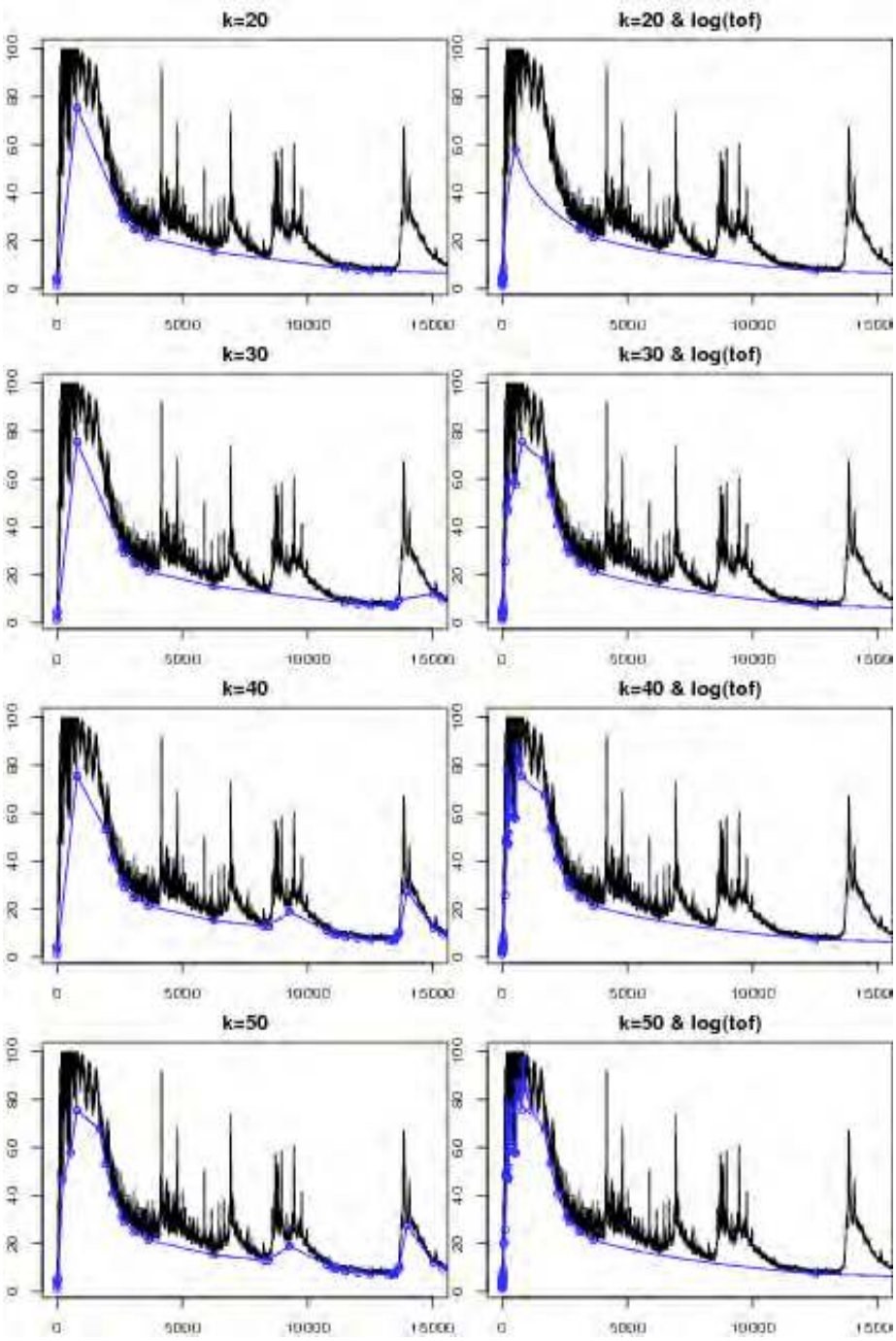


FIGURE 3.3 – Les lignes de bases calculées par l'algorithme 2. Dans la colonne de gauche, les lignes de bases sont calculées sur le signal original, dans la colonne de droite elles sont calculées après une transformation logarithmique de l'axe des abscisses.

le nombre de segments dans la ligne de base, varie. Les résultats obtenus sont assez satisfaisants, mais souffrent tout de même de quelques problèmes : lorsque $k \geq 30$, la ligne de base suit trop fidèlement la forme du signal sous les pics situées à 10kDa et 15kDa, et à l'inverse lorsque $k \leq 30$ la ligne de base ne suit pas assez le signal dans la région 0-2000Da. De plus, en ce qui concerne l'ajustement du paramètre k que notre approche était censée faciliter, on voit qu'au contraire il est au moins aussi difficile à régler que le nombre de voisins dans les approches temporelles. En effet, la forme de la ligne de base est très sensible aux petites variations du paramètre k, même si pour régler ce paramètre on peut s'aider des courbes d'erreurs (figure 3.7) pour voir que le k idéal se situe autour de $k = 20$ comme nous allons en discuter un peu plus bas.

Ces problèmes d'ajustement dont souffre l'algorithme 2 apparaissent car il accorde la même importance au points situés en fin de spectres qu'aux points situés en début de spectres. Pourtant, on sait que la ligne de base est plus importante en début de spectre qu'en fin de spectre, et donc elle devrait s'ajuster plus fidèlement au signal dans les régions de faible m/z que dans les régions de fort m/z. Afin de corriger ce problème, nous avons cherché à accorder un poids plus important à cette partie du signal. Pour ce faire, plutôt que d'utiliser les coordonnées m/z des points du signal, nous avons utilisé le logarithme de leur temps de vol (en réalité nous avons utilisé le logarithme de l'index des points comme estimation du temps de vol), car comme le font remarquer [116] et [48], cette transformation produit un signal plus uniforme, où les pics ont des largeurs plus homogènes et la ligne de base prend toute son importance. La figure 3.4 montre en exemple le spectre que l'on obtient lorsque l'on transforme le spectre de masse de la figure 3.3 par cette opération. On voit clairement que cette transformation a eut pour effet d'étendre les points de faible m/z sur une région d'abscisse beaucoup plus large, et au contraire de compacter les points dont le m/z était élevé. Ainsi l'aire sous la courbe formée par les points de m/z faible prennent plus d'importance, et ainsi notre algorithme leur accordera plus de poids. Pour le vérifier, l'algorithme 2 a été appliqué pour calculer l'enveloppe du signal dont l'abscisse à été transformé en logarithme du temps de vol. Les lignes de base que nous avons obtenus en faisant varier le paramètre k sont données par les quatre lignes de bases à droite dans la figure 3.3. Cette fois le résultat est plus satisfaisant, et l'on voit qu'effectivement la ligne de base s'ajuste plus fidèlement aux régions de faible m/z qu'aux régions de grand m/z, et sa forme est moins sensible au paramètre k. C'est la méthode que nous recommandons pour le calcul de la ligne de base. Malheureusement, cet algorithme n'est utilisée que pour traiter les données que nous utilisons dans la première partie du document (sur le pré-traitement), les données utilisées dans la deuxième étant générées avant l'existence de l'algorithme.

Dans la deuxième partie de ce document – concernant les méthodes d'apprentissage pour l'extraction des biomarqueurs –, nous avons utilisé le filtre temporelle proposé par [12] pour calculer la ligne base. Celui-ci aussi produit une ligne de base située en dessous du signal original ($B(t) \leq f(t)$), et le résultat qu'il génère est très similaire à celui obtenu avec le filtre minimum de [4] ou avec le logiciel *Ciphergen ProteinChip* [32], mais il a l'avantage d'épouser mieux le

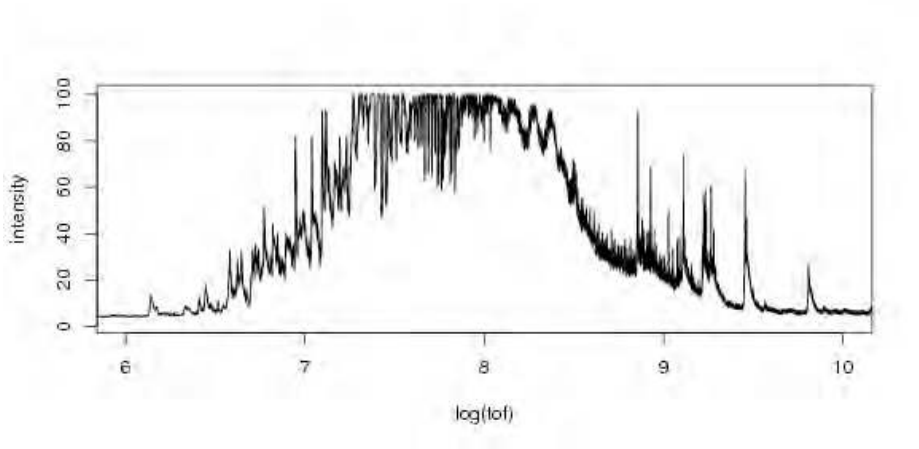


FIGURE 3.4 – Forme du spectre obtenus en remplaçant l'axe m/z par le logarithme du temps de vol.

signal et par conséquent de produire plus de valeurs nulles que ces deux la. L'opérateur utilisé pour cela est connu sous le nom d'*opérateur d'ouverture* en mathématique morphologique. Il consiste en une application successive de deux filtres : un premier qui cherche pour chaque point du signal $f(t)$ le minimum de ses n voisins et produit un signal $f_1(t)$ (identiquement à la ligne de base de [4]) ; le second extrait de $f_1(t)$ les maximums des n voisins pour produire la ligne de base $B(t)$. En d'autre termes :

$$f_1(t) = \min_{-n/2 \leq i \leq n/2} f(t+i) \tag{3.4}$$

$$B(t) = \max_{-n/2 \leq i \leq n/2} f_1(t+i) \tag{3.5}$$

La figure 3.5 illustre son fonctionnement, et montre bien que la seconde opération permet à la ligne de base de mieux épouser le signal original. Le résultat obtenu après suppression de la ligne de base par cette méthode, est aussi connu comme étant le résultat d'un filtre top-hat en mathématique morphologique.

3.2.2 Comparaison des méthodes de calcul de ligne de base :

Essayons de comparer les différentes méthodes de calcule de ligne de base et tâchons de fixer leurs paramètres. Pour commencer faisons le rapprochement entre le paramètre k qui contrôle le nombre de segments dans l'algorithme 2 (LLE), et le paramètre n qui contrôle le nombre de voisin dans le filtre top-hat. Si l'on suppose que les segments de LLE ont des longueurs égales, alors chaque segment couvre $n = l/k$ points du signal, où l est le nombre total de points

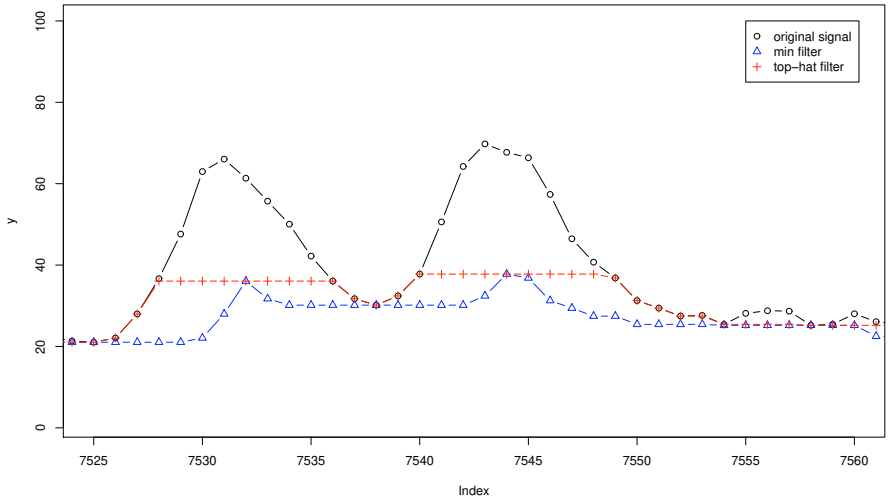


FIGURE 3.5 – Opérateur d'ouverture pour $n = 9$ voisins. En rond noir, le signal original $f(t)$. En triangle bleu le résultat du filtre minimum (résultat intermédiaire $f_1(t)$), qui est la ligne de base pour [4]. En croix rouge le résultat de l'opérateur d'ouverture : le maximum des minimums (résultat final $B(t)$).

dans le signal. Ayant observé cette relation, nous pouvons tenter de comparer les lignes de base calculées par l'algorithmes LLE pour un k fixé, avec les lignes de bases calculées par le filtre top-hat avec $n = l/k$.

La figure 3.6 montre les lignes de base calculées avec un filtre top-hat lorsque k varie, et peuvent être comparées à celles de la figure 3.3 obtenus avec l'algorithme LLE. Noter qu'ici la transformation logarithmique des abscisses n'aurait aucun effet sur le résultat du filtre top-hat, et donc nous ne présentons que les résultats sur le signal original. Avec cet algorithme, pour avoir des lignes de base qui s'ajuste plus fidèlement aux données de faible m/z qu'aux données de grand m/z, il faudrait adopter des fenêtres glissantes dont la taille augmente avec le m/z (plus difficile à mettre en oeuvre). Les lignes de bases de la figure 3.6, sont assez satisfaisantes pour $k = 20$ et $k = 30$, et trop proche du signal original si k est supérieur. Sinon, elles ressemblent assez à ce que l'on obtient avec LLE sans transformation logarithmique des abscisses. Afin d'observer plus en détail l'influence du paramètre k sur les résultats des différents algorithmes de calcul de lignes de base, on peut voir sur la figure 3.7 l'évolution de l'erreur entre la courbe de la ligne de base et le spectre de masse original lorsque k varie. Ces courbes montrent des comportements assez différents de algorithme LLE et du filtre top-hat. Le premier obtient une courbe en escalier avec une cassure autour de $k = 20$, alors que le second a une décroissance moins brutale qui colle plus ra-

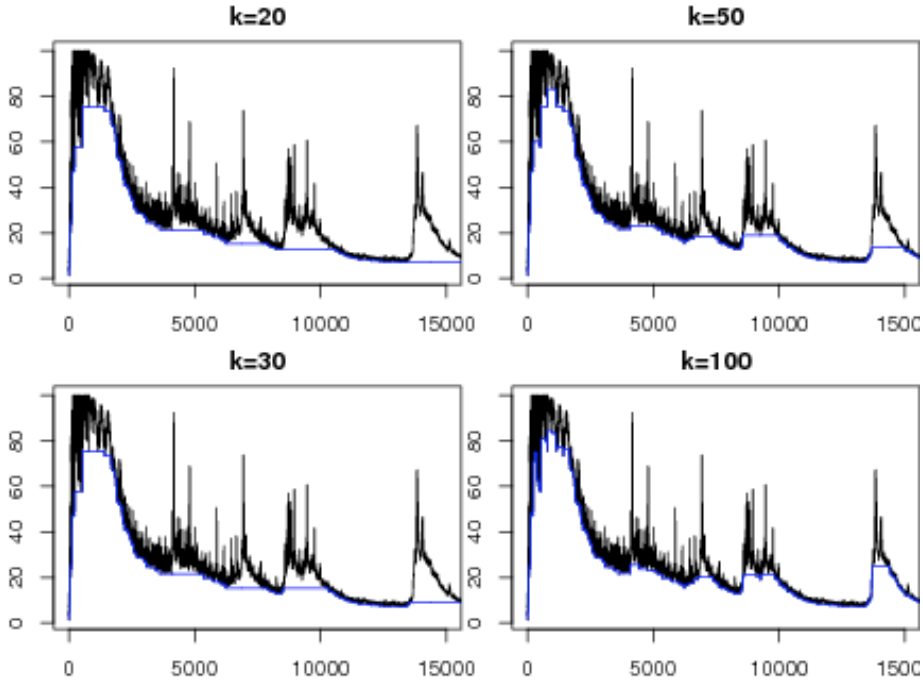


FIGURE 3.6 – Lignes de bases calculées par un filtre top-hat.

pidement au signal (quand $k = 10$ l'erreur du filtre top-hat est déjà relativement faible, alors que l'erreur de LLE est toujours très haute).

L'apparition des l'escaliers dans l'algorithme LLE est intéressante. Effectivement, le seuil $k = 20$ ou il apparaît correspond au nombre minimum de segments nécessaires pour commencer à avoir une bonne estimation du signal. Ce seuil donne donc une indication pour régler le paramètre k de l'algorithme : il faut le choisir légèrement au dessus de ce seuil, par exemple $k = 30$. Avant le seuil, il reste trop d'imprécisions dans la ligne de base, et si on s'éloigne trop du seuil, la ligne de base risque de trop coller au signal. Au final, en suivant ces observations, nous pourrions envisager d'ajuster automatiquement le seuil, k, de l'algorithme LLE pour fournir une méthode sans paramètre. Dans le cas du filtre top-hat, comme l'erreur diminue de manière plus continue, une telle approche est plus difficile à envisager.

Signalons enfin que dans la deuxième partie de la thèse nous avons utilisé l'algorithme top-hat comme algorithme de calcul de ligne de base lors de la génération des jeux de données (et non pas LLE). La valeur de paramètre que nous avons utilisé avec cet algorithme est $n = l/k = 1/20$ème du nombre total de points dans le spectre. Cette valeur apparaît effectivement bien adapté au vu des figures 3.7, 3.6, et des tests réalisés sur nos jeux de données. De plus, nous

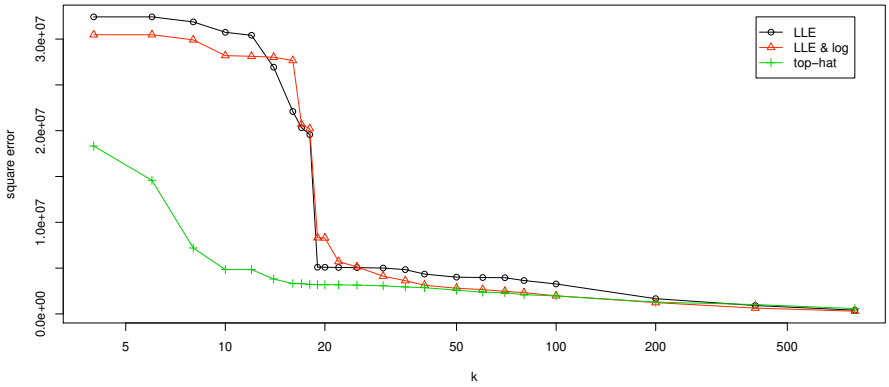


FIGURE 3.7 – Évolution de l'erreurs d'approximation entre les lignes de base $B_k(t)$ calculées par les différents algorithmes et le signal original $f(t)$ lorsque le paramètre k qui contrôle l'ajustement varie. Dans le cas de l'algorithme 2 (LLE), k correspond au nombre de segments dans l'approximation du signal. Dans le cas du filtre top-hat k correspond à la taille de la fenêtre glissante exprimé comme une fraction du nombre de points dans le signal. par exemple $k = 20$ correspond à une taille de fenêtre égale à 1/20ème du nombre de points dans le signal.

verrons dans le chapitre suivant que l'élimination de la ligne de base n'a qu'une influence mineure sur l'algorithme de détection de pics que nous proposons, et qui est lui le point central du pré-traitement.

3.3 Le traitement du bruit dans les spectres de masse

Le bruit est une composante de très haute fréquence et de faible intensité dans les spectres de masse dont l'origine est électronique et chimique [39] – dans le modèle de l'équation 3.1, il est matérialisé par la fonction $\epsilon(t)$ –. L'origine chimique du bruit est essentiellement dut à la complexité du processus d'ionisation durant lequel des molécules de la matrice et/ou de l'analyte se fragmentent de manière inattendue et s'ionise. Les fragments produits ont une masse plus ou moins aléatoire, et ils parasitent le signal des spectres de masse [82, Section 2.3.11]. Il peut aussi arrivée que des particules chimiques (de la poussière), externe à l'expérience, se déposent sur les surfaces SELDI et bruite le signal, si bien que certains protocoles expérimentaux incluent un rinçage des plaques [52] avant leur utilisation. Quant à l'origine électronique du bruit, elle est liée à la sensibilité des capteurs qui détectent les ions et à l'électronique qui amplifie le

courant. Nous pouvons d'ailleurs en partie observer cet aspect du bruit en nous reportant à la figure 2.2 qui montre les spectres de masse SELDI-TOF obtenus après différentes phases du protocole expérimentale qui analyse un échantillon "vide". Il y a dans ces données du bruit, et il est même déjà présent avant l'ajout de la matrice dans le protocole expérimental, a cette étape l'origine du bruit est essentiellement électronique.

Le bruit ayant essentiellement un comportement aléatoire, on le modélise généralement par un processus stochastique. Pour Morris et al. [79], par exemple, le bruit est une valeur aléatoire qui s'ajoute au signal, et qui suit une distribution normal dont l'écart type évolue de façon continue le long des spectres ($\epsilon(t) \sim \mathcal{N}(0, \rho^2(t))$). Cette modélisation suppose que des valeurs négatives s'ajoutent au signal, or ce n'est pas cohérent avec notre façon de calculer la ligne de base en dessous du signal car cela présuppose que le bruit est exclusivement positif. De plus, Djafari et al. [77] soulignent que l'hypothèse d'une distribution gaussienne pour le bruit pourrais ne pas être valide car ce qui est mesuré est proportionnel au nombre d'ions. Une distribution de Poisson, ou une distribution gamma, pourrais donc être plus adaptée pour définir le bruit. Malgré ces difficultés de modélisation, plusieurs méthodes ont été proposé pour s'abstraire du bruit qui perturbe la détection des pics, car les motifs qu'il contient sont similaires aux motifs que l'on retrouve au niveau des pics. Parmi ces méthodes de traitement du bruit, on distingue celles qui tentent d'opérer un lissage du signal de celles qui se contentent d'estimer le niveau du bruit.

3.3.1 Elimination du bruit par lissage du signal

Le lissage à pour but d'éliminer les variations du spectre de masse qui correspondent au bruit, pour ne conserver que le signal sous-jasent. Cette opération est délicate car elle altère le signal original et peut donc potentiellement détruire l'information qu'il contient. Pour les massistes, qui travaillent avec des spectre de haute résolution, il est notamment important de veiller à ce que le lissage conserve la position des pics en m/z, leur forme (et leur intensité). Le travail de [70] est particulièrement révélateur des ces difficultés. Son approche de lissage par un filtre de Stavisky-Golay est soupçonnée de biaisé d'avantage les intensités des pics de faible hauteur (plus proche du bruit) que les intensités des pics de grande hauteur. Cependant, si la résolution est bonne, il est possible de s'aider des distributions isotopiques des pics du signal pour mieux distinguer le bruit du signal et effectuer le lissage au mieux [3]. Dans le cas contraire qui nous concerne plus particulièrement avec les instruments SELDI-TOF, les spectres ont une faible résolution et ne permettent pas de distinguer les isotopes. Dans ce cas, la transformé du signal en ondellettes [106] apparaît comme une technique de débruitage des plus efficace pour les spectres de masse [60, 21, 79, 6, 69, 71, 29, 100, 46, 80, 67]. On peut également songer à utiliser l'approximation du signal par des courbes *spline* [53], ou encore des méthodes à noyau [45, 84].

Les techniques que nous venons de mentionner sont en mesure d'opérer un lissage individuellement sur les spectres de masse, mais parfois, nous disposons

de plusieurs spectres et il peut s'avérer plus efficace de fusionner leurs informations pour en extraire le signal sous-jasent. Trois cas peuvent se présenter :

1. Les spectres ont été obtenus en répétant plusieurs fois la même expérience sur le même échantillon biologique et ils sont censés renfermer la même information. Pour ce cas, Pratapa et al., [94], propose une méthode intéressante avec un modèle probabiliste à base de chaînes de Markov. A partir de ce modèle, il est capable d'extraire le spectre de masse sous-jasent le plus probable qui a généré les différents répliquas de l'expérience tout en prenant en compte le bruit à la fois selon l'axe des m/z (relatifs aux problèmes de calibration des instruments) et des intensités.
2. Les spectres ont été obtenus à partir du même échantillon mais dans des conditions expérimentales légèrement différentes. Par exemple, Andreev et al. [3] adresse spécifiquement le problème de l'élimination du bruit dans les expériences de LCMS dont les spectres de masse produits à des temps de rétention proches se ressemble beaucoup. Il préconise d'opérer une réduction de bruit selon la dimension chromatographique (c'est à dire en comparant les spectres les uns aux autres) avant d'opérer une analyse des données spectrales. Sa méthode de lissage quand à elle agit sur la transformé de Fourier des signaux qu'il ajuste en fonction des caractéristiques fréquentielles du bruit. Mentionnons également Li et al. [70] qui essai aussi d'exploiter les redondances d'information dans les différents spectres d'une expérience LCMS pour réduire le bruit sur les intensités des pics.
3. Enfin, les spectres peuvent aussi provenir d'échantillons biologiques différents mais acquis selon le même protocole expérimental. Dans ce cas, les données sont également corrélées dans une certaine mesure, mais il est un plus ambitieux de chercher à exploiter ces corrélations pour réduire le bruit. Il est en effet, probable que les pics apparaissent aux même positions dans les spectres, mais la régularité à laquelle cela se produit est difficile à modéliser. Pourtant Morris et al., [79], envisage une idée simple dans le but de réduire le bruit et de détecter les pics présents dans les différents spectres de masse SELDI-TOF : il fait la moyenne des n spectres en sa possession pour obtenir un spectre moyen ou le bruit est réduit d'un facteur $\sqrt{n}$ et y opère la détection des pics. En procédant ainsi, les pics qui apparaissent régulièrement dans les spectres, se retrouveront forcement dans le signal moyen, en revanche, les pics qui sont présents dans très peux de spectres risque bien d'être éliminé. On retrouve aussi une idée proche dans le travail de Randolph et al. [100].

3.3.2 Estimation du niveau de bruit dans un spectre

A l'inverse des méthodes de lissage dont le but est de nettoyer le signal pour faciliter la détection de pics, les méthodes d'estimation du niveau de bruit se contentent d'estimer l'intensité du bruit en chaque point du signal sans altérer le signal d'origine. Cependant, l'essentielle de ces méthodes intègrent un lissage du signal dans leur processus afin d'estimer les variations du signal brute

par rapport au signal espéré [48]. Dans ces méthodes, l'estimation de la variation de l'intensité est faite localement dans une fenêtre glissante avec une méthode de préférence robuste aux valeurs extrêmes (*outliers*) qui sont susceptibles d'apparaître dans les régions du signal qui montrent des pics [82, 79, 19]. Par exemple, Morris et al., [79], lisse le signal avec une décomposition en ondelettes qu'il soustrait du signal brute, et en estime le niveau de bruit grâce à la *déviation absolue médiane*[1] calculée dans une fenêtre glissante. Un des problèmes de ce type d'approches, est qu'elles nécessitent de paramètrer à la fois l'algorithme de lissage, et la taille de la fenêtre glissante. En comparaison, la méthode que nous proposons dans la suite est capable d'estimer le niveau de bruit sans lisser le signal ce qui la rend plus facile à paramétrer. On s'inspire en partie dans notre travail du travail de Coombes et al. [19], qui propose pour estimer le bruit d'utiliser la médiane de la première différence entre les intensités successives des spectres de masse, ce qui nécessite moins de paramètres à ajuster.

3.3.3 Estimation du bruit par écart d'intensité entre points de retournement

Pour s'abstraire du bruit, notre préférence est allé vers les méthodes d'estimation du niveau de bruit plutôt que vers les méthodes de lissage pour éviter une détérioration du signal qui pourrais détruire l'information qu'il contient. Cette exigence n'est certes pas essentielle pour les spectres de masse de type MALDI/SELDI, qui nous concernent prioritairement dans ce travail, car leur résolution est plutôt faible et ils sont naturellement assez fluctuants. Il y a donc peu de chance de perdre de l'information dans le lissage du signal. Cependant, en préférant les méthodes d'estimation du niveau de bruit, nous pouvons envisager d'étendre nos idées aux données de haute résolution pour lesquelles le lissage est d'avantage susceptible de détruire l'information. Nous avons aussi cherché à définir un algorithme facile à paramétrer pour les utilisateurs par rapport aux méthodes classiques mentionnées plus haut.

Notre objectif est, rappelons-le, d'estimer les variations de haute fréquence du signal. Pour cela, nous décidons de nous concentrer sur les différences d'intensité entre les maxima et les minima du spectre de masse, c'est à dire les points du spectre où la courbe passe de croissante à décroissante et vice-versa – à la différence de [19] qui se concentre sur les différences d'intensité entre deux points successifs du spectre même s'il ne s'agit pas de maxima/minima. Ces maxima et minima alternent dans la mesure ou entre deux points maxima (resp. minima) successifs, il y a forcement un point minimum (resp. maximum), de plus ces optimums qui se succèdent sont étroitement liées aux variations de haute fréquence que nous cherchons à estimer. Nous décidons donc de prendre comme estimation du niveau de bruit en un point p du spectre la différence d'intensité

1. la déviation médiane absolue (MAD), est la médiane des écarts à la médiane :

$$\mathrm{MAD}(\mathbf{x}) = \mathrm{median}(|x_i - \mathrm{median}(\mathbf{x})|)$$

médiane entre les optimums successifs (maxima et minima) situées autour de p. Le nombre d'optimum situé autour de p à considérer est fixé par l'utilisateur, et constitue le seul paramètre de notre méthode. Dans les expériences qui suivent nous avons fixé ce paramètre à 500 (250 de chaque coté).

L'avantage de notre approche par rapport aux approches existantes est certes faible, mais son principal intérêt réside dans la définition du paramètre qui définie le voisinage. Généralement la taille du voisinage d'un point p est exprimé en dalton, en temps de vol, ou en nombre de valeur d'échantillonnage, or ces valeurs dépendent en partie des conditions d'acquisition des spectres. Dans notre approche, le voisinage est exprimé en nombre d'optimum, et on peut espérer que cette valeur qui caractérise les mouvements de haute fréquence des spectres de masse soit mieux adaptées aux variations des conditions d'acquisition. De plus, au niveau des pics, le signal est plus stable que dans les autres régions du spectre de masse, et il peut s'avérer judicieux d'élargir le voisinage au niveau des pics pour estimer le bruit. Cette stabilité du signal, se caractérise par moins de maximum et de minimum sur les pentes des pics que dans le reste du signal, et dans un tel contexte, notre approche à le bon comportement puisqu'elle se voit dans l'obligation d'augmenter son voisinage. Pour ces raisons, l'approche semble attractive, mais il reste à vérifier que le niveau de bruit qu'elle calcul est cohérent avec les autres approches d'estimation du niveau de bruit.

Validation de la méthode

Pour vérifier que notre approche génère des niveaux de bruits cohérents par rapport à une approche acceptée pour l'estimation du bruit, nous procédons à une comparaison de leurs résultats. Pour cela, nous utilisons des spectres de masse MALDI-TOF produits dans différentes conditions expérimentales. Les données proviennent des expériences décrites dans notre article [131], qui emploies trois types de matrice (un des principal constituant du bruit) : *a-cyano-4-hydroxy-cinnamic* (ACCA-UF5kD), *5-dihydroxybenzoic* (DHB-UF10kD), *sinapinic* (SPA-UF10kD). Pour chacune des trois conditions expérimentales, deux spectres sont testés : un spectre blanc censé ne contenir aucun pic car réalisé à partir d'un échantillon vierge, et un spectre classique produit à partir d'un échantillon d'urine. Le niveau de bruit calculé par notre méthode (*delta min/max*), est comparé au niveau de bruit calculé par une approche plus classique qui consiste en : 1) lisser le signal avec un filtre "moyenneur" à fenêtre glissante (de taille 100), et 2) estimer le niveau de bruit en calculant la déviation standard du signal brute par rapport au signal lissé (dans une fenêtre glissante de taille 500). Les résultats obtenus avec ces deux méthodes sont fournis sur les figures 3.8, 3.9 et 3.10, c'est à dire une figure par condition expérimentale.

Ces figures montrent que les niveaux de bruit calculés avec les deux méthodes sont très similaires : quelques soient les conditions expérimentales, les courbes bleues et les courbes rouges se superposent presque. La courbe de la déviation standard (en rouge) montre cependant de petites bosses aux endroits où il y a des pics dans le spectre alors que notre méthode (la courbe bleue) est plus monotone (voir par exemple l'agrandissements sur la figure 3.8). Comme nous l'avons

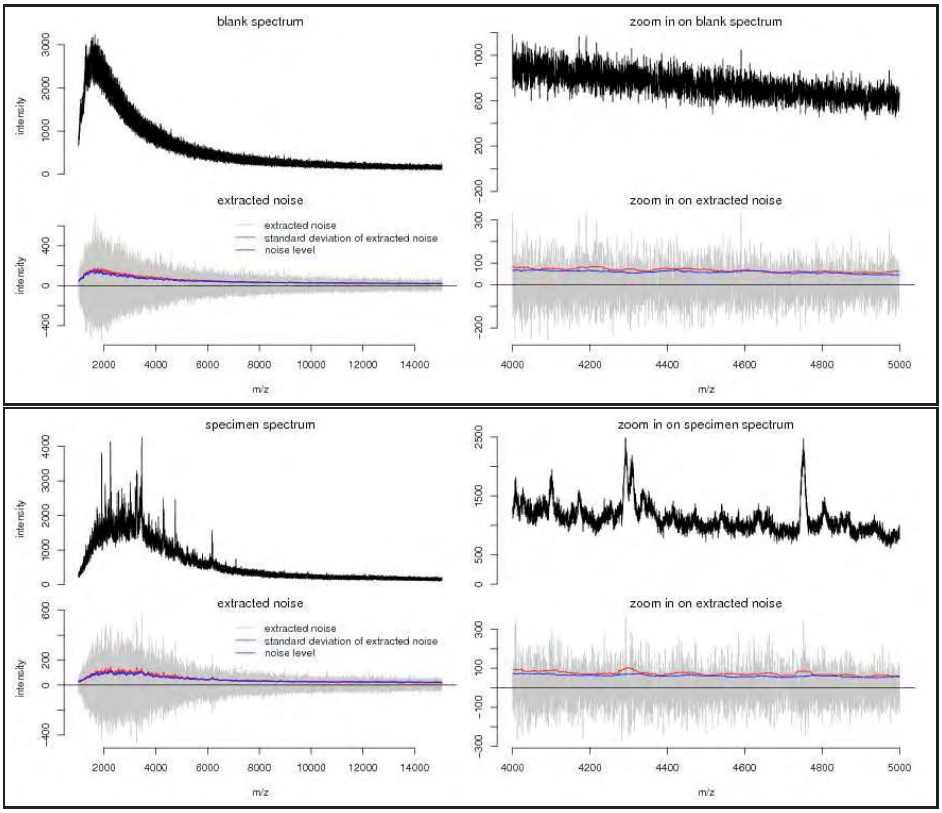


FIGURE 3.8 – Résultat de l'analyse du niveau de bruit sur les deux spectres MALDI-TOF réalisés avec une matrice *a-cyano-4-hydroxy-cinnamic*, avec agrandissement sur la région 4000Da-5000Da. Le spectre de masse est matérialisé par la ligne noire. La ligne grise et la ligne rouges sont respectivement le bruit restant lorsque le signal lissé est enlevé du signal brute, et le niveau de bruit estimé par déviation standard. La courbe bleue est le niveaux de bruit estimé par notre méthode directement sur le signal original.

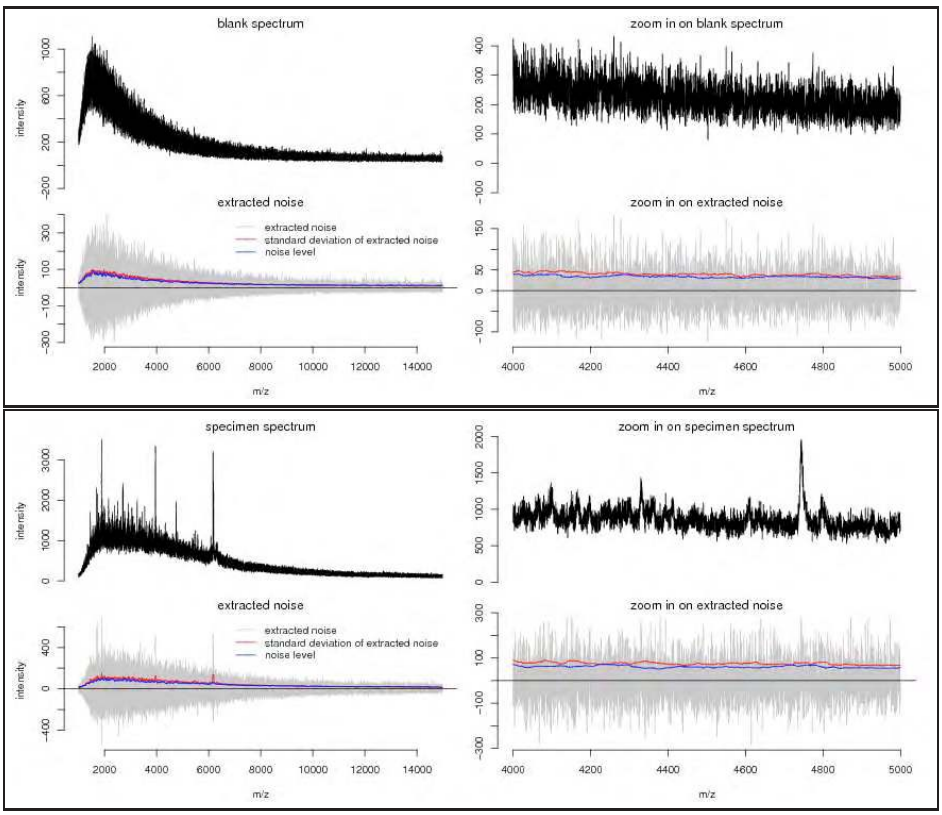


FIGURE 3.9 – Résultat de l'analyse du niveau de bruit sur les deux spectres MALDI-TOF réalisés avec une matrice *5-dihydroxybenzoic*, avec agrandissement sur la région 4000Da-5000Da. Le spectre de masse est matérialisé par la ligne noire. La ligne grise et la ligne rouges sont respectivement le bruit restant lorsque le signal lissé est enlevé du signal brute, et le niveau de bruit estimé par déviation standard. La courbe bleue est le niveaux de bruit estimé par notre méthode directement sur le signal original.

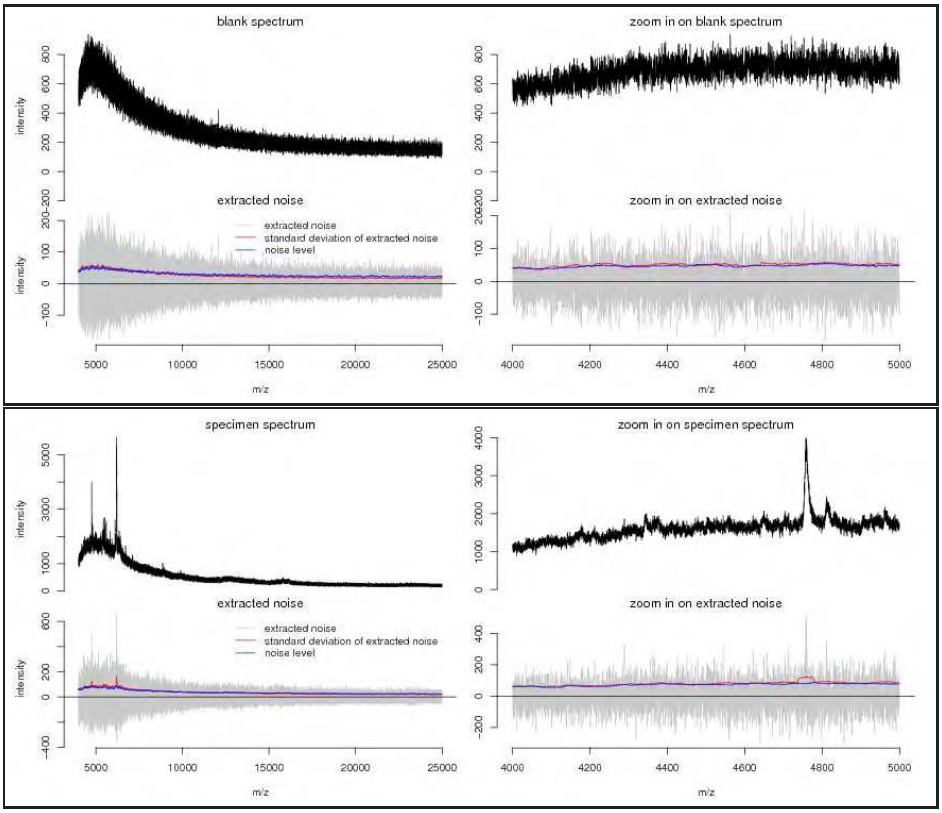


FIGURE 3.10 – Résultat de l'analyse du niveau de bruit sur les deux spectres MALDI-TOF réalisés avec une matrice *sinapinic*, avec agrandissement sur la région 4000Da-5000Da. Le spectre de masse est matérialisé par la ligne noire. La ligne grise et la ligne rouges sont respectivement le bruit restant lorsque le signal lissé est enlevé du signal brute, et le niveau de bruit estimé par déviation standard. La courbe bleue est le niveaux de bruit estimé par notre méthode directement sur le signal original.

mentionné, cela vient de la sensibilité de la mesure de déviation standard aux valeurs extrêmes, alors que la médiane y est moins sensible. Nous avons aussi comparé ces algorithmes sur des spectres de masse SELDI-TOF en provenance d'une autre source (jeu de donnée Stroke [90]). Ceci a nécessité d'ajuster correctement les paramètres de lissage, mais ensuite, les niveaux de bruits calculés par les deux approches était à nouveau comparables (résultats non fournis). Les deux méthodes semblent donc produire des résultats très proches, mais la notre à l'avantage de ne nécessiter aucun lissage ce qui la rend plus facile à paramétrer.

Signalons enfin que nous avons tenté d'appliquer ces approches sur des spectres générés virtuellement, malheureusement les niveaux de bruit obtenus était dans ce cas bien différents. En effet, en suivant le travail Morris et al. [79], nous avons simuler le bruit d'un spectre en générant 100000 valeurs aléatoires, indépendemment les unes des autres, selon une distribution normale (centrée réduite). La déviation standard de ce bruit simulée est donc de 1, mais notre méthode estime le niveau de bruit autour de 1.6. Nous observons donc une différence sur les données virtuelles qui n'existe pas dans les données réelles car nous avons vu que les deux méthodes produisent des niveaux de bruit comparables dans la réalité. Si la simulation du bruit était correcte, nous aurions du avoir encore une fois les mêmes estimations du niveau de bruit, on en déduit donc que la distribution normale n'est pas adaptée à la modélisation du bruit dans les spectres de masse alors qu'elle est couramment utilisée. Le fait que nous obtenons un niveau de bruit de 1.6 en basant notre calcul sur des maxima/minima quand la déviation standard est de 1.0, signifie que la distribution normal génère plus de valeurs extrême que la distribution observée en réalité. Il faut donc certainement remettre en question cette manière de simuler le bruit, en choisissant une distribution de probabilité plus "pointue" pour laquelle notre méthode d'estimation du niveau bruit calculerai une valeur qui se rapproche d'avantage de la déviation standard de la distribution. Peut-être faut il aussi remettre en question l'hypothèse d'indépendance faite pour simuler le niveau de bruit pour chaque valeur d'échantillonnage.

3.4 Normalisation des intensités

L'intensité du signal est une information capitale dans un spectre de masse car elle indique l'abondance des protéines dans les échantillons analysés, or on cherche généralement les protéines dont la quantité change d'un échantillon à l'autre. Malheureusement, l'intensité d'un pic est une information beaucoup moins fiable que la position en m/z des pics. Selon la technologie utilisée, le coefficient de variation de l'intensité des pics peut varier entre 10% et 50% d'un spectre à l'autre [69], et s'agissant des technologies MALDI-TOF et SELDI-TOF ce coefficient se situe autour de 30% [131]. Signalons aussi que la comparaison de l'intensité des pics à l'intérieur d'un même spectre est beaucoup plus fiable que les variations d'intensité d'un spectre à l'autre. C'est d'ailleurs une des raisons qui motive l'utilisation de technologies de quantification de type ICAT (basé sur le ratio d'intensité entre pics), et c'est également une de nos motivations dans la

deuxième partie pour proposer une méthode qui exploite les intensités relatives des pics à la place des intensités absolues.

Les variations d'intensité dans les pics sont liées à plusieurs facteurs difficiles à contrôler dans la pratique comme la quantité totale de molécules déposées dans l'instrument (soumise aux erreurs de manipulation des expérimentateurs) et l'efficacité de l'ionisation. Ces facteurs sont susceptibles de varier d'une expérience à l'autre ce qui rend difficile la comparaison directe des intensités entre les spectres de masse. Le but de la normalisation des intensités est donc de rendre plus fiable ces comparaisons en tentant de corriger l'intensité de chaque spectre de masse par un facteur multiplicatif qui rend les valeurs moins dépendantes des conditions expérimentales.

Idéalement, pour réaliser correctement la normalisation des spectres de masse, il faudrait disposer d'une (ou plusieurs) protéine de masse et de concentration connue à l'intérieur de chaque échantillon et utiliser l'intensité de son pic comme référence pour étirer/rétrécir l'intensité des spectres de masse. Clarke et al. [17] utilise par exemple l'intensité du pic de créatinine, présent à l'intérieur de chacun de ses échantillons d'urine, pour normaliser les spectres de masse. Cependant dans le cas général, l'introduction artificielle de substances dans des échantillons d'expérience SELDI-TOF à un impact sur le résultat qui s'explique par plusieurs facteurs tels que l'effet de suppression des ions ou la compétition pour s'attacher à la surface SELDI [32]. Ce qui fait que dans la majorité des cas, nous ne disposons pas de tel pics de référence pour normaliser les spectres.

En l'absence de référence, l'alternative la plus répandue est de diviser les intensités des spectres de masse par le courant ionique total, c'est à dire par la somme des intensités du spectre. Le courant ionique total est une estimation du nombre d'ion dans les échantillons, cette normalisation fait donc la supposition (forte) que cette quantité est constante dans tous les échantillons. De plus, malgré les apparences, de nombreuses précautions doivent être prises pour calculer correctement le courant ionique total. Il faut effectivement éliminer la ligne de base du spectre masse pour ne garder que les pics du spectre. Or, a ce sujet, Coombes et al. [21] fait remarqué que si celle-ci est située sous le signal, la somme des intensités est largement influencé par le bruit du spectre, et en conséquence il faut aussi lisser le signal. Hilario et al. [48] souligne également que, parfois, une seule protéine très abondante (comme l'albumine dans les échantillons de plasma humain) peut affecter grandement le calcul du courant ionique total et qu'elle devrait être exclu du calcul – à moins que sa concentration ne soit constante –.

Signalons enfin, que nous n'avons mentionné ici que les approches de normalisation relatives au technologies MALDI-TOF et SELDI-TOF à faible résolution. Effectivement, pour d'autres technologies ou les données sont beaucoup plus structurées, d'autres méthodes plus avancées peuvent être envisagées. Evoquons par exemple l'excellent travail de [70] à ce sujet. Celui-ci cherche à quantifier les protéines dans une expérience de type ESI-LC-MSMS pour laquelle on dispose de l'identification des pics. Comme les protéines sont digérées en peptides dans ce type d'expérience, l'information est répétée à plusieurs endroits dans les spectres. De plus chaque peptide peut apparaître dans plusieurs spectres suc-

cessifs de la dimension LC, et sous des états de charges différents ce qui disperse d'autant plus l'information. En se basant sur l'identification des pics, l'auteur propose un moyen d'agréger les informations pour délivrer une quantification optimale.

3.4.1 Méthodes insensibles à la normalisation des intensités

Il semble donc assez difficile de définir et de réaliser une normalisation idéale des intensités en l'absence de pics de références et de données plus structurées. Nous remarquons cependant que la normalisation des intensités n'est nécessaire que si l'on envisage de comparer directement les intensités des pics d'un spectres à l'autre. De nombreuses méthodes d'analyses de données fonctionnent ainsi, mais il en existe d'autres qui fonctionne de manière moins directe : elles commencent par comparer les intensités des pics à l'intérieur d'un même spectre, et vérifie si les résultats des comparaisons sont identiques d'un spectre de masse à l'autre. L'avantage de ces comparaisons indirectes entre les données est que les algorithmes d'apprentissage sont moins sensible à la normalisation des spectres, et certaines méthodes n'ont même pas besoin de normalisation. C'est vers ces méthodes que nous nous tournons dans la la seconde partie de la thèse, cependant pour pouvoir tout de même appliquer les autres méthodes plus courantes, et car cela ne changera en rien nos résultats, les données sont normalisés de manière classique en utilisant le courant ionique total (après suppression de la ligne de base).

Le principe des méthodes qui sont insensibles à la normalisation des spectres est le suivant : comme elles ne peuvent pas comparer directement l'intensité du pic A_{3000} – située dans le spectre A à la position 3000Da – avec l'intensité du pic B_{3000} – située dans le spectre B à la position 3000Da – pour savoir si l'expression de la protéine est plus forte dans l'échantillon A ou dans l'échantillon B, elles fonds des comparaisons indirectes. Par exemple, elles comparent les intensités des pics A_{3000} et A_{4000} ainsi que celles de B_{3000} et B_{4000} qui se trouvent dans les mêmes spectres. Si elles constate que $A_{3000} < A_{4000}$ alors que $B_{3000} > B_{4000}$, elles peuvent tirer la conclusion qu'il y a un changement d'expression de ces protéines entre les deux échantillons. De plus, peu importe les facteurs de normalisation qui sont appliqués aux spectres A et B, à partir du moment qu'ils sont strictement positifs, la conclusion est toujours la même.

On retrouve cette idée de comparaison indirecte entre les intensités des pics dans les travaux de Slotta [108], et Geman [34] qui ont respectivement tenté d'analyser des données de spectrométrie de masse et de puce à ADN. Nous reviendrons plus en détail sur ces méthodes dans la deuxième partie de la thèse où nous proposons une approche qui repose sur une idée similaire, et qui ne nécessite pas de normalisation des spectres.

3.5 Contrôle qualité des spectres de masse

Avant de conclure ce chapitre, mentionnons une étape que nous n'avons pas abordé, et qui pourrait aussi relever du traitement des signaux, c'est le "contrôle qualité" des spectres de masse. Le "contrôle qualité" consiste à déterminer si un spectre est de bonne qualité et qu'il faut le conserver pour les étapes ultérieures, ou si il est de mauvaise qualité et qu'il faut l'éliminer pour éviter de fausser les conclusions de l'analyse. Les raisons de la mauvaise qualité des spectres de masse sont liées à des défaillances dans la préparation des échantillons et/ou dans l'acquisition des spectres de masse. D'après l'étude de [71], une expérience SELDI-TOF génère seulement 63% de spectres de bonne qualité, et 37% de qualité moyenne à faible. Dans cette étude, un spectre de bonne qualité est défini comme un spectre dont le rapport signal/bruit dépasse les 10dB, et on le détermine par une décomposition du signal en ondelettes. D'autres travaux, [50, 18] ont des taux de rejet plus faible (13%), par contre ils évaluent la qualité des spectres de masse à partir de mesures statistiques plus classiques (corrélations, variance, t-test, ...) et ils nécessitent davantage de pré-traitement (élimination de la ligne de base, et détection des pics). Signalons cependant qu'en règle général, un contrôle qualité visuel et une élimination manuel des spectres de mauvaise qualité sont réalisés par les opérateurs des expériences SELDI-TOF.

3.6 Bilan sur le traitement du signal MS

Dans ce chapitre, nous avons vu les étapes de traitement des signaux couramment réalisées sur un spectre de masse pour l'uniformiser. Des algorithmes ont été proposés pour réaliser chacune de ces étapes en se focalisant sur des approches simples et qui répondent le mieux possibles aux spécificités des problèmes. Par exemple pour la ligne de base, on s'est efforcé de définir une méthode qui génère une ligne de base en dessous du signal, avec un seul paramètre intuitif à ajuster. Nous avons également mené plusieurs expériences pour analyser le comportement de nos algorithmes et s'assurer de leur bon fonctionnement, mais davantage d'expériences viendront dans les chapitres qui suivent lorsque le pipeline de pré-traitement sera mieux défini. Nous pourrons alors étudier le comportement du pré-traitement dans son ensemble, et l'influence d'une étape de pré-traitement sur le résultat d'une autre. Par exemple, dans le chapitre 4, nous verrons que l'élimination de la ligne de base a peu d'impact sur le résultat de la détection des pics, et le chapitre 7 est entièrement consacré à l'optimisation des paramètres du pré-traitement et à l'évaluation du pré-traitement dans son ensemble. Dans le chapitre qui suit, nous continuons donc à décrire les étapes de pré-traitement d'un spectre de masse en passant au thème de la détection des pics et en gardant à l'esprit le même soucis de simplicité de la méthode proposée.

Chapitre 4

La détection des pics

Les étapes de traitement des signaux du chapitre précédant permettent de corriger les spectres de masse pour qu'ils soient davantage comparable les uns aux autres. Ces étapes permettent aussi de simplifier les signaux pour faciliter la détection des pics dont il est question dans ce chapitre. Le but de la détection des pics est d'extraire du spectre la liste des positions ou il y a un pic, et également d'extraire leurs caractéristiques (m/z, intensité, largeur, ...). La détection est donc le point central du pré-traitement, puisqu'elle sélectionne l'information à retenir pour l'apprentissage en focalisant sur les pics qui sont censés être les seules données biologiquement pertinentes dans un spectre. Pour donner un ordre de grandeur, la détection de pics résume les dizaines de milliers de valeurs d'échantillonnage du spectre de masse à quelques centaines de pics, c'est à dire qu'elle réduit la quantité d'information d'un facteur 100 à 1000. Pour éviter une perte de l'information pertinente dans de telles conditions, nous avons consacré un soin particulier à cette étape. L'algorithme de détection de pics qui en ressort est a mon avis une contribution importante de cette thèse pour le pré-traitement des spectres de masse.

4.1 État de l'art

En ce qui concerne le problème de la détection des pics, deux situations peuvent se présenter selon que la résolution des spectres de masse laissent apparaître ou non la distribution isotopique des pics. Le cas le plus simple est celui ou les pics montrent une distribution isotopique car on peut alors caractériser les pics par leur forme et ainsi mieux les distinguer du bruit. Pour cela, on décrit généralement la distribution isotopique par un modèle paramétrique, comme par exemple une mixture de gaussiennes [39, 82, 12, 63], et on essai de reconnaître cette forme dans le spectre de masse. Cela facilite grandement la recherche car il suffit de reconnaître dans les spectres les emplacements pour lesquels le modèle s'ajuste convenablement aux données, et il est même possible de détecter des pics qui se chevauchent. Le lecteur intéressé peut se reporter à [48] pour une

revue avec plus de détails et de techniques, sur la détection des pics. Il peut aussi consulter [67, 100, 29, 80] pour des approches de détection de pics à base d'ondellettes que l'on peut considérer, dans une certaine mesure, comme l'ajustement d'un modèle de pics sur des signaux. Enfin, concernant la détection de pics dans les spectres de masse de haute résolution, on peut se référer à la littérature relative aux logiciels pour l'identification des péptides/protéines dans les expériences de type LCMS [69, 114, 7, 62, 36, 8, 120, 44, 78, 81, 109].

Il est en revanche plus difficile de modéliser les pics des spectres SELDI-TOF qui ne laisse pas apparaître la distribution isotopique des pics : la forme des pics dans ces spectres est assez similaire aux motifs que l'on retrouve dans le bruit et leur largeur est difficile à estimer. [21] suggère que l'on peut modéliser la forme des pics dans un spectre SELDI-TOF en simulant le processus physique du spectromètre, mais aucune méthode n'emploie ce principe. Du coup, comme un modèle de pic est difficile à définir et que leur forme est difficile a caractériser, il existe peu de méthodes efficaces pour détecter les pics dans un spectre SELDI-TOF. Les méthodes existantes reposent essentiellement sur des techniques non paramétriques, et en particulier de nombreux algorithmes que l'on retrouve dans la littérature dérivent de celui de Yasui et al. [128]. La détection de pics que propose cette auteur consiste en un simple lissage de signal suivi d'une recherche des maxima locaux situés au dessus du niveau de bruit. [116, 79, 94, 19, 21] reprennent notamment cette idée.

Le principal défaut de la détection des pics par maxima locaux est qu'elle est contrôlée par un paramètre qui définit le nombre de voisins à considérer pour déterminer si on est en présence d'un maximum local ou non, or celui-ci est très difficile à ajuster correctement car il dépend des conditions d'acquisition du spectre et de la largeur des pics qui varie à l'intérieur même d'un spectres. Un autre de ses défauts, est qu'elle détecte de nombreux "faux pics" qui sont en réalités du bruit de fond du signal [94]. Aussi, les erreurs commises lors de l'estimation de la ligne de base du spectre ont un impact important sur le résultat de la détection de pics : une sous-estimation de la ligne de base surélève le signal et augmente les rapports signaux/bruits ce qui conduit des maxima locaux situées sous le niveau de bruit à être considéré comme des pics. A l'inverse, une surestimation de la ligne de base diminue les rapports signaux/bruits, et risque de changer le statut de certains maximums locaux en bruit. Comme le suggère plusieurs auteurs [79, 94], le lissage du signal avant la recherche des maximums locaux peut améliorer les résultats, mais cette transformation qui altère le signal risque aussi de détruire les informations qu'il contient.

L'amélioration la plus intéressante de l'algorithme de recherche de maxima locaux proposé par Yasui et al.[128] est certainement celle que propose Pratapa et al. [94]. Cet auteur introduit une file d'attente à priorité dans laquelle ils rangent la totalité des maximums locaux trouvées dans le signal, pour ensuite les extraire un à un selon un ordre de confiance liée à leur rapport signal/bruit. A chaque pic qui sort de la file, ils recherchent sa zone d'influence dans le signal en identifiant les vallées du pics situées sur sa gauche et sur sa droite (ceci permet au passage de calculer l'aire du pic). Les éléments de la file qui tombent dans les régions déjà assignées à un pic sont simplement ignorés. L'avantage

de ce processus est qu'il supprime des "faux pics" détectés par l'algorithme de [128]. Cela vient d'abord du fait qu'il traite les maxima du signal selon un ordre de confiance et non sequentiellement de la gauche vers droite, ce qui permet de stopper le processus après avoir obtenu le nombre de pics désirés, et non pas en ayant fixé un seuil signal/bruit a priori. Ensuite, d'autre "faux pics" détectés par l'algorithme de [128] sont également évités par cet amélioration car elle interdit que deux pics soit plus proche que ce que la résolution de l'instrument ne le permet. Par contre l'utilisateur est toujours ennuyé par le paramètre qui contrôle le voisinage, et la méthode utilisée pour rechercher les vallées des pics est discutable (elle est assez compliqué car elle est basée sur un calcul de gradient difficile à réaliser sans un lissage du signal).

Le logiciel propriétaire *Ciphergen ProteinChip* adopte quand à lui une approche différente pour la détection des pics dans les spectres SELDI-TOF, en introduisant une notion de vallée. Il parcourt le signal pour y identifier les endroits ou le spectre monte suffisamment, puis redescend dans sa vallée au même niveau que le point initial [32]. Seul les endroits qui ont des vallées assez profondes, et qui ont une aire minimum sont considérés comme des pics. Le gros avantage de cette approche par rapport à la recherche des maximums locaux c'est qu'elle ne fait aucune supposition sur la largeur des pics, mais simplement sur leur intensité, et du coup elle n'a pas de paramètre qui contrôle la taille du voisinage à considérer. De plus la ligne de base à un impact limité sur le résultat de la détection car les pics sont repérés à l'aide de différences entre des informations très locales où la ligne de base peut être supposée constante. Un autre avantage de l'algorithme, c'est que l'identification des vallées des pics, permet au logiciel de calculer leurs aires en intégrant les intensités du signal entre les deux vallées. Par contre il semble encore une fois qu'un lissage du signal soit nécessaire pour améliorer les résultats de cet algorithme. De plus le paramètre qui contrôle la profondeur minimale des vallées est difficile à paramétré a priori, et il faut ré-exécuter entièrement l'algorithme si on souhaite changer sa valeur.

Wallace et al. [121] proposent encore une autre solution, non paramétrique, au problème de la détection des pics et des vallées dans les spectres de masse. Pour cela, ils réalisent une approximation du signal par une fonction linéaire par morceau à l'opposé de ce que nous proposons pour le calcul de la ligne de base. En effet, plutôt que de partir du signal original et de supprimer un à un les points pour simplifier le signal, il débute par un segment qui joint les deux extrémités du signal et le brise petit à petit en ajoutant les points les plus éloignés de cette approximation. Le processus est itéré tant que la distance entre le point le plus éloigné et l'approximation dépasse un certain seuil. Les points de cassures définissent les pics et leurs vallées. L'avantage cette fois est que l'algorithme ne requière aucun lissage du signal, et ne fait aucune hypothèse sur la forme des pics. Par contre il nécessite des calcules inutilement coûteux.

En ce qui concerne les méthodes paramétriques applicables à la détection des pics dans les spectres de faible résolution, citons [67, 77], et rappelons que leur avantage est de pouvoir distinguer des pics qui se superposent. La première, [67], utilise une décomposition en ondellettes du signal pour trouver les pics candidats, couplée à l'ajustement d'un modèle pour la forme des pics. Quand

à [77] il s'agit d'une approche Baysienne du problème qui requière aussi un modèle pour la forme des pics. Or, nous avons vu qu'un telle modèle est difficile à définir, et l'approcher par une simple gaussienne perturberait ces algorithmes qui pourraient voir plusieurs pics superposés en un seul. Nous ne nous attardons pas davantage sur ces approches et passons plutôt maintenant, à la description de l'algorithme que nous proposons pour la détection des pics que l'on peut voir comme une fusion entre la file d'attente de [94] et la recherche des vallées du logiciel *Ciphergen ProteinChip* qui nous permet de cumuler les avantages des deux approches.

4.2 Méthode pour la détection de pics dans un spectre de masse

L'algorithme final que nous proposons pour la détection des pics a évolué en trois temps. Chacune des évolution est une généralisation des concepts introduits dans la version précédente de l'algorithme. D'ailleurs les résultats de la détection de pics obtenus avec la première version de l'algorithme peuvent également être produit à l'aide de la deuxième et de la troisième version. Nous pourrions présenter seulement la version final la plus aboutie, cependant nous préférons présenter les trois versions de l'algorithme de détection de pics, car chacune à l'avantage de présenter le même problème sous un angle de vu différent, ce qui fourni un aperçu de l'universalité de la méthode.

4.2.1 Version initiale : Détection de pic par recherche de maxima

La version initiale de notre algorithme de détection de pics est décrite dans Prados et al. [90]. Elle est assez similaire à l'algorithme de la file d'attente utilisée par Pratapa et al. [94] mais plus simple car nous reconsidérons l'usage des maxima locaux, ce qui nous permet d'éviter le paramètre encombrant au sujet du nombre de voisins à considérer pour le maximum local. Signalons avant de commencer que l'algorithme cherche seulement à trouver les positions des pics dans le spectre, et l'estimation de l'intensité des pics et de leurs aires viendra ultérieurement.

Notre algorithme fonctionne de la manière suivante : tous les points du spectres sont d'abord rangée dans une file d'attente, et à chaque itération de l'algorithme on en extrait le point le plus intense du spectre masse pour le considérer comme un pic candidat [1]. Pour ce point, p, le plus intense, on recherche les premiers points l et r situés respectivement à sa gauche et à sa droite tel la différences d'intensité par rapport à p soit supérieures à une valeur seuil vd. La région du spectre qui va de l à r est ensuite associé au point p et

1. Notez, que nous recherchons l'intensité la plus élevée et non pas le rapport signal/bruit le plus élevé. Le bruit est pour l'instant supposé constant, mais en réalité il en sera tenu compte une fois que le signal associé au pic sera connu. De plus, cela nous garanti que les pics que nous détectons sont des maxima locaux du signal.

si cette région ne chevauche pas une région déjà parcouru, alors p est considéré comme un nouveau pic. Pour optimiser la recherche, les points de la file d'attente qui tombent dans des régions du signal déjà affectées sont ignorés. vd contrôle donc la hauteur minimum que doivent avoir les pics, et par la même leur étendu en m/z. Ce paramètre peut également être vu comme contrôlant la sensibilité de la détection de pic : plus la valeur de vd est grande et moins il y a de pics détectés dans le spectre, mais plus la confiance dans les pics détectés est grande . L'algorithme 3 fourni le pseudo code du processus que nous venons de décrire. Il peut être implémenté avec une complexité $\mathcal{O}(n)\log(n)$ (n étant le nombre de points dans le spectre).

Algorithm 3 peakdetectV1(f[],vd) : Algorithme de détection de pics (version 1)

entrée $f[]$: Le spectre (=le tableau contenant la séquence des intensités)
entrée vd : Le seuil minimum d'intensité pour un pic
sortie $\psi \leftarrow \emptyset$: Liste des pics détectés
initialisation : marquer tout les points de f[] comme non traités
while (il y a un point de $f[]$ non traité) **do**
  $p \leftarrow$ index du point non traité le plus intense de $f[]$
  $l \leftarrow$ premier point à gauche de p tq l est traité ou $(f[p] - f[l] \geq vd)$
  $r \leftarrow$ premier point à droite de p tq r est traité ou $(f[p] - f[r] \geq vd)$
  Marquer les points de f[] situés entre l et r comme traité
  if $(f[p] - f[l] \geq vd)$ et $(f[p] - f[r] \geq vd)$ **then**
    $\psi \leftarrow \psi \cup \{p\}$ {p est un pic}
  end if
end while
return ψ

L'algorithme fonctionne assez bien, et il cumule les avantages de ceux vus précédemment : il est rapide, il ne fait aucune supposition sur la largeur des pics et sur leur forme, il peut être appliqué sans lissage du signal (mais nous verrons dans le chapitre 7 que cela améliore sa performance), et enfin, il est très peu sensible à l'élimination de la ligne de base. Effectivement, concernant ce dernier point, remarquez que l'algorithme base sa décision (pic/non pic) seulement sur des différences d'intensité entre des informations locales sur lesquelles on peut supposer que la ligne de base est constante. La différence d'intensité n'est donc pas (ou peu) influencée par la ligne de base ce qui rend possible l'utilisation de l'algorithme directement sur les signaux brutes sans avoir à éliminer la ligne de base. Cette affirmation est vérifiée plus bas par des expériences.

Remarquez aussi que l'algorithme n'utilise pas la valeur m/z des points du spectre mais il a seulement besoin de connaître leur séquence (en m/z) et leur intensité. Ainsi, cet algorithme est capable de détecter des pics de largeur et de forme quelconque, même si elle change à l'intérieur même du spectre.

4.2.2 Amélioration de la détection de pics par calcul des profondeurs de vallées

La version initiale de la détection de pics a cependant un inconvénient, c'est qu'il faut choisir à priori la valeur seuil vd. De plus, elle se contente d'extraire les pics mais ne leur associe aucune mesure de confiance. Ainsi, pour chaque point du spectre, elle indique de manière booléenne si nous sommes en présence d'un pic ou pas, alors qu'il pourrait être plus avantageux d'avoir une mesure de confiance (continue) qui indique par exemple la probabilité qu'un point du spectre soit un pic. Le paramètre vd fournit pourtant, dans une certaine mesure, un moyen de contrôler la confiance minimum dans les pics en demandant qu'ils aient une intensité minimum. Mais ce paramètre est difficile à régler à priori, et le résultat rendu ne permet pas d'ordonner les pics détectés selon leur vraisemblance. Enfin, si l'on modifie le paramètre vd, il faut ré-exécuter totalement l'algorithme pour obtenir un nouveau résultat. Pourtant, on peut remarquer que l'ensemble des pics détectés pour une valeur seuil vd_1 selon l'algorithme initial (algorithme 3) est inclut dans l'ensemble des pics détectés pour une valeur seuil $vd_2 \leq vd_1$ ce qui définit une relation d'ordre sur les pics. Il est donc par exemple possible de déterminer les 10 pics les plus vraisemblables d'un spectre en recherchant (par dichotomie) la valeur vd pour laquelle exactement 10 pics sont détectés. L'ennui c'est que c'est coûteux en temps de calcul puisque à chaque changement de la valeur seuil, l'algorithme doit être ré-exécuté dans sa totalité.

L'amélioration que nous proposons ici (présentée dans notre article Zerefos et al.[131]) est capable de calculer cette relation d'ordre avec une complexité faible et sans paramètre. Ce nouvel algorithme est effectivement en mesure d'évaluer pour chaque point p du spectre la valeur seuil vd_p jusqu'à laquelle ce point est considéré comme un pic par la version initiale. De manière intéressante, cette valeur correspond également à la hauteur maximum que peu avoir un pic situé en ce point, et l'ordre des valeurs seuils obtenu est équivalent à la relation d'ordre sur les pics vu plus haut. Il est alors facile de retrouver les pics détectés par une exécution de la version initiale pour un seuil vd spécifique, car ceux sont les points p du spectre qui ont les valeurs vd_p supérieures au vd désiré. De plus, nous allons voir que le résultat se visualise simplement et se révèle très instructif pour l'utilisateur. Effectivement, comme le seuil vd_p est un indicateur de notre confiance dans la présence du pic au point p, il suffit de tracer les valeurs seuils obtenues en chaque point du spectre pour visualiser la position des pics (voir exemple sur la figure 4.2). Le grand avantage de cette version par rapport à son ancêtre, réside dans le fait qu'il n'y a pas de paramètre. Il s'agit simplement d'une transformation du spectre de masse en un signal plus "pure" qui indique en chaque point du spectre la valeur seuil à partir de laquelle ce point devient un pic, et le choix du seuil de détection approprié est fait une fois que cette information est connue de l'utilisateur. De plus, comme la valeur seuil correspond à la hauteur maximum que le pic peut avoir par rapport à l'algorithme initial (algorithme 3), elle peut être interprété comme le signal associé à chaque point du spectre. Enfin, si maintenant nous sommes

intéressé par l'extraction des 10 pics les plus probables dans un spectre, il suffit d'extraire les 10 points qui ont les valeurs seuils les plus élevées alors qu'une recherche dichotomique était nécessaire avec la version initiale.

L'algorithme de recherche de vallées

Pour calculer le signal en un point p du spectre (c'est à dire la valeur seuil jusqu'à laquelle un pic est détecté par l'algorithme 3), il suffit de rechercher sur la gauche de p (resp. sur la droite de p) le point d'intensité minimum v_l (resp. v_r) tel que la portion de spectre entre v_l (resp. v_r) et p ne contient aucun point d'intensité supérieure à p. Nous appelons ces points les vallées gauches et droites de p que nous illustrons sur la figure 4.1. Noter entre particulier sur la figure 4.1b la position du point v_l qui n'est pas entre les deux pics mais bien avant le premier pic dont l'intensité est inférieure à celle du point p. En cela, notre définition de vallée diffère des définitions vu plus haut pour lesquelles la vallée doit être située entre deux pics. Noter aussi sur la figure 4.1c la position du point v_r qui coïncide avec celle de p, car le point à la position $p + 1$ a déjà une intensité supérieure à celle du point p. La profondeur de la vallée droite en ce point, donnée par la différence d'intensité entre les points p et v_r, est donc de 0. Étant données ces définitions de vallées, le seuil à partir duquel l'algorithme initial (algorithme 3) considère un point p du spectre comme un pic est donnée simplement par le minimum entre la profondeur de sa vallée gauche et la profondeur de sa vallée droite. Cette valeur représente pour nous le signal au point p.

Afin de rechercher les vallées v_l et v_r en chaque point du spectre, nous utilisons l'algorithme 4. Celui-ci parcours la séquence des points dans le spectre de la gauche vers la droite en stockant dans une pile les points pour lesquelles la vallée droite (v_r) n'est pas encore déterminable. Il est très simple, et extrêmement rapide puisque il a une complexité linéaire $\mathcal{O}(n)$ (n étant le nombre de points dans le spectre), c'est à dire inférieure à la complexité de la version initiale pour un résultat plus général et plus pertinent. Quand au calcul du signal en un point p, il est obtenu par l'algorithme 5 qui renvoie simplement le minimum de la profondeur entre les deux vallées trouvées par l'algorithme 4.

La figure 4.2 montre un exemple de visualisation du résultat rendu par cet algorithme. La ligne noire représente le spectre de masse en entré de l'algorithme, l'histogramme bleu donne le signal obtenus en sortie. Le résultat que dévoile cette figure est très satisfaisant car l'histogramme bleu (le signal calculé) est bien plus élevé aux endroits où les pics sont les plus marqués. Par contre, nous observons aussi que les pics situées dans le bruit ont un signal non nul (même si leur intensité est inférieure à celle des pics réelles). La difficulté est maintenant de fixer le seuil de confiance vd en dessous duquel nous considérons que nous sommes en présence de bruit. En cela, la visualisation de la figure 4.2 fournie une aide précieuse à celui qui souhaite réaliser le seuillage manuellement. Par contre, comme le bruit n'est pas forcement constant dans un spectre de masse, il est préférable de considérer le rapport signal/bruit des pics pour décider si l'on est en présence d'un pic ou pas, en utilisant une des méthode vu dans la section 3.3 pour l'estimation du niveau de bruit.

Algorithm 4 findValleys(f[]) : Algorithme de recherche des vallées

```
1: entrée f[] : Le spectre (= le tableau contenant la séquence des intensités)
2: sortie (left[], right[]) : position des vallées gauche et droite
3: stack = new Stack() /*une structure de pile*/
4: iMin = 1 /*position de la plus petite valeur connue valeur entre i et le
   sommet de pile stack.top()*/
5: for i in 1..length(f[]) do
6:    if (f[iMin] > f[i]) then
7:       iMin = i
8:    end if
9:    while ( !stack.isEmpty() & (f[i] ≥ stack.top())) do
10:      right[stack.top()] = iMin
11:      if (f[left[stack.top()]] < f[iMin]) then
12:         iMin = left[stack.top()]
13:      end if
14:      stack.pop()
15:   end while
16:   left[i] = iMin
17:   stack.push(i)
18:   iMin = i
19: end for
20: while ( !stack.isEmpty()) do
21:   right[stack.top()] = iMin
22:   if (f[left[stack.top()]] < f[iMin]) then
23:      iMin = left[stack.top()]
24:   end if
25:   stack.pop()
26: end while
```

Algorithm 5 vdThresholds(f[]) : Algorithme de calcul du signal

```
1: entrée f[] : Le spectre (= le tableau contenant la séquence des intensités)
2: sortie signal[] : le signal = le seuil de détection à partir duquel le point est
   considéré comme un pic par l'algorithme 5
3: (left[],right[]) = findValleys(f[])
4: for i in 1..length(f[]) do
5:    signal[i] = min(f[i] - f[left[i]], f[i] - f[right[i]])
6: end for
```

Ceci peut être pris en compte soit en matérialisant le niveau de bruit par une ligne (comme la ligne verte de la figure 4.2) et en considérant comme pic les points où l'histogramme est au dessus de cette ligne ; soit en divisant chaque point de l'histogramme bleu par une estimation du niveau de bruit en ce point, ce qui donne une estimation du rapport signal/bruit de chaque point du spectre. Un exemple de ce dernier cas peut être trouvé sur la figure 4.3b, qui montre le rapport signal/bruit obtenu sur un spectre de masse "blanc" censée ne contenir aucun pic. Nous revenons sur l'utilité d'un tel exemple dans la sous-section qui suit.

Ajustement du seuil de détection

Concernant l'ajustement du seuil de détection de pics dans le contexte de la recherche de bio-marqueurs, nous recommandons d'adopter une politique laxiste, c'est à dire d'utiliser une sensibilité élevée de détection au risque de laisser passer des pics qui sont en réalité du bruit. La raison pour cela est que dans la suite du pré-traitement, lors de l'alignement (voir section 5.2), nous conservons seulement les pics que l'on retrouve dans plusieurs spectres de masse justement pour éliminer d'éventuelles "faux pics", car si un pics est en réalité du bruit, il y a peu de chance pour qu'on le retrouve dans un autre spectre de masse. Toujours a propos de l'ajustement du seuil de la détection de pics, nous recommandons aussi dans Prados et al. [90] de s'aider d'un spectre blanc pour régler le seuil de détection. Un tel spectre est censée ne contenir aucun pic, il nous faut donc régler la sensibilité de la détection de pics de manière à ce que aucun pic (ou plutôt très peu pour prendre en compte d'éventuelles contaminants) ne soit détecté dans ce spectre. La figure 4.3 montre un exemple de détection sur un spectre blanc. L'histogramme du bas indique les rapports signaux/bruits calculée en chaque points du spectre, et suggère que le rapport signal/bruit des pics est d'au moins 2.0. Effectivement, en fixant le seuil à cette valeur seulement 12 pics sont détectés, avec un m/z $>$ 800Da. Cette valeur constitue donc une borne inférieur du seuil de détection vd pour ce type de spectre en dessous de laquelle il faut éviter descendre pour ne pas détecter des pics qui sont en réalité du bruit.

Insensibilité à la suppression de la ligne de base

Afin de montrer la faible sensibilité de l'algorithme de détection de pics à la suppression de la ligne de base, nous avons mené une expérience qui compare les signaux calculées par l'algorithme 5 lorsqu'on l'applique sur un spectre de masse avant et après avoir supprimer sa ligne de base. La figure 4.4a montre le résultat de cette comparaison. La ligne noire représente le spectre de masse utilisé, la ligne rouge sa ligne de base calculée avec l'algorithme 2, et l'histogramme bleu la différence entre les signaux calculés sur le spectre (en noir) et ceux calculés sur le spectre sans sa ligne de base (noir - rouge). Si l'algorithme est complètement insensible à la suppression de la ligne de base, l'histogramme bleu doit afficher la valeur constante 0. Or, comme plusieurs valeurs sont significativement non nulles

sur la figure, on pourrait penser que l'algorithme est sensible à la suppression de la ligne de base. Mais, en y regardant de plus prêt, on s'aperçoit que les valeurs non nulles sont produites par des artefacts. Le grossissement de la figure 4.4b montre plus en détail ce qui se passe au endroits qui posent problèmes. Ce que l'on y observe c'est que les valeurs qui contribue positivement à l'histogramme sont légèrement décalées par rapport aux valeurs qui contribuent négativement à l'histogramme. Ce phénomène se produit lorsque plusieurs valeurs consécutives dans les intensités du spectre de masse sont proches du maximum local, ce qui peut provoquer un léger décalage de la position du maximum local lorsqu'on élimine la ligne de base. Si l'on ignore ces petits décalages, nous observons une symétrie presque parfaite de l'histogramme bleu par rapport à l'axe des x, ce qui tend à montrer que l'algorithme de détection de pics est (quasi) insensible à l'élimination de la ligne de base. Il y a cependant une exception, c'est le pic d'intensité maximum dans le spectre original : le signal calculé pour ce pic est très affecté par l'élimination de la ligne de base. Cette observation est pourtant logique, car, sur la figure 4.4, le pic d'intensité maximum est situé à un endroit où la ligne de base est élevée, et ses vallées sont les deux points d'intensités minimum situés sur sa gauche et sur sa droite. Comme ces trois points sont éloignés, il peut y avoir des différences de correction lors de l'élimination de la ligne de base.

Une quantification plus précise de la stabilité du signal calculé face à l'élimination de la ligne de base s'impose. Afin d'étudier les variations de signal lorsque la ligne de base est élimine, comparons l'ensemble des pics qui ont un rapport signal/bruit supérieur à 3.0 avant et après élimination de la ligne de base, et tachons de résoudre les problèmes d'alignement. On trouve 232 pics qui passent le seuil avant élimination de la ligne de base, et 237 après. L'alignement des deux listes de pics (en utilisant l'algorithme MZCL du chapitre 5) montre qu'il y a 227 pics en commun, c'est à dire que 5 des 232 pics qui avaient été identifiés avant élimination de la ligne de base ne sont pas retrouvées après, et 10 pics sont apparus. Sur les 227 pics en commun, les rapports signaux/bruit entre les pics trouvés avant et après élimination de la ligne de base augmente en moyen que de 0.006, avec une déviation standard de 0.44. Noter que ces valeurs sont fortement influencées par le pic d'intensité maximum (discuté ci-dessus) dont le rapport signal/bruit chute lui de 6.0 points lorsque la ligne de base est éliminée. D'ailleurs, si on ignore ce pic, la moyenne passe à 0.03, et la déviation standard à 0.18. Ces petites valeurs indiquent que le signal calculés par l'algorithme 5 varient très peu lorsqu'on élimine la ligne de base[2]. Si on s'intéresse maintenant aux 5 pics qui ont disparus, et aux 10 pics qui sont apparus, on voit qu'ils ont tous un rapport signal/bruit proche du seuil 3.0 que l'on s'est fixé (en moyenne leur rapport signal/bruit est de 3.03). On peut donc expliquer leur apparition et leur disparition par les petites incertitudes observées sur les rapports signaux/bruit qui font qu'il franchisse le seuil de détection ou pas.

Pour améliorer la précision de ces valeurs, les calcules ont été répété sur 50

2. Mentionnons que les résultats ne sont pas biaisés par le calcul du niveau du bruit qui est calculé sur le signal original dans une première phase, et reste donc le même que l'on élimine la ligne de base ou pas.

spectres de masses acquis dans des conditions identiques. Le jeu de données utilisé, ACCA-UF5kD, est décrit dans [131] et dans la section 6.1.1). Les résultats indiquent que sur l'ensemble des données, il y a en moyenne 284 pics qui ont un rapport signal/bruit supérieur à 3.0 avant élimination de la ligne de base, et 286 après. Les deux listes de pics résultants de chaque spectre partagent en moyenne 278 pics. Le rapport signal/bruit du pic le plus intense chute en moyenne de 11.3 points, mais la variation du rapport signal/bruit n'est en moyenne que de 0.04 avec une déviation standard de 0.28. Enfin, les pics qui apparaissent et disparaissent ont en moyenne un rapport signal/bruit de 3.11. Ces résultats sont légèrement au dessus de ceux observés précédemment sur un seul des spectres, mais ils restent acceptable et permettent de tirer les mêmes conclusions qu'au dessus. Ils sont donc doublement intéressant car 1) ils tendent à montrer que l'algorithme de détection de pics garantie une certaine stabilité du résultat, peu importe le pré-traitement qui à été appliqué en amont (suppression de la ligne de base ou pas) ; et car 2) ils nous invitent à ne pas supprimer la ligne de base pour faire la détection de pics ce qui évite cette étape où des erreurs peuvent être introduites.

Similitudes avec les algorithmes "watershed"

Il est intéressant de faire le rapprochement entre l'algorithme de détection de pics dans sa version initiale, et les méthodes de segmentation watershed que l'on rencontre en traitement d'image. Dans une segmentation watershed, le signal est considéré comme un relevé topographique que l'on immerge progressivement d'eau. Lors de l'immersion, des bassins se forment dans les minima du signal et grossissent progressivement jusqu'à se rencontrer en des points qui définissent la segmentation de l'image. La difficulté dans ce type d'algorithme est de déterminer la liste des minima initiaux desquelles il faut partir en évitant le bruit. En transformant les spectres de masse SELDI-TOF par symétrie horizontale, notre procédure de détection de pics s'apparente à du watershed sur 1-dimension. La version améliorée de la détection de pics, écarte la difficulté de choisir la liste des minima initiaux en décidant de les prendre tous en considération. Cette similitude avec les algorithmes de traitement d'images nous pousse à considérer une extension de la version amélioré pour traiter des données de dimension supérieurs comme les images. C'est précisément ce en quoi consiste l'extension suivante de l'algorithme où nous généralisons le concept de profondeur des vallées que nous avons définie pour une séquence d'intensité, à la notion de vallée dans un graphe de voisinage. Comme nous allons le voir, cela nous permet de traité des données bi-dimensionnelles et même de dimension supérieure.

4.2.3 Extension pour la détection bidimensionnelle des pics

Comme nous venons de le mentionner, l'amélioration présentée ici à pour but de généraliser le concept de vallée introduit dans la version précédente à

des signaux de plus grande dimension. En particulier une généralisation de l'algorithme à la 2D nous intéresse pour la détection des pics dans des données LCMS que l'on peut voir comme une image en niveau de gris, voir exemple figure 4.5a. D'autres applications possibles pourrait être la détection des spots dans les images de puces à ADN (2D), la détection de tumeurs dans les images médicales (3D ou 4D), ou, plus ambitieux, la détection d'objets dans des vidéos (3D). Mais nous nous concentrerons seulement sur la spectrométrie de masse, en étendant légèrement notre propos aux données LCMS qui offrent des données bidimensionnels contrairement au donnée unidimensionnel du SELDI-MS.

Pour réaliser cette extension, il nous faut remarquer que la version améliorée de l'algorithme de calcul du signal recherche simplement dans le voisinage de chaque point du spectre un autre point dont l'intensité est supérieure ou égale à la sienne. Dans un spectre de masse, chaque point à deux voisins, un à gauche et un à droite, mais dans une image, un pixel en a généralement quatre (haut, bas, gauche et droite) ou neuf si on considère le voisinage en diagonal, et dans une image 3D c'est plutôt six voisins (ou 26 avec les voisins en diagonal). Pour généraliser cette notion de voisinage nous considérons l'emploi d'un "graphe de voisinage", dont les noeuds sont les éléments qui nous intéressent (par exemple les points des spectres de masse, ou les pixels d'une image) et les arcs lient deux éléments qui sont dans le voisinage l'un de l'autre. Par exemple, pour une image 2D, le graphe de voisinage est une grille dont les noeuds sont les pixels. Mais, dans certains cas, le graphe de voisinage peut être plus difficile à construire. Par exemple, les données brutes d'une expérience LCMS consistent en une suite de spectres de masses réalisés les uns à la suite des autres. Chaque spectre est généralement fournies sous forme d'une liste ordonnées de points qui indiquent son m/z et son intensité, en omettant les points d'intensités nulles pour compresser l'information. D'un spectre à l'autre les valeurs m/z des points peuvent être légèrement décalées du fait d'un re-calibrage de l'instrument. Comme les données ne sont pas alignées comme les pixels d'une image, il nous faut reconstruire le graphe de voisinage à partir des coordonnées des points (voir exemple figure 4.5b). Afin d'identifiées le voisinage de chaque points, on peut envisager d'utiliser une triangulation de Delaunay [41], mais l'inconvénient de cette approche est que le graphe construit dépend des unités que l'on choisi pour représenter les deux dimensions de l'expérience LCMS (le temps de rétention et le rapport masse/charge). Nous avons donc préférer rechercher pour chaque point les quatre voisins les plus proches situées en dessus, en dessous, à gauche et à droite de lui, tout en restant dans un périmètre borné.

Nous avons généralisé l'algorithme de recherche des vallées aux graphes de voisinage, ce qui permet en particulier de traiter des données de dimension n, pour peu que l'utilisateur soit en mesure de fournir un graphe de voisinage. A partir du graphe de voisinage auquel une information d'intensité est associée à chaque noeud, le nouvel algorithme est en mesure d'identifier les noeuds vallées de chaque noeud du graphe. Pour cela, il nous faut étendre la notion de vallée que nous avons vu pour les spectres de masse aux graphes de voisinage. Rappelons que dans le cas unidimensionnel des spectres de masse, une vallée est définie comme le point minimum entre deux points de même intensité, que chaque point

n'a que deux vallées (une à gauche et une à droite), et que seul la moins profonde des deux vallées nous intéresse. Rappelons aussi qu'une intensité est associée à chaque noeud du graphe de voisinage, dans ce contexte la généralisation de la définition d'une vallée est naturel : la vallée d'un noeud p du graphe, est le noeud v d'intensité minimum par lequel il faut passer pour atteindre un autre point d'intensité supérieure ou égale à celle de p en prenant le chemin qui "descends le moins bas possible" (en terme d'intensité)[3]. En d'autre termes, si M est l'ensemble des noeuds d'intensité minimum de tous les chemins qui lient p à un autre noeud d'intensité supérieure à celle de p, alors la vallée v est le point d'intensité maximum de l'ensemble M.

La recherche des vallées dans un graphe de voisinage se réalise de manière assez similaire à celle de la recherche des vallées gauche et droite dans un spectre unidimensionnel : l'algorithme 6 qui réalise cela parcours le graphe de voisinage en commençant par le noeud d'intensité maximum et s'étant petit à petit au noeuds alentours en stockant sur une pile les noeuds pour lesquelles on ne peut pas encore déterminer la vallée. Par contre comme il ne s'agit plus d'une séquence, mais d'un graphe, à chaque itération il y a plusieurs choix pour le prochain voisin à parcourir, et il faut évidemment choisir celui qui possède l'intensité la plus élevé en premier pour garantir que l'on trouve le plus grand minimum en premier. C'est pourquoi les voisins candidats à la prochaine itération sont stockés dans une file d'attente avec priorité de laquelle sont extrait les points d'intensité maximale. L'algorithme 6 a une complexité de $\mathcal{O}(n^2)$ (ou n est le nombre de noeuds dans le graphe) si le graphe de voisinage est complètement connecté (ce qui n'aurait aucun sens !), par contre comme la connectivité du graphe de voisinage est généralement très faible et même souvent constante (par exemple le degré ne dépasse pas 4 dans le cas 2D des LCMS, ou des images), la complexité de l'algorithme est au pire de $\mathcal{O}(n \log(n))$, c'est à dire très efficace.

Pour tester l'algorithme nous avons effectué une détection de pics sur les données bi-dimensionnelles LCMS de la figure 4.5. Sur cette figure, les points bleus du graphe de voisinage indiquent la position des pics qui ont été détectés par le nouvel algorithme, c'est à dire les noeuds du graphe pour lesquelles la différence d'intensité entre eux et leurs vallées est supérieure à une valeur seuil. Par rapport à une détection classique, spectre par spectre, qui aurait trouvé un pic sur chaque spectre, la détection bidimensionnels à l'avantage de regrouper automatiquement les pics dans la dimension du temps de rétention. L'analyse des pics ainsi détectés pose des questions intéressantes. Par exemple : pourquoi ne détecte-t-on pas tous les pics isotopiques au même temps de rétention ? Est-ce à cause du recouvrement de plusieurs pics ? Est-ce du bruit ? De la même manière, trouve-t-on les pics mono-chargés et multi-chargés d'un même peptide au même temps de rétention ? Il aurait été très intéressant d'exploiter plus en

3. Remarque : on peut encore généraliser la notion de vallée à la notion de k-vallée qui pourrait être utile dans certains cas ou les données sont bruitées : la k-vallée d'un noeud p est le noeud v d'intensité minimum par lequel il faut passer pour atteindre k autres points de même intensité que p en prenant le chemin qui descends le moins bas possible. Mais nous n'avons pas étudié cette extension qui pourrais peut-être se révéler utile pour éliminer encore du bruit.

détail les possibilités offertes par cet algorithme pour l'étude de données LCMS, mais comme cela nous écarte du contexte SELDI-MS de cette thèse, nous n'avons pas mené d'avantage d'expérimentations.

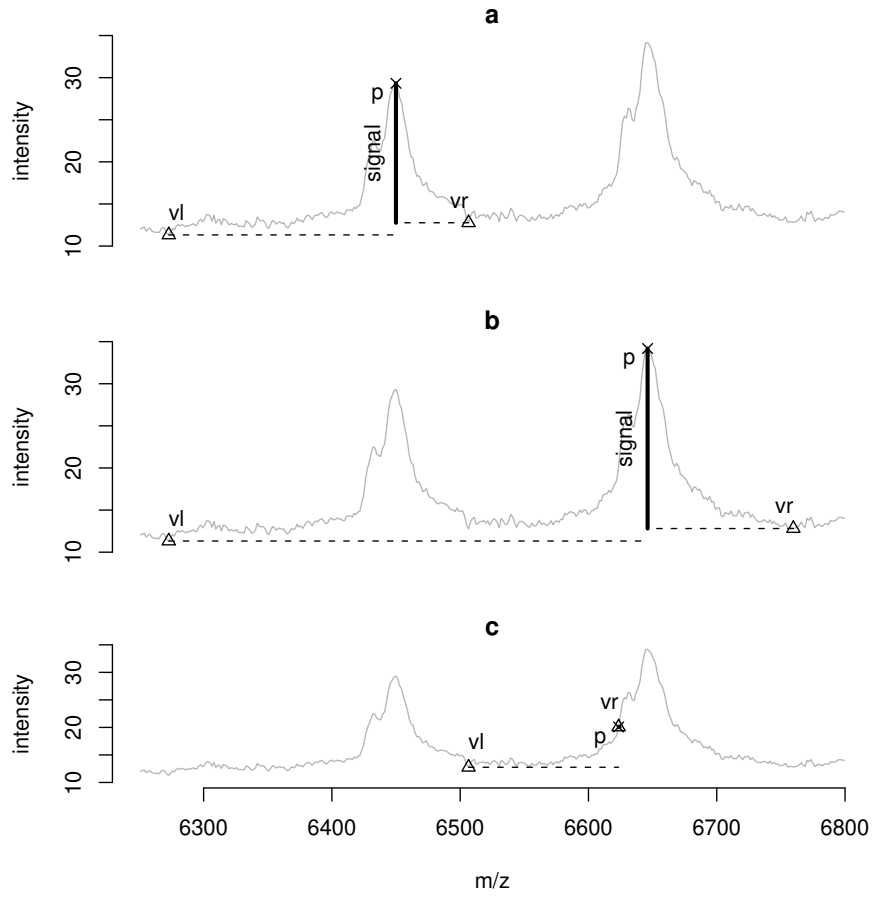


FIGURE 4.1 – Illustration de la définition des vallées et du signal pour 3 points p d'un spectre de masse. Les vallées v_l et v_r d'un point p sont déterminés en recherchant à sa gauche et à sa droite les points d'intensité minimum avant de rencontrer un point d'intensité supérieur à celle de p. Le signal correspond à la profondeur de vallée la plus petite. a) il n'y a pas de points d'intensité supérieur à p à sa gauche, mais il y en à un à sa droite qui limite le champs de recherche de v_r. Pour cette raison, v_r est situé entre les deux pics plutôt que après le deuxième pic. b) il n'y a pas de point d'intensité supérieur à p qui limite le champ de recherche des vallées, c'est pourquoi v_l et v_r sont les points d'intensité minimal située à gauche et a droite de p, même si pour v_l cela implique qu'il se situe avant le premier pic. c) p est situé sur la pente du deuxième pic. L'horizon de recherche de v_l est limitée par la présence du premier pic qui contient un point d'intensité supérieur à p. De plus, l'intensité du point situé immédiatement à la droite de p est supérieure à la sienne ce qui explique que p et v_r coïncident et implique un signal nul. a,b,c) Dans les trois cas, le signal est calculé par rapport à v_r car son intensité est à chaque fois supérieure à celle de v_l.

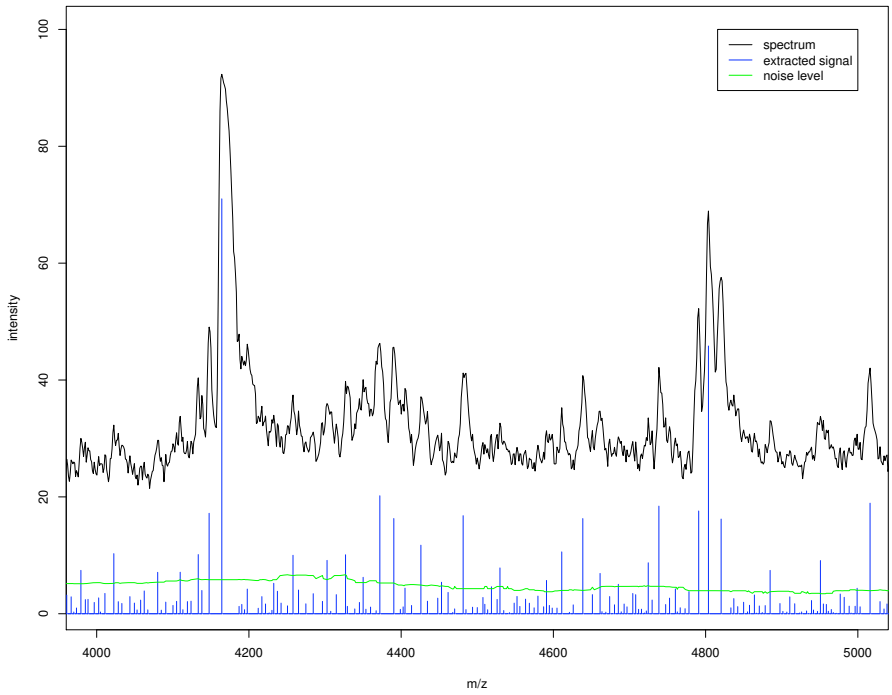


FIGURE 4.2 – Exemple de signal calculé avec l'algorithme 5. La ligne noire représente le spectre donnée en entré à l'algorithme, l'histogramme bleu est le signal calculé par l'algorithme. La ligne verte est une estimation du niveau de bruit calculé selon l'algorithme vu dans la section 3.3. Un pic est détecté lorsque le signal (en bleu) est au dessus du bruit (en vert).

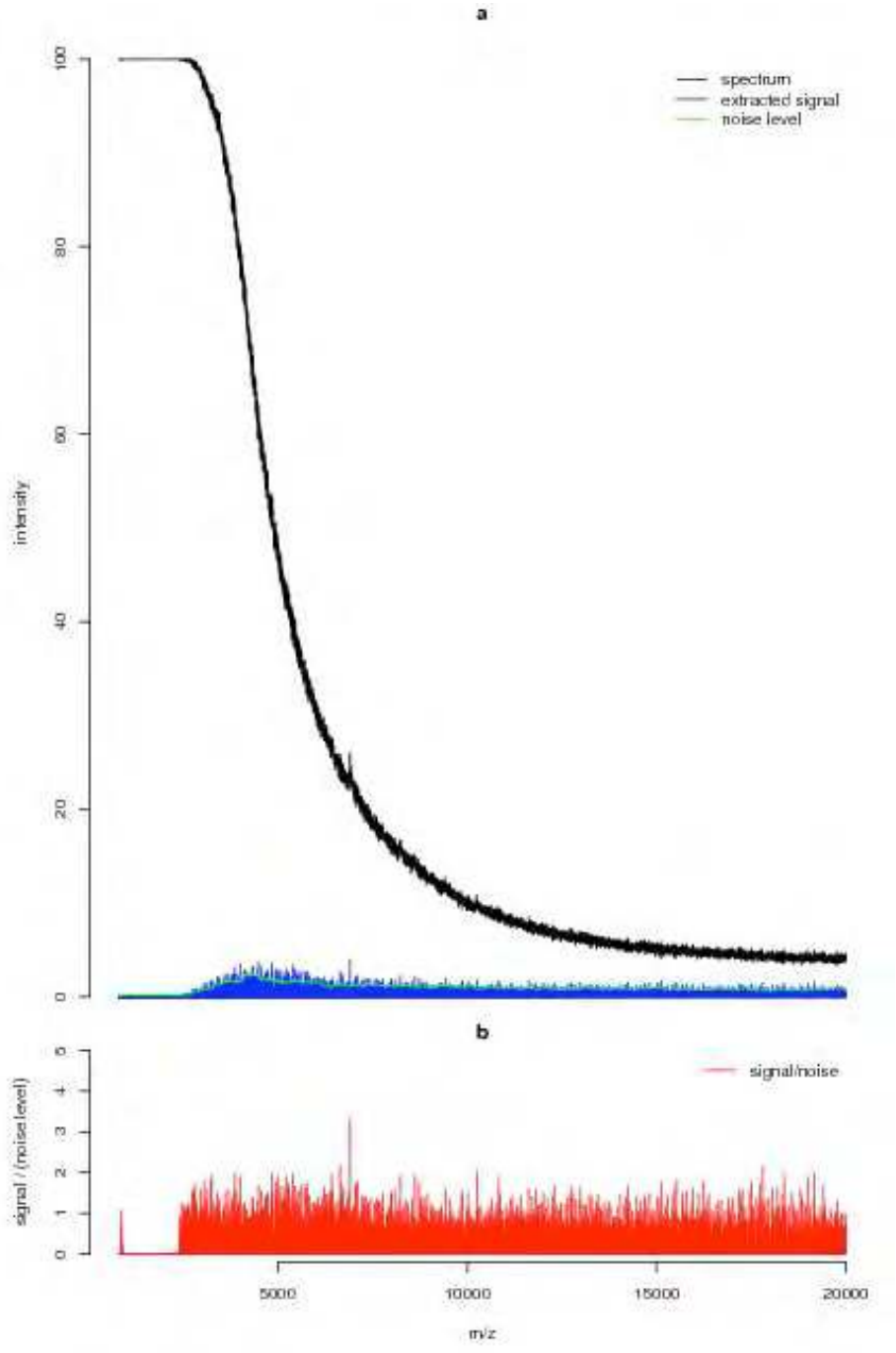


FIGURE 4.3 – Résultat de la détection de pic sur un spectre blanc. a- visualisation du signal (en bleu) calculé à partir du spectre (en noir), avec en vert le niveau du bruit. b- le rapport signal/bruit obtenu en divisant le signal (en bleu) par le niveau de bruit (en vert).

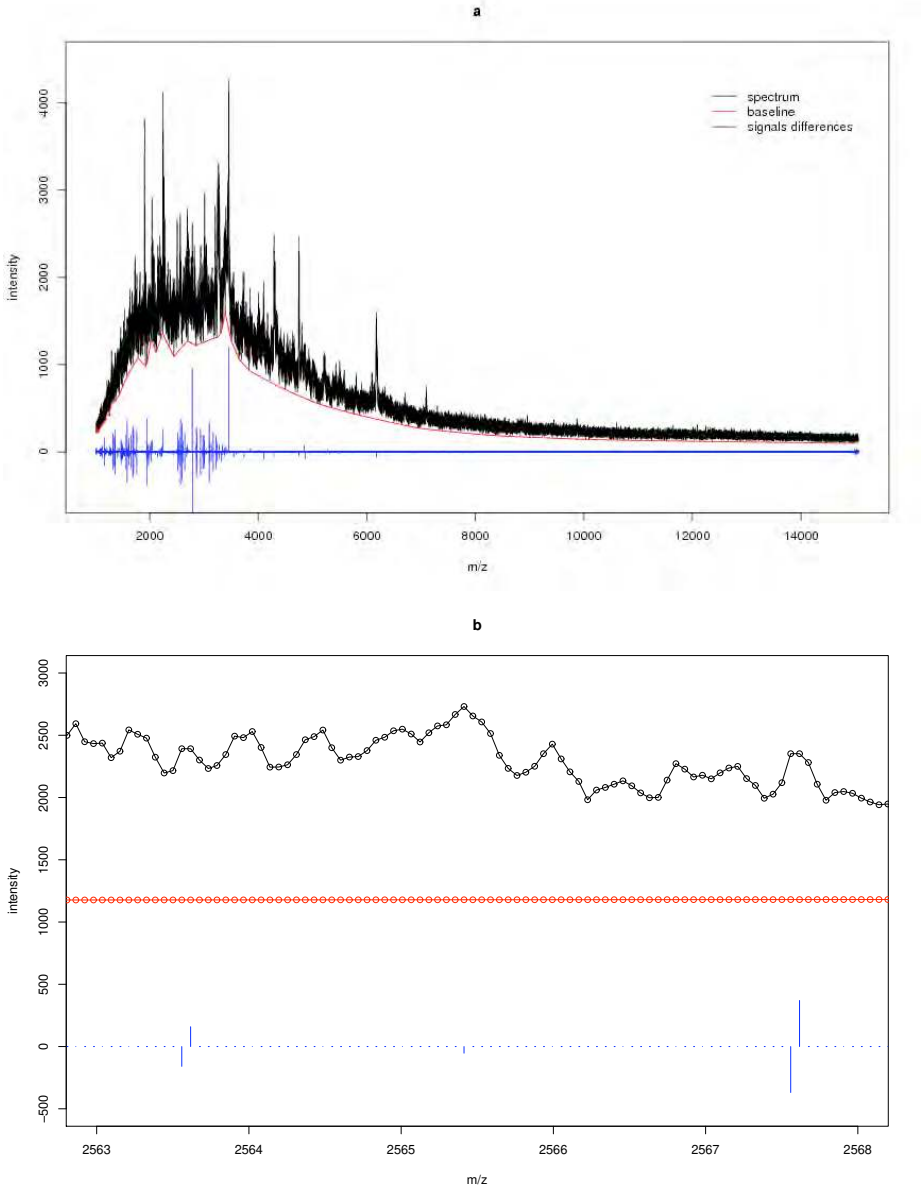
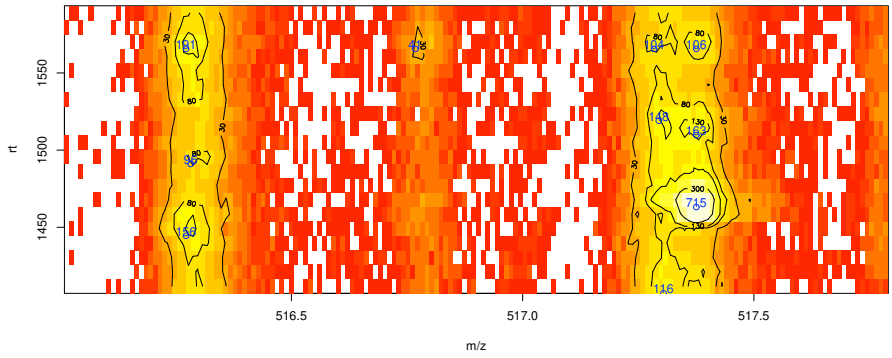


FIGURE 4.4 – a- Démonstration de l'insensibilité de la détection de pics à la suppression de la ligne de base ; l'histogramme bleu montre la différence entre les signaux calculés d'une part sur le spectre original (en noir) et d'autre part sur le spectres sans sa ligne de base (en rouge). b- un grossissement de la figure a, pour expliquer les différences dans les signaux.

a



b

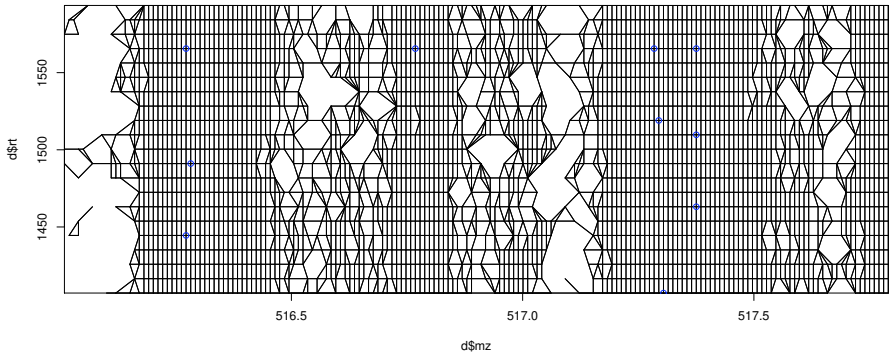


FIGURE 4.5 – a) Exemple d'une infime portion des données produite par une analyse LCMS. L'image est composée de plusieurs spectres de masse (un par ligne), l'axe des abscisses donne le m/z des points capturés, et la couleur indique leur intensité. L'axe des y représente le temps de rétention à laquelle le spectre de masse est produit. Les lignes noires sont des courbes de niveau, les nombres indiquent leur niveau d'intensité et aussi les intensités des pics. b) L'image montre le graphe de voisinage correspondant à l'image a). En bleu est indiqué les pics détectés avec l'algorithme 6.

Algorithm 6 graphValleys(G) : Algorithme de recherche des vallées dans un graphe de voisinage

```
1: Input : G a neighbor Graph with intensity value associated to each node
2: Output : valley[] a map wihch associate to each node of G his valley node
3: PriorityQueue pq ; {The neighborhood of the elements that have been pushed in the stack, in increasing intensity order}
4: Stack stack ; {A stack of element to process in decreasing intensity order.}
5: fill(valley[],UNKNOWN) ; {initialize the valley of each node}
6: for each node k of the graph s.t. (valley[k] == UNKNOWN) do
7:    minNode = {node with highest intensity in subgraph containing node k}
8:    pq.push(k) ;
9:    valley[pq.top] = INPQ ;
10:   while ( !pq.empty()) do
11:      if (pq.top.intensity < minNode.intensity) then minNode = pq.top ;
12:      {remove from stack elements lower than pq.top}
13:      while ( !stack.empty() and (stack.top.intensity ≤ pq.top.intensity)) do
14:         if (valley[stack.top].intensity < minNode.intensity) then
15:            swap(valley[stack.top],minNode) ;
16:         end if
17:         stack.pop() ;
18:      end while
19:      stack.push(pq.top) ;
20:      valley[stack.top] = minNode ;
21:      minNode = stack.top ;
22:      pq.pop() ;
23:      {put in pq all outgoing neighbor of stack.top}
24:      for each node i in the neighborhood of stack.top do
25:         if (valley[i] == UNKNOWN) then
26:            valley[i] = INPQ ;
27:            pq.push(i) ;
28:         end if
29:      end for
30:   end while
31:   {Clear stack}
32:   minNode = valley[stack.top] ;
33:   while (stack.size() > 1) do
34:      if (valley[stack.top].intensity < minNode.intensity) then
35:         swap(valley[stack.top],minNode) ;
36:      end if
37:      stack.pop() ;
38:   end while
39:   valley[stack.top] = minNode ;
40:   stack.pop() ;
41: end for
```

4.3 Aire, largeur et hauteur des pics

L'algorithme de détection des pics que nous avons vu dans la section 4.2.2 est capable de calculer le signal caché derrière un spectre de masse et d'extraire la position des pics, c'est à dire leur m/z. De plus, nous avons vu qu'il peut être exécuté directement sur les données brutes des spectres de masse car il ne nécessite pas de lissage du signal et qu'il est très peu sensible à l'élimination de la ligne de base. Afin maintenant d'extraire les informations relatives à l'expression des protéines cachées derrière les pics, il faut maintenant nous intéresser à leur intensité (ou hauteur), et à leur aire qui est réputée être une meilleure estimation de l'expression des protéines [94]. Dans la littérature, l'intensité des pics est généralement donnée par l'intensité du spectre de masse à l'endroit ou se trouve le pic une fois que la ligne de base a été éliminé. L'aire des pics se calcule généralement en considérant l'aire sous la courbe MS située sur toute la largeur du pic (souvent après lissage et élimination de la ligne de base). Pour déterminer cette largeur, on peut, comme Li et al. [35], se contenter de considérer que la résolution du spectre est constante mais celle-ci est parfois difficile à estimer dans les spectres SELDI-TOF. Il apparaît donc préférable de chercher à délimiter dans le spectre la largeur de la région influencée par le pic comme le font Fung et Enderwick [32] et Pratapa et al. [94]. Pratapa propose par exemple de rechercher de chaque coté du pic, la première position où le gradient du spectre lissé change de signe, mais il se peut que ce point soit très éloigné du pic et le problème du paramétrage du lissage se pose.

Dans notre travail en revanche, nous cherchons à définir une procédure de calcul des intensités et des aires qui ne nécessite ni l'élimination de la ligne de base, ni un lissage des spectres, à l'inverse des approches mentionnées. Si nous parvenons à cela, nous aurons définit une procédure de détection de pics et d'extraction des informations d'expression qui peut être exécutée directement sur les données brutes, et qui nécessite un minimum d'étape de pré-traitement (l'estimation du niveau de bruit est éventuellement nécessaire pour déterminer les pics, mais sinon pas besoin de lissage des spectres ni d'éliminer la ligne de base,). Notre approche passe par la définition d'un modèle qui caractérise la forme d'un pic et qui va être ajusté sur les données afin d'extraire les informations recherchées.

Souvent, le modèle que l'on cherche à ajuster sur les pics est de forme gaussienne [39]. La méthode que nous proposons dans [90], et que nous reprenons maintenant repose à l'inverse sur un modèle linéaire en quatre morceaux qui n'est pas symétrique par rapport au sommet du pic comme un modèle gaussien peut l'être. Notre modèle est illustrée sur la figure 4.6, il est ajusté une fois que l'on a déterminé les sommets p des pics à l'aide de l'algorithme vu dans la section 4.2.2. Ce modèle est composé de deux segments obliques partant du sommet p et qui s'ajuste sur les pentes du pic avec à leurs extrémités deux segments horizontaux pour modéliser la base du pic. Étant donnée les positions p, l'algorithme d'ajustement du modèle se charge de le découper le spectre en région en sectionnant au niveau des points d'intensité minimum situés entre

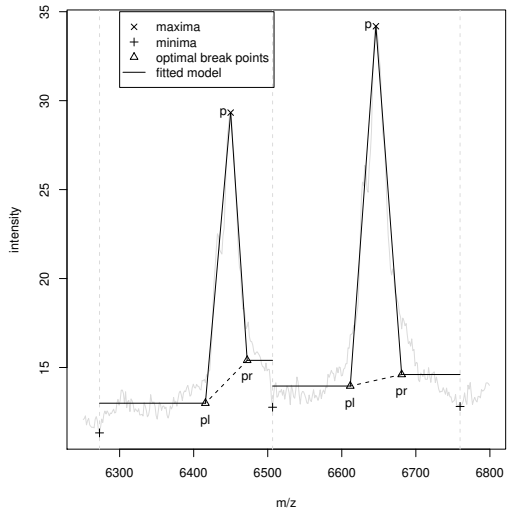


FIGURE 4.6 – Illustration de l'ajustement du modèle linéaire en deux morceaux pour le calcul de l'aire d'un pic. Le modèle est constitué du segment joignant p à p_r et de la demi-droite horizontal d'origine p qui est ajusté de manière optimal sur les données.

deux pics consécutifs. De cette manière, chaque région ne contient qu'un seul pic et on opère individuellement sur chacune un ajustement des modèles. L'algorithme recherche donc à l'intérieur de chaque région le point de départ p_l et le point final p_r du pic respectivement situés à gauche et à droite du sommet p. La position de p_r (resp. p_l) est déterminée en ajustant optimalement (au sens des moindres carrées) le modèle linéaire en deux morceaux constitué du segment joignant les points p et p_r (resp. p_l), et du segment horizontal d'origine p_r (resp. p_l). Formellement, le point p_r est donc obtenu à partir de l'équation 4.3 qui minimise à la fois l'erreur sur le segment oblique (équation 4.1) et sur le segment horizontal (équation 4.2) – dans ces formules, p^x and p^y sont respectivement l'index et l'intensité du point p dans le spectre de masse –.

$$\mathrm{err}_{\mathrm{seg}}(p_r) = \sum_{i=p^x}^{p_r^x - 1} (s_y[i] - \frac{p_r^y - p^y}{p_r^x - p^x}(i - p^x))^2 \tag{4.1}$$

$$\mathrm{err}_{\mathrm{ray}}(p_r) = \sum_{i=p_r^x}^{end} (s_y[i] - p_r^y)^2 \tag{4.2}$$

$$p_r = \underset{p_r}{\mathrm{argmin}}(\mathrm{err}_{\mathrm{seg}}(p_r) + \mathrm{err}_{\mathrm{ray}}(p_r)) \tag{4.3}$$

D'un point de vu pratique, la solution de ce problème d'optimisation est

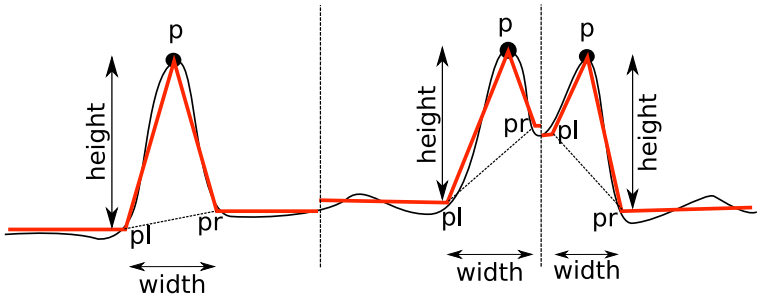


FIGURE 4.7 – Extraction des caractéristiques (largeur, hauteur, aire) des pics, avec et sans chevauchement. Lorsque deux pics se chevauchent, la hauteur des pics n'est pas influencé par la cuvette qui se forme car dans le calcul on considère seulement la vallée gauche/droite la plus profonde pour chaque pics.

trouvée en énumérant exhaustivement les abscisses p_r^x de p_r (resp. p_l^x de p_l) pour chaque index entre p et la fin de la région. L'ordonnée p_r^y, de p_r, est obtenue par optimisation classique des moindres carrées. Au final, l'algorithme à une complexité très faible car pouvons ajuster les modèles de tous les pics en $\mathcal{O}(n)$, où n est le nombre de valeurs d'échantillonnage dans tous le spectre de masse.

Une fois les modèles ajustés aux données, nous définissons la hauteur du pic p comme $h = \max(p^y - p_l^y, p^y - p_r^y)$, la largeur du pic comme la distance $p_r^x - p_l^x$, et l'aire du pic p comme la surface du triangle (p_r, p, p_l). Noter que ces dernières valeurs dépendent de l'unité utilisé en abscisse. Le plus logique est d'utiliser le temps de vol comme unité car les valeurs d'échantillonnage sont espacés régulièrement. Noter aussi que dans le cas ou deux pics se chevauchent, une cuvette se forme entre les deux pics à une intensité élevée, et la base des pics sera localisée dans cette cuvette. Par contre, la hauteur h du pic ne sera pas influencé par ce phénomène car dans son calcul on considère le maximum des deux hauteurs (voir exemple de la figure 4.7). Aussi, remarquer que pour l'aire du pic, nous préférons considérer la surface du triangle qui se forme dans le modèle plutôt que l'aire sous la courbe dans la région p_l-p_r. Nous avons ainsi moins de problèmes avec les pics qui se chevauchent, l'estimation de l'aire est plus robuste au bruit, et nous évitons les problèmes si la ligne de base est mal estimée dans certaines régions du spectre. Effectivement, comme pour l'algorithme de détection de pics, comme les trois points p, p_l et p_r qui définissent le pic sont définies localement et que le calcul de l'aire ne fait intervenir que des différences, l'élimination de la ligne de base n'affecte pas beaucoup le calcul des aires.

Mentionnons également une alternative intéressante pour estimer l'aire des pics lorsque nous ne possédons pas le signal brute, mais seulement l'intensité et

la position des pics ce qui peut arriver dans des situations ou un opérateur à déjà opérer la détection des pics. Pour cela, rappelons que dans la section 3.2, nous avons vu que la transformation d'un spectre de masse SELDI-TOF qui utilise en abscisse le logarithme du temps de vol génère des signaux dont les pics ont des largeurs plus uniforme que dans les spectres originaux. Faisons la supposition que cette transformation génère des pics de largeurs constantes w tout au long du spectre. Un pic qui se trouve initialement à la position t occupe donc, après transformation logarithmique, la région qui va de $\left(\log(t) - \frac{t}{2}\right)$ jusqu'à $\left(\log(t) + \frac{w}{2}\right)$. Cela correspond dans le spectre initial à la région de spectre qui débute en $p_l = \left(e^{\log(t) - \frac{w}{2}}\right)$ et se termine en $p_r = \left(e^{\log(t) + \frac{w}{2}}\right)$, or la longueur de cette région est proportionnelle au temps de vol t où se trouve le pic :

$$p_r - p_l = t(e^w - e^{-w}) \propto t$$

Ce résultat suggère donc que la largeur des pics augmente proportionnellement avec leur temps de vol dans le spectre. Ce résultat n'est pas surprenant et revient à faire l'hypothèse couramment faite que la résolution[4] des spectres SELDI-TOF est constante au long du spectre [90, 35, 32, 5, 116]. Faisons maintenant la supposition que les pics ont une forme de triangle isocèle dont la hauteur h est égale à la hauteur du pic, et la base est égale à la largeur du pic. Comme la largeur de la base est proportionnelle à la position t du pic, alors son aire A suit la formule : $A \propto ht$. En conséquence nous pouvons estimer l'aire d'un pic simplement en multipliant sa hauteur par son temps de vol.

4.4 Bilan sur la détection des pics

Avant de poursuivre, il est bon de faire le point sur le pipeline de pré-traitement des spectres de masse que nous proposons dans les chapitres 3 et 4. La figure 4.8 rappel le diagramme du flux des données dans le pipeline de pré-traitement. Rappelons que nous avons essayé de limiter le nombre d'étapes qui se succèdent pour passer du spectre original à la liste de pics pour réduire au maximum le risque de perdre des informations dans ce processus de pré-traitement. C'est pourquoi, on remarque que "l'élimination de la ligne de base" apparaît sur le diagramme, mais qu'elle n'entre pas dans le pipeline qui génère la listes des pics. Les algorithmes que nous avons définis pour la détection des pics et l'extraction de leur caractéristiques (aire, largeur) sont effectivement peu sensible à la présence de la ligne de base si bien qu'il n'est pas nécessaire de l'ôter – le calcul de la ligne de base est cependant utile pour vérifier si les résultats sont identiques qu'elle soit présente ou pas, à celui qui souhaite estimer le courant ionique total, et pour remplir les valeurs manquantes de la matrice d'expression comme nous le verrons dans le chapitre suivant. De la même manière, le lissage du spectre de masse est marqué comme facultatif, car nous avons tenté de définir des algorithmes qui soient peu sensible au bruit pour pouvoir les appliquer directement sur les données brutes, d'autant que

4. la résolution est égale au rapport $\frac{\text{position du pic}}{\text{largeur du pic}}$.

le lissage d'un spectre de masse est une opération sensible qui peu mener à une détérioration des informations qu'ils contiennent. Pourtant nous verrons dans le chapitre 7 que cette opération améliore la stabilité de la détection de pics. Enfin soulignons l'aspect "universelles" des algorithmes qui composent le pipeline en mentionnant en particulier l'algorithme de calcul des signaux. Celui-ci est basé sur une notion de pic et de vallée, au départ définies pour les séquences d'intensités que l'on extrait des spectres, mais qu'il est possible d'étendre à des graphes de voisinage qui définissent des maillages plus complexes et peuvent représenter une information spatiale plus riche comme par exemple des données de grande dimension (image 2D, vidéo 3D, etc...).

Ce pipeline permet donc de traiter individuellement chaque spectre de masse pour extraire leurs pics et leurs caractéristiques. Mais, une expérience de spectrométrie de masse est composée de plusieurs spectres de masse pour lesquels on souhaite comparer les pics qu'ils contiennent. Il est nécessaire pour cela de construire une matrice d'expression en passant par une étape de "fusion" des listes de pics extraites des différents spectres de masse. Le but de cette fusion est d'identifier les pics qui ont la même protéine sous-jascente dans les différentes expériences. Cette fusion est justement l'objet du chapitre suivant est constitue la dernière étape de pré-traitement avant de pouvoir étudier les données et les exploiter avec des algorithmes d'apprentissage automatique.

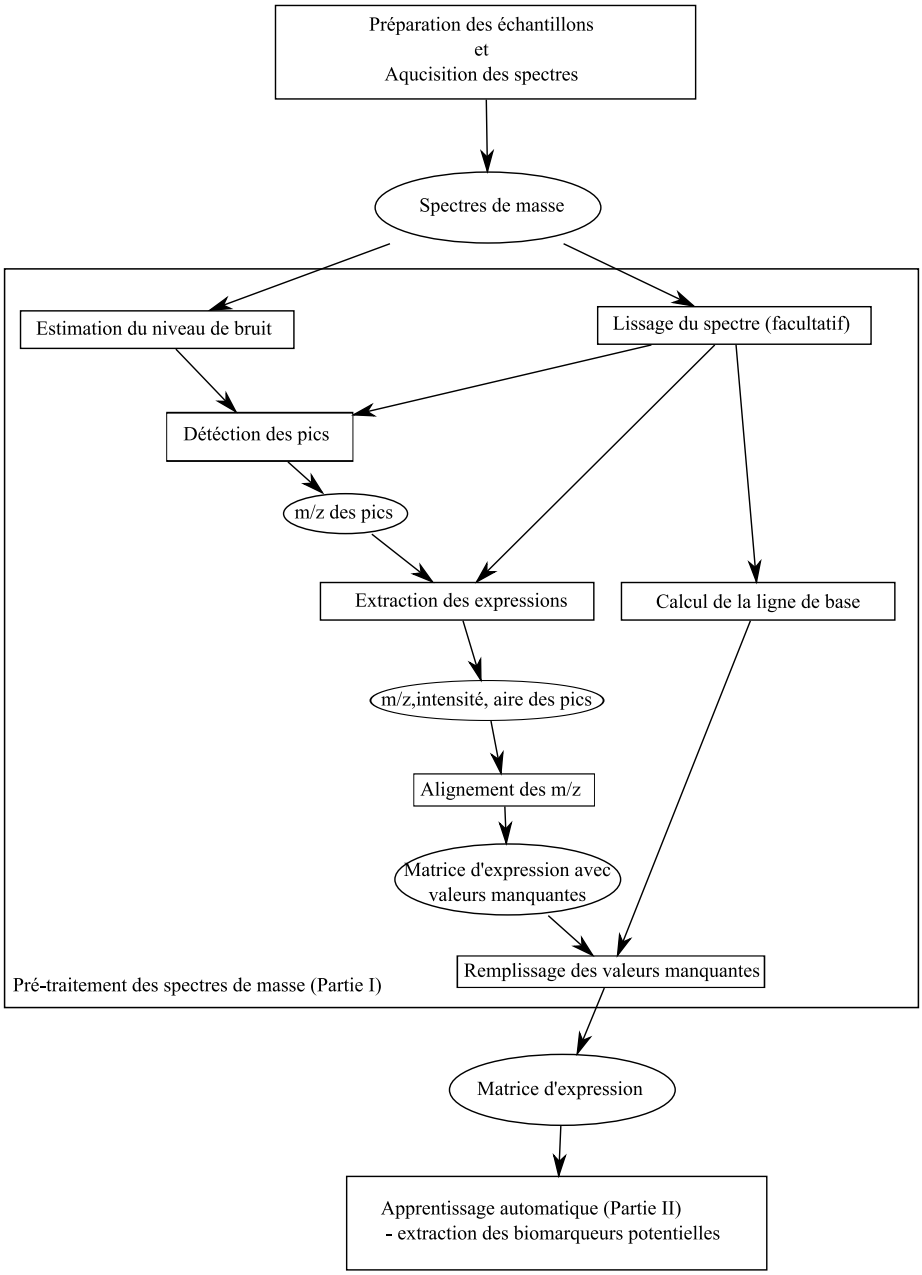


FIGURE 4.8 – Diagramme du flux des données dans le pipeline de pré-traitement.

Chapitre 5

Alignement et fusion des pics

Les méthodes de traitement du signal et de détection des pics qui ont été évoquées jusqu'ici travaillent (essentiellement) sur un spectre à la fois, sans considérer les autres spectres des expériences. Au terme de leur exécution, ces pré-traitements fournissent pour chaque spectre une liste de pics qui se veut robuste aux variations expérimentales (ligne de base, bruit, etc.) et qui contient au minimum la position en m/z des pics – elle peut aussi indiquer d'autres informations tel que leur intensité, leur aire, leur largeur, leur état de charge, etc... –. Ce chapitre traite maintenant de la fusion des listes de pics extraites des différents spectres de masse. Cette phase constitue l'étape finale du pré-traitement qui permet de construire la matrice d'expression X nécessaire aux algorithmes d'apprentissage automatique. Rappelons que, idéalement, cette matrice doit contenir autant de lignes que de spectres de masse considérés, et autant de colonnes que de protéines distinguables dans les échantillons ; chacune des cases ij de la matrice doit indiquer la concentration de la protéine j dans l'échantillon i.

Comme Fung et al, [32], nous distinguons deux phases dans la fusion des listes de pics : la première est une étape d'alignement des différentes listes de pics pour y identifier ceux qui dissimulent la même protéine ; et la seconde a en charge de générer la matrice d'expression en complétant les informations manquantes. Les deux sections qui constituent ce chapitre traitent respectivement de ces sujets.

5.1 Alignement

La construction de la matrice d'expression nécessite identifier dans les différents spectres de masse les pics qui dissimulent la même protéine sous-jascente. En théorie, ces pics devraient avoir exactement la même position m/z, mais dans la réalité leurs positions fluctuent à cause de variations dans la calibration des instruments et des incertitudes sur l'emplacement exact des pics à la détection.

L'alignement des pics cherche à rectifier ces décalages et à unifier les pics des différents spectres qui correspondent à la même protéine.

5.1.1 Calibration

Un bon alignement des pics passe tout d'abord par une bonne calibration des instruments. La calibration est une étape essentielle dans le protocole expérimentale qui vise à régler les instruments et qui à un effet sur la distorsions des spectres de masse selon leur axe m/z. La calibration d'un instrument consiste à ajuster l'axe m/z des spectres de masse d'un échantillon de référence, dont le contenu est connue, de manière à ce que les pics observés coïncident avec leur position théorique. En ce qui concerne les instruments de type "temps de vol", la calibration agit essentiellement au niveau des paramètres de l'équation quadratique qui convertit le temps de vol des ions en leur m/z pour calibrer les instruments [32]. Si cette équation est inconnue, il est généralement suffisant de rechercher la transformation affine qui permet d'ajuster au mieux les m/z des pics expérimentaux avec les m/z des pics théoriques [48, 53]. Pourtant même avec des instruments bien calibrés, de petites erreurs sur la position des pics persistent. On estime par exemple pour nos données SELDI-TOF que l'erreur m_{err} est de l'ordre de $\pm 0.3\%$, c'est à dire que le pic d'une molécule dont la masse est 10000Da peut avoir un m/z qui varie entre 9970Da et 10030Da.

La calibration des spectres de masse peut également être vu comme un alignement des signaux brutes. A ce sujet mentionnons l'existence des algorithmes de type Dynamic Time Warping (DTW) et Correlation Time Warping (COW), qui sont capables de produire des alignements non-linéaires d'un signal sur un autre en utilisant les concepts de la programmation dynamique. Contrairement aux méthodes de calibration, ces algorithmes n'ont pas besoin de points de référence pour trouver l'alignement, en revanche le résultat qu'ils produisent peut rapidement devenir incohérent si la mesure de distance choisie n'est pas appropriée. Ces algorithmes ont été employé dans le cadre de l'alignement d'expériences de spectrométrie de masse, toutefois leur utilisation s'est limité aux expériences de type LCMS et à l'alignement des temps de rétention entre plusieurs expériences [118, 93, 95]. Ils sont effectivement mieux adaptés à ce problème, plutôt qu'à l'alignement des m/z, du fait de la non-linéarité des transformations qu'ils peuvent extraire.

Les erreurs de positionnement des pics qui persistent après calibration des spectres posent encore des problèmes pour identifier les pics des différents spectres qui cachent la même protéine et qui forment une colonne de la matrice d'expression. La figure 5.1 montre un exemple simple et exagéré de configuration où il est difficile de décider de l'alignement des pics. Cette figure montre la position de 4 pics détectés dans 3 spectres de masse différents, et où se pose la question du regroupement le plus probable. Parmi les possibilités nous avons : 1) les 4 pics correspondent à 4 protéines protéines différentes ; 2) il y a une protéine derrière les pics P1-P2-P3, et une autre derrière P4 ; 3) il y a une protéine derrière les pics P1-P3, et une autre derrière les pics P2-P4 ; etc... La section qui suit, propose un algorithme d'alignement des pics pour résoudre ce type de

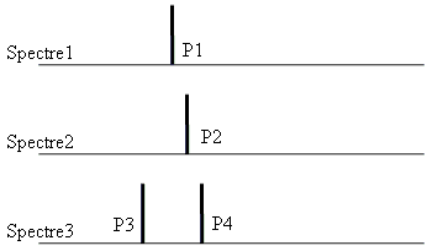


FIGURE 5.1 – Exemple de configuration difficile à aligner.

problème.

5.1.2 MZCL

Pour résoudre au mieux le problème d'alignement des pics, nous proposons dans Prados et al. [89] un algorithme de clustering hiérarchique, baptisé *MZCL* (pour m/z clustering). *MZCL* apporte une amélioration importante par rapport aux méthodes d'alignement antérieures qui consistait en un tri des pics selon leur m/z et une fenêtre glissante pour effectuer le regroupement [1, 108, 122] et qui pouvait hélas parfois aboutir à des résultats incohérents. A la différence de ces méthodes, *MZCL* utilise une méthode non supervisée de clustering hiérarchique pour créer des regroupements compacts de pics (des clusters) en fonction de leur position m/z. Les groupes de pics qu'il construit sont plus cohérent, et correspondent d'avantage aux protéines que l'on peut distinguer dans les échantillons. On retrouve d'ailleurs par la suite les idées de *MZCL* dans les articles de Tibshirani et al. [116] et Pratapa et al. [94]. Le fonctionnement de *MZCL* est illustrée figure 5.2 et nous décrivons son fonctionnement maintenant.

MZCL considère au départ qu'il y a autant de protéines dans les échantillons que ce qu'il y a de pics dans les spectres de masse, ce qui signifie qu'il construit initialement autant de groupe que de pics. Puis il fusionne tour à tour les deux groupes les plus proches en m/z en considérant que la masse exacte de la protéine sous-jacente est égale à la moyenne des m/z des pics qui constitue le groupe. En d'autres termes, il s'agit d'un algorithme de clustering hiérarchique agglomératif avec "centroïde-linkage" [30] qui se base sur la position m/z des pics détectés. En plus de cela, deux contraintes additionnelles sont insérées dans la procédure pour respecter la nature spéciale du problème. Ces contraintes imposent lors de la fusion de deux groupes de pics que 1) tous les pics du nouveau groupe proviennent de spectres différents, et 2) l'erreur maximum entre deux pics du nouveau groupe soit inférieure à l'erreur de l'instrument. Ainsi, sur la figure 5.2, il n'y a pas de fusion à l'étape 5 car la première contrainte serait violée (le nouveau groupe contiendrait deux pics du spectre 1 et deux du spectre 3) ; et idem à l'étape 6 pour la deuxième contrainte.

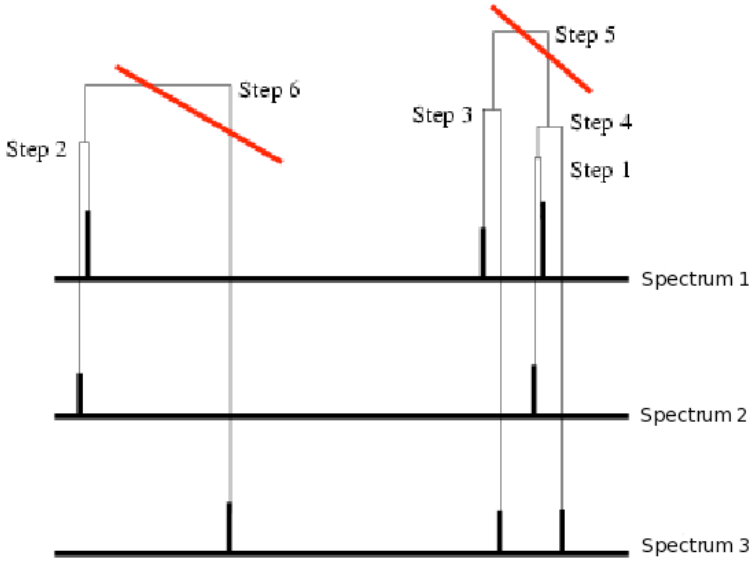


FIGURE 5.2 – Illustration du fonctionnement de l'algorithme *MZCL* pour l'alignement des pics.

Pour déterminer les deux groupes de pics les plus proches qu'il faut fusionner, nous n'utilisons pas directement le m/z des pics car nous souhaitons aussi prendre en compte l'erreur de positionnement des pics (m_{err}) qui augmente proportionnellement avec le m/z. En effet, supposons que l'on ait le choix entre fusionner le pic situé à 999Da avec celui à 1001Da ou de fusionner le pic situé à 4998Da avec celui à 5002Da. Les deux premiers sont espacés de 2Da et les deux seconds de 4Da, donc si on considère ces écarts en absolue pour décider des pics à fusionner nous fusionnerions en premier les pics 999Da et 1001Da. Cependant comme l'erreur augmente proportionnellement avec la position en m/z, il est plus probable que les pics 4998Da et 5002Da soient issus de la même distribution que les pics 999Da et 1001Da. C'est pourquoi, plutôt que de fusionner les pics les plus proches en m/z nous préférons fusionner ceux dont l'erreur relativement à leur moyenne est la plus faible. L'erreur est donc calculée au moyen de l'équation 5.1, et cette valeur peut être comparée directement avec l'erreur de l'instrument ($m_{\text{err}} = \pm 0.3\%$).

$$\text{error}(m_i, m_j) = \frac{|m_i - m_j|}{\mu}; \mu = \frac{|m_i + m_j|}{2} \tag{5.1}$$

L'algorithme 7 fournit le pseudo code pour *MZCL*. Il utilise la fonction booléenne *mergeable*, définie par l'algorithme 8, qui est vrai si les groupes (clusters) passés en paramètres ne violent pas les deux contraintes de fusion. Il a été implémenté avec une complexité de $\mathcal{O}(n \log(n))$, où n est le nombre de pics dans l'ensemble des spectres de masse.

Algorithm 7 mzcl(p,$m_{\text{err}} = 0.003$) : la procédure de d'alignement de pics. Entrées : m_{err}, la tolérance de l'instrument ; p, l'ensemble des pics détectés dans les spectres.

With the n peaks in p, generate n clusters C_i of one peak
loop
  $(C_i, C_j) \leftarrow \min_{i \neq j}(\text{error}(\text{centroid}(C_i), \text{centroid}(C_j)))$
  if (no more candidate) **then end loop**
  if $(\text{error}(\text{centroid}(C_i), \text{centroid}(C_j)) > 2m_{\text{err}})$ **then end loop**
  if mergeable(C_i,C_j,e) **then** merge(C_i, C_j)
end loop
return the clusters

Algorithm 8 mergeable(C_i,C_j,$m_{\text{err}} = 0.003$) : a prédicat vrai si les deux clusters C_i et C_j satisfont les contraintes de fusion.

if $\max_{(p_i,p_j) \in C_i \times C_j} (\text{error}(p_i, p_j)) > 2e$ **then**
  {C_i and C_j have two peaks outside the error tolerance}
  return *false*
else if all peaks of C_i and C_j are in different spectra **then**
  return *true*
else
  return *false*
end if

Les groupes de pics formés par l'alignement servent à générer la structure de la matrice d'expression qui est ensuite utilisée pour l'apprentissage automatique. La matrice comporte autant de ligne qu'il y a de spectres de masse dans les expériences, et autant de colonnes que de groupes de pics identifiés par l'alignement MZCL. Les cases de la matrice sont remplies soit avec l'intensité du pic correspondant, soit avec son aire – qui est sensée être une estimation plus précise de la concentration de la protéine dans l'échantillon –.

Le problème qui se pose lors de la génération de la matrice, est que les groupes de pics identifiés par alignement ne contiennent pas forcement un pic dans chaque spectre. Ceci peut provenir d'un défaut de détection ou simplement de l'absence du pic dans le spectre concerné. En conséquence, la matrice d'expression que l'on peut construire après alignement n'est pas complète est montre des valeurs manquantes. Par exemple, l'alignement des trois spectres de la figure 5.2 comporte quatre groupes qui impose que la matrice d'expression soit de taille 3x4. Par contre, seul la quatrième colonne est complète car seul son groupe comporte 3 pics (1 dans chaque spectre), pour les autres les pics manquants se traduisent par des valeurs manquantes dans la matrice d'expression. Ceci soulève deux questions importantes : Comment remplir les valeurs manquantes ? Que faire avec les groupes qui couvrent peu de spectres ? Ces questions, déjà discuté dans Prados et al. [90], sont discutées à nouveau dans la section suivante.

5.2 Construction de la matrice d'expression

5.2.1 Traitement des groupes à faible couverture

Il arrive que certains groupes formées durant l'alignement aient beaucoup de pics manquants. Dans le cas extrêmes, ils peuvent même contenir qu'un seul pic, c'est à dire que parmi tous les spectres de masse, un seul pic est présent dans cette région du spectre. Ceci constitue une indication que le pic en question est en réalité du bruit, car sinon nous aurions certainement trouvé ce pic dans d'autres spectres. Plus généralement, moins un groupe contient de pics, et plus il y a de chance qu'il corresponde à du bruit. C'est pourquoi, nous estimons qu'il est mieux d'éliminer de la matrice d'expression les colonnes qui ont plus de 95% de valeurs manquantes.

Le nettoyage que nous réalisons par cette opération est une manière d'exploiter les corrélations entre les spectres de masse, et offre la possibilité d'éliminer des pics qui correspondent à du bruit. Cette procédure à également une conséquence intéressante sur le paramétrage de la détection des pics. Il est effectivement assez difficile de régler la sensibilité de la détection de pics (c'est à dire de régler le rapport signal/bruit minimum pour considérer que l'on est en présence d'un pic), or comme nous réalisons ce nettoyage lors de la construction de la matrice d'expression, nous préférons adopter une politique de détection sensible. Ainsi, on détecte un assez grand nombre de pics dans chaque spectre de masse, au risque de détecter des pics de bruits, et on compte sur le nettoyage de la

matrice pour éliminer les faux pics.

5.2.2 Remplissage des valeurs manquantes

Les "trous" qui reste dans la matrice d'expression après alignement des pics peuvent poser des soucis pour l'apprentissage automatique car peu d'algorithmes de classification savent traiter les valeurs manquantes. Il est donc important de les remplir de manière informative. On peut envisager pour cela deux stratégies :

1. soit remplacer les valeurs manquantes par la valeur zéro pour signifier que la concentration de la protéine est absente de l'échantillon car aucun pic ne correspond à la masse de cette protéine. Cette stratégie fait toutefois l'hypothèse que la détection de pic est très fiable et n'oublie aucun pic.
2. soit remplacer les valeurs manquantes par l'intensité du spectre à l'endroit où devrait se trouver les pics manquants. Mais cette stratégie de remplissage ne permet que de traiter des matrices d'expression contenant des intensités. Elle est plus difficile à mettre en oeuvre pour remplir des matrices d'expression contenant des aires car se pose alors le problème de déterminer la largeur et l'aire d'un pic indécelable.

Dans [90], nous avons comparé ces deux stratégies de remplissage, et la seconde s'est révélée la meilleure dans le sens ou nous avons réussi à extraire davantage d'information des données. Ces résultats seront également présenté dans le chapitre 7.

5.3 Bilan de l'alignement

Ce chapitre sur l'alignement des pics et le remplissage de la matrice d'expression nous a permis de décrire la dernière étape du pré-tratiement des données. A ce sujet, la méthode de clustering hiérarchique que nous proposons pour réaliser l'alignement des spectres SELDI-TOF est une contribution intéressante de notre travail par rapport au méthodes existantes auparavant. Néanmoins, ses limites restent à mon avis très importantes, car elle ne fonctionne que si les spectres sont correctement calibrés, et elle est difficile la généraliser pour l'alignement de données bi-dimensionnelle de type LCMS. Sur ce point, Lange et al. [68] adopte a mon avis la bonne démarche pour aligner les expériences LCMS. L'algorithme qu'elle propose commence par calculer les distances entre tous les couples de pics présents à l'intérieur d'une expérience, puis compare ces distances entre deux expériences pour en déduit la meilleure correspondance entre les pics. L'idée utilisée ici est proche de celle que nous avons évoquée dans la section 3.4.1 pour rendre nos méthodes insensible à la normalisation des spectres de masse : les positions des pics ne sont pas comparées directement d'une expérience à l'autre, mais elles sont d'abord comparées à l'intérieur d'une expérience avant que l'on analyse le résultat de cette comparaison entre les expériences. En procédant ainsi, de manière relative, on s'abstrait plus facilement des positions absolus des pics qui sont soumis à des aléas, alors que les informations relatives à l'intérieur d'une même expérience sont plus robustes.

Mentionnons également le travail de Ressom et al. [101] qui propose un moyen d'incorporer l'alignement dans le processus d'apprentissage. L'idée de ce travail est de ne pas prendre de décision d'alignement durant le pré-traitement, mais de construire un jeu de données avec plusieurs possibilités d'alignement, puis de laisser choisir le processus d'apprentissage avec des critères de performance de classification. Dans le chapitre 7, nous allons voir comment exploiter ce critère de performance, ainsi que des critères de stabilité, pour évaluer la qualité de notre pipeline de pré-traitement qui est maintenant complètement décrit. Mais avant cela, dans le chapitre 6, voyons comment ont été exploité ces algorithmes de pré-traitement pour étudier la reproductibilité des protocoles biologiques.

Chapitre 6

Evaluation des protocoles de préparation MALDI

En introduction, nous avons vu que la reproductibilité des expériences MALDI-TOF et SELDI-TOF pose un certain nombre de problèmes (section 2.3). Ces problèmes peuvent provenir de plusieurs sources, et on peut donc agir à plusieurs niveaux pour tenter d'améliorer la reproductibilité globale des analyses (section 2.3.1). Dans le chapitre 7, nous cherchons à optimiser la qualité des méthodes de pré-traitement bio-informatique qui permettent d'extraire la matrice d'expression à partir des spectres de masse. Les études menées dans ce chapitre 7 permettent d'avoir une vision claire pour guider notre choix des méthodes de pré-traitement. Mentionnons en particulier, la section 7.3.2 qui donne une idée de l'impact du pré-traitement sur la reproductibilité des analyses menées. Dans ce nouveau chapitre, c'est par contre la reproductibilité des protocoles de préparation des échantillons qui nous intéresse.

Effectivement, dans le présent chapitre, nous étudions la reproductibilité des expériences MALDI-TOF pour huit protocoles différent de préparation des échantillons biologiques et d'acquisition des spectres de masse. Nous allons voir que le choix du protocole à un impact important sur la reproductibilité des expériences, et au terme de ce chapitre nous sommes en mesure de donner le protocole le plus robuste pour l'analyse des échantillons d'urine dont il est question dans ce chapitre. Pour mener cette étude, le travail des chapitres précédents (est celui du chapitre 7) fourni des outils précieux pour réaliser le pré-traitement au mieux, cela permet de se concentrer seulement sur la reproductibilité des protocoles. Signalons enfin que le travail présenté ici et le fruit d'une collaboration avec la fondation pour la recherche biomédicale de l'académie d'Athènes qui a mis en place l'expérience biologique et s'est occupée de la génération des données, nous sommes intervenu pour l'analyse bio-informatique. Ce travail est consigné dans un article, [131], dont nous reprenons l'essentiel de notre contribution dans la suite.

Le chapitre contient trois sections. La section 6.1 présente la configuration

expérimentale de nos analyses, on y décrit d'une part comment sont obtenus les données (les spectres de masse) et d'autres part les analyses bio-informatiques réalisées. La section 6.2 présente les résultats obtenus avec ces analyses. Enfin, la section 6.3, tire les conclusions des expériences.

6.1 Configuration expérimentale

Comme nous venons de le mentionner, cette section présente la configuration expérimentale de nos analyses, on y décrit d'une part la manière dont sont obtenus les spectres de masse (section 6.1.1) et d'autres part les analyses bio-informatiques qui y sont effectuées (section 6.1.2).

6.1.1 Préparation des échantillons

L'analyse de reproductibilité que nous menons repose sur plusieurs répétitions de la même expérience MALDI-TOF sur le même échantillon biologique. L'échantillon examiné est un mélange d'urines qui proviennent de quatre individus en bonne santé. En comparaison aux échantillons de serum, l'analyse des profiles urinaires par spectrométrie de masse à fait l'objet de nettement moins d'études. Les urines présentent pourtant l'avantage d'être facile à collecter et elles contiennent des protéines utiles pour refléter l'état de santé général des patients. Elles sont donc un support idéal pour la recherche de nouveaux bio-marqueurs, et c'est pourquoi il est intéressant de se pencher sur l'analyses des protocoles de préparation de ces échantillons.

Plusieurs protocoles de préparation couramment employées ont été appliquées à l'échantillon d'urine en question avant de le soumettre à spectrométrie de masse. Ces protocoles ont fait l'objet d'une première sélection empirique basée sur la qualité des spectres de masse qu'il permettent d'obtenir à partir d'un échantillon d'urine (voir [131]). Parmi les protocoles sélectionnés, on distingue tous d'abord deux approches de dilution et deux approches d'ultrafiltration des échantillons. Les détails des protocoles biologiques sont données dans [131], mais pour faire simple retenons que soit les urines sont traitées à l'urée avant dilution (SMU), soit elles ne le sont pas (SM) ; et soit la méthode d'ultrafiltration laisse passer les protéines qui on un poids moléculaire supérieur à 10kDa (UF10kD), soit elle laisse passer celles qui ont un poids supérieur à 5kDa (UF5kD) [1]. En plus de cela, l'acquisition MALDI-MS des spectres est menée avec un des 3 types de matrice suivant : soit une matrice de type α-cyano-4-hydroxy-cinnamic (ACCA), soit une matrice de type 2,5-dihydroxybenzoic (DHB), soit enfin une matrice de type synapinic (SA) [2]. Au final, selon le pro-

1. La méthode d'ultrafiltration doit en théorie éliminer toutes les protéines dont le poids moléculaire est inférieur au seuil 5kDa ou 10kDa, mais dans la pratique toutes les protéines ne sont pas filtrées. Ainsi, les spectres de masse contiennent de nombreux pics avec des masses inférieures au seuils de filtration.

2. Remarquer que les protocoles de préparation ne font pas intervenir de surface d'affinité pour capturer les protéines, on préférera donc parler ici de technologie MALDI-TOF plutôt que SELDI-TOF malgré leur ressemblances.

tocole de préparation et la matrice utilisée, 8 combinaisons sont considérées dans nos analyses : ACCA/SM, ACCA/SMU, ACCA/UF10kD, ACCA/UF5kD, DHB/SM, DHB/UF10kD, DHB/UF5kD, SA/UF10kD. Pour chacune de ces combinaisons, le protocole de préparation est répété 10 fois sur dix aliquots distincts de l'échantillon d'urine, et chaque préparation est déposée sur 5 plaques pour la soumettre 5 fois à l'analyse MALDI-TOF. On obtient donc 50 spectres de masse pour chacune des 8 configuration expérimentale, soit un total de 400 spectres de masse à analyser. En théorie, les 50 spectres de masses réalisés selon la même configuration expérimentale sont exactement identiques car ils résultent de la même analyse du même échantillon de départ. En pratique cependant, on observe des différences liées aux problèmes de reproductibilité des expériences, et notre objectif dans la suite est de quantifier ces différences et d'étudier les effets de la préparation des échantillons sur cette reproductibilité. Pour cela, on s'aidera du découplage en 5x10 spectres de masse, dans chaque configuration, afin de dissocier les variabilités liées à la préparation des échantillons des variabilités liées à l'acquisition des spectres de masse. En plus, nous disposons également, pour chaque configuration, de 5 spectres de masse obtenus à partir d'un échantillon "blanc" censée ne contenir aucun pic qui nous aiderons à établir un résultat de référence pour nos analyses. Nous allons voir dans la suite comment exploiter ce découplage et les données des spectres blancs lors de l'analyse des données, mais avant cela quelques mot sur l'acquisition des spectres.

L'acquisition des spectres de masse est effectuée sur un instrument Ultraflex I MALDI-TOF-TOF-MS (Bruker Daltonics). Les échantillons sont traités dans un ordre aléatoire pour répartir d'éventuelles effets liées à l'acquisition des spectres sur toutes les expériences. En plus de cela, une calibration externe de l'instrument est réalisée au moyen d'échantillons standards (dont on connaît la composition) afin d'ajuster la position des pics observées sur leur position théorique. La plage de valeur pour l'acquisition des spectres est 1000-15000 Da pour les échantillons préparés avec les matrices ACCA et DHB, et elle est de 4000-25000Da pour les échantillons préparées avec la matrice SA, car SA est connue pour faciliter l'ionisation des protéines qui ont un poids moléculaire supérieure. Les différences entre les plages d'acquisition nous poussera à traiter les échantillons de type "SA" séparément des autres lorsqu'il s'agira de comparer les protocoles entre eux. Concernant la limite inférieur de l'acquisition à 1000Da, elle permet d'éviter les problèmes liées aux petites molécules de matrice qui sont responsable de l'apparition d'une ligne de base importante en début de spectre. La section qui suit présente le traitement bio-informatique des spectres de masses qui sont obtenus lors de ces acquisitions.

6.1.2 Analyse bio-informatique des spectres de masse

L'analyse bio-informatique des spectres de masse commence par une détection des pics dans chacun d'eux. Nous réalisons ces détections à l'aide de l'algorithme de recherche des vallées présenté dans la section 4.2.2, et après avoir opéré un lissage du signal grâce à une moyenne mobile des intensités qui emploie une fenêtre glissante de 30 valeurs (dans la section 7.3.2, nous avons observé que ce

lissage tend à améliorer l'efficacité de la détection). L'estimation du niveau de bruit, nécessaire à la détection des pics, est obtenue en calculant la déviation standard entre le signal original et le signal lissé dans une fenêtre glissante de 1000 valeurs. Seul les pics avec un rapport signal/bruit supérieur à un seuil S/N donné sont considérés. Cette procédure permet de déterminer la position en m/z des pics, et elle est peu sensible à la présence d'une ligne de base dans les spectres de masse (voir section 4.2.2).

Concernant l'intensité des pics, nous avons choisi de considérer la différence d'intensité entre le sommet du pic et la ligne de base, si bien qu'un calcul de la ligne de base est tout de même nécessaire après que nous ayons déterminé la position des pics. Nous utilisons pour cela l'opérateur d'ouverture mentionné dans la section 3.2 avec une fenêtre glissante contenant 1/20ème du nombre total de valeurs disponible dans le spectre. Aussi, afin de pouvoir comparer les intensités des pics des différents spectres, nous les normalisons par le courant ionique total du spectre (la somme des intensité après lissage et élimination de la ligne de base). Au final, selon la valeur du seuil S/N, nous obtenons une liste de pics plus ou moins longue, pour chaque spectre de masse, qui indique la position des pics en m/z et leurs intensités normalisées. Notre travail dans la suite est d'analyser ces listes de pics et de les comparer entre elles.

La comparaison de plusieurs listes de pics nécessite que l'on soit en mesure d'identifier les éléments qu'elles ont en communs, et ceux qui diffèrent. Cela passe par un alignement des pics que nous réalisons avec notre algorithme de clustering hiérarchique MZCL (présenté dans la section 5.1.2), en tolérant une erreur maximum de 0.2% sur la position des pics dans chaque cluster qu'il construit. Aussi, il est utile de définir une mesure pour estimer la similarité entre deux ensembles d'éléments. Nous utiliserons la distance de Tanimoto qui à deux ensemble X et Y associe la valeur

$$D(X,Y) = \frac{|X| + |Y| - 2|X \cap Y|}{|X| + |Y| - |X \cap Y|}$$

égale à 0 lorsque les ensembles sont identiques, et à 1 lorsqu'ils sont disjoints.

6.2 Résultats

On distingue trois niveaux logiques dans l'analyse de nos résultats. Les trois niveaux sont la conséquence du découplage de nos expériences en 8x10x5 : pour rappel, chacun des 8 protocoles de préparation de l'échantillon est répété 10 fois et chaque préparation est analysée par spectrométrie de masse à 5 reprises. Le premier niveau consiste à étudier la reproductibilité des expériences pour les groupes de 5 spectres de masse qui forment un répliqua ; le deuxième niveau porte sur l'étude de la reproductibilité entre les 10 répliquas ; enfin, le troisième niveau s'intéresse à la comparaison des protocoles de préparation entre eux. Nous décrivons chacun de ces niveaux d'analyse dans l'ordre dans les sous-section qui suivent.

6.2.1 Analyse intra-répliqua

Nous commençons par nous intéresser à la reproductibilité des groupes de 5 spectres de masse qui forment un répliqua. Pour cela, on fixe un seuil de détection de pics S/N, on extrait la liste des pics que contient chacun des 5 spectres, et on opère un alignement des listes afin de déterminer les pics que les spectres de masse ont en commun. Si la technologie est parfaitement reproductible, on doit détecter les mêmes pics dans chacun des 5 spectres de masse et obtenir après alignement une matrice d'expression avec 5 lignes identiques et sans valeurs manquantes. Si en revanche la technologie n'est pas parfaite, certains pics présents dans un des spectres de masse seront absents d'autres spectres de masse, et cela conduit à l'apparition de valeurs manquantes dans la matrice d'expression obtenue après alignement. En étudiant les valeurs manquantes de la matrice d'expression, on obtient donc des informations sur la reproductibilité des expériences. En particulier, on peut déterminer pour chaque pic détecté le nombre de spectre de masse dans lequel il apparaît, et en déduire le pourcentage de pics qui apparaissent dans seulement 1 des 5 spectres, dans 2 des 5 spectres, dans 3/5 spectres, dans 4/5 spectres, ou dans la totalité des 5 spectres. Ces informations fournissent une mesure facilement compréhensible de la reproductibilité : une expérience est parfaitement reproductible si 100% des pics apparaissent dans la totalité des 5 spectres, et plus ce pourcentage se répartie dans les petites tranches, et plus la reproductibilité se dégrade. La morphologie de la matrice d'expression dépend également beaucoup du seuil S/N choisi pour la détection des pics : si S/N est petit, on risque de détecter des pics dans le bruit des spectres et de générer de nombreuses valeurs manquantes dans la matrice d'expression ; à l'inverse, si il est trop grand, on risque de se focaliser sur les pics les plus prononcés des spectres et qui ont plus de chance d'être présent dans tous les spectres. C'est pourquoi, dans notre étude nous n'avons pas fixé un seuil S/N à priori, mais nous avons préféré étudier les résultats pour des valeurs de S/N qui vont de 0.6 à 6 avec un pas de 0.2.

méthode	# de pics
ACCA/SM	50.3
ACCA/SMU	34.1
ACCA/UF10kD	66.1
ACCA/UF5kD	105.7
DHB/SM	34.9
DHB/UF10kD	63.7
DHB/UF5kD	94.2
SA/UF10kD	31.2

TABLE 6.1 – Nombre moyen de pics détecté par spectre de masse avec un seuil S/N=3.4. L'information est calculée en moyenne pour les 50 spectres de masse à disposition dans chaque protocole.

Deux exemples de résultats obtenus en mesurant les pourcentages des 5 tranches pour les différents seuils S/N sont illustrés figure 6.1 : le premier

exemple vient d'un répliqua de ACCA/UF5kD pris au hasard, le second vient d'un répliqua de ACCA/UF10kD également pris au hasard. En parallèle, la figure montre également ce que l'on obtient pour l'échantillon "blanc" qui n'est censée produire aucun pic dans les spectres de masse. Des deux protocoles de préparation, ACCA/UF5kD est clairement le plus reproductible puisque pour une très large plage de S/N (de 3 à 6), plus de 40% des pics sont retrouvés systématiquement dans la totalité des 5 spectres de masse, et environ 50-60% des pics sont retrouvés dans au moins 4 des 5 spectres du répliqua. Pour la même plage de S/N, le protocole ACCA/UF10kD obtient respectivement 10% et 20%. En plus de ces résultats, il faut également considérer que le protocole ACCA/UF5kD permet de détecter plus de pics par spectre que ACCA/UF10kD ce qui le rend plus attractif : en effet, avec un seuil de S/N=3.4, on détecte en moyenne 105 pics par spectre avec le premier, alors que l'on en trouve 66 avec le second (table 6.1).

FIGURE 6.1 – Reproductibilité intra-répliqua pour un répliqua de ACCA/UF5kD et un de ACCA/UF10kD ainsi que les échantillons blancs correspondants. L'axe des abscisse indique le seuil S/N utilisé pour la détection des pics, et les 5 couleurs font références aux 5 tranches. L'axe des ordonnées permet de lire le pourcentage (cumulé) de pics présents dans 5/5 spectres (couleur la plus foncée), 4/5 spectres, ..., 1/5 spectres (couleur la plus claire). Par exemple, la 2ème tranche de ACCA/UF5kD/A pour un S/N=3.0 indique une valeur autour de 60% qui signifie que 60% des pics apparaissent dans au moins 2/5 spectres.

L'analyse des résultats pour les échantillons blancs indique que des pics sont

présents dans les spectres de masse issu de cet échantillon. Pour les seuils les plus faibles de la détection de pics, on retrouve même certains des pics dans tous les spectres de masse, néanmoins au dessus du seuil S/N=3.4, la reproductibilité décroît significativement (voir également table 6.2). Deux explications à cela, d'abord la présence de contaminant dans l'échantillon blanc qui fait apparaître des pics dans tous les spectres de masse, ensuite, le très grand nombre de pics détectés dans le bruit avec des seuils S/N faibles qui augmente la probabilité de trouver des pics à la même position dans tous les spectres. Signalons au passage que pour ACCA/UF5kD, nous ne disposons que de 4 spectres blancs et pas 5, c'est pourquoi la figure 6.1 ne comporte que 4 tranches. Malgré tout, on peut constater que les deux profils associés aux échantillons blancs sont assez proches l'un de l'autre bien qu'ils soient obtenus avec des protocoles différents.

méthode	# de pics
ACCA/SM	6
ACCA/SMU	NA
ACCA/UF10kD	1
ACCA/UF5kD	1*
DHB/SM	0
DHB/UF10kD	2
DHB/UF5kD	1
SA/UF10kD	2

TABLE 6.2 – Nombre de pics détecté avec un seuil S/N=3.4 dans au moins 3 des 5 spectres de masse réalisé à partir de l'échantillon "blanc". *On dispose seulement de 4 spectres de masse pour le protocole ACCA/UF5kD (et pas 5) si bien que le nombre fourni correspond au nombre de pics présents dans au moins 3 des 4 spectres.

Les résultats de la figure 6.1 ne montrent que deux exemples de répliquas, alors que les données contiennent 10 répliquas par protocole et 8 protocoles. Pour compléter nos résultats, il faut s'assurer que nos observations se généralisent à l'ensemble des données et il faut donner une vue d'ensemble des résultats. Pour cela, les pics que l'on retrouve dans la totalité des 5 spectres des répliquas fournissent une bonne indication de la reproductibilité des expériences. La table 6.3 propose de faire la moyenne de ces informations et reporte pour chaque protocole de préparation et chaque seuil de détection, le pourcentage moyen de pics (sur les 10 répétions) que l'on retrouve dans la totalité des 5 spectres des répliquas. Les valeurs de la table confirment nos observations et révèlent que le niveau de reproductibilité le plus élevé est celui de ACCA/UF5kD (avec environ 40% des pics présents dans les 5 spectres de masse des répliquas pour S/N>3.0), suivi de près par SA/UF10kD (environ 40% aussi) et ACCA/SM (35%). D'une manière générale, on voit que l'utilisation d'une matrice ACCA par rapport à une matrice DHB améliore la reproductibilité des expériences. Rappelons que ces analyses prennent en considération des proportions sans tenir compte du nombre de pics détectés par chaque méthode. Cette information est un complément important

qui indique la capacité du protocole à couvrir une partie plus ou moins grande du protéome de l'échantillon d'urine. La table 6.1 donne une idée du nombre de pics moyen observé dans les spectres de masse de chaque méthode lorsque l'on fixe le seuil de détection à S/N=3.4. On peut constater que la méthode d'ultrafiltration UF5kD à tendance à produire des spectres avec un plus grand nombre de pics que les autres approches. C'est intéressant si en plus le protocole est reproductible. En particulier cela rend ACCA/UF5kD très attractif car à la fois il détecte plus de pics que les autres en moyenne, et environ 40% de ces pics sont présents dans tous les spectres des répliquas.

S/N	ACCA SM	ACCA SMU	ACCA UF10kD	ACCA UF5kD	DHB SM	DHB UF10kD	DHB UF5kD	SA UF10kD
0.6	19.9%	20.2%	18.8%	20.4%	16.3%	14.9%	16.1%	19.9%
0.8	20.3%	19.8%	19.3%	22.5%	13.9%	13.0%	15.0%	19.3%
1.0	21.1%	19.6%	19.5%	25.4%	11.7%	10.9%	13.8%	19.4%
1.2	21.6%	18.4%	19.1%	26.9%	9.1%	9.0%	11.8%	17.5%
1.4	19.2%	16.2%	18.3%	27.5%	6.8%	6.9%	10.3%	15.8%
1.6	17.8%	14.0%	17.8%	28.1%	5.0%	6.2%	9.3%	14.8%
1.8	17.1%	13.4%	17.8%	29.5%	4.0%	5.8%	9.3%	15.5%
2.0	18.3%	13.7%	18.8%	32.1%	3.5%	5.7%	9.4%	16.9%
2.2	19.4%	14.8%	20.7%	35.5%	3.5%	5.6%	10.1%	18.2%
2.4	21.5%	15.9%	22.5%	38.6%	4.1%	5.6%	10.2%	21.1%
2.6	24.3%	17.3%	23.9%	41.0%	4.2%	5.7%	10.6%	26.1%
2.8	26.8%	18.5%	26.1%	43.0%	3.5%	5.9%	10.2%	30.4%
3.0	29.9%	20.6%	28.2%	42.9%	3.4%	6.0%	9.6%	34.8%
3.2	31.7%	21.1%	29.5%	42.9%	3.4%	5.9%	9.1%	38.8%
3.4	32.1%	21.6%	29.3%	42.6%	3.4%	5.5%	9.0%	42.6%
3.6	32.8%	22.5%	29.2%	41.3%	3.4%	5.1%	8.6%	42.9%
3.8	33.0%	23.4%	29.2%	41.6%	3.4%	4.8%	8.4%	41.9%
4.0	33.4%	22.2%	30.7%	41.4%	2.5%	4.3%	8.2%	40.1%
4.2	32.1%	21.8%	30.1%	41.3%	2.6%	4.0%	7.9%	38.2%
4.4	31.9%	23.1%	30.3%	39.8%	2.8%	3.4%	7.8%	37.9%
4.6	32.1%	25.0%	28.7%	39.1%	2.4%	3.0%	6.7%	37.5%
4.8	32.1%	23.9%	27.3%	38.8%	2.1%	3.0%	6.5%	36.5%
5.0	33.5%	22.8%	26.8%	38.6%	2.2%	2.9%	6.4%	36.0%
5.2	32.9%	23.8%	25.7%	39.0%	2.3%	2.8%	6.2%	34.4%
5.4	34.3%	23.2%	26.4%	38.0%	2.8%	2.6%	6.1%	34.1%
5.6	35.6%	22.5%	26.5%	36.5%	2.8%	2.6%	6.0%	36.8%
5.8	36.4%	22.5%	26.4%	36.1%	2.6%	2.6%	6.1%	36.4%
6.0	37.1%	20.6%	26.5%	35.0%	2.9%	2.7%	5.8%	38.5%

TABLE 6.3 – Proportion moyenne de pics (sur les 10 répliquas) détectés dans la totalité des 5 spectres.

6.2.2 Analyse inter-répliqua

Afin de caractériser la variabilité qui peut exister entre les 10 répliquas, nous commençons par extraire de chaque répliqua une liste de pics robuste aux variabilités interne du répliqua (que nous avons étudié section 6.2.1). Pour cela, nous ne retenons des répliquas que les pics qui apparaissent dans au moins 3 des 5 spectres de masse qui les composent. Les "pics robustes", identifiés dans les 10 répliquas d'un protocole, sont alignés par MZCL afin de procéder à une analyse similaire à celle que nous avons mené pour l'analyse intra-protocole. Le processus de détection et d'alignement des pics robustes est illustré figure 6.2 pour le protocole ACCA/UF5kD, et les résultats de l'analyse inter-protocole sont fournis figure 6.3.

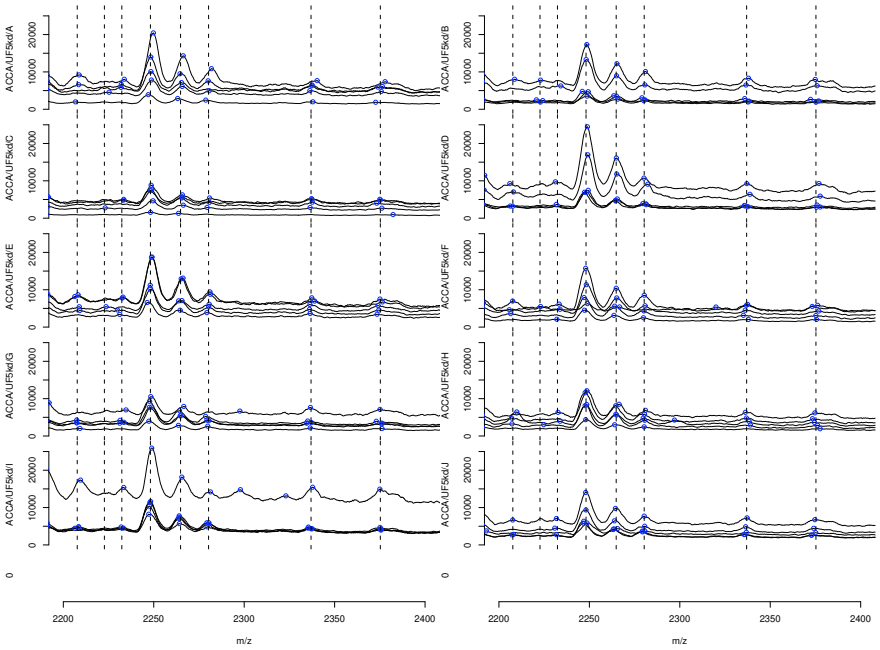


FIGURE 6.2 – Illustration de l'alignement des pics robustes entre les répliquas pour ACCA/UF5kD. On peut observer les 50 spectres de masse de ACCA/UF5kD dans la région 2200-2400 Da, réparties en 10 répétions de 5 spectres. Les points bleus matérialisent les pics détectés dans chaque spectre avec un seuil S/N=3.4. Les lignes verticales montrent le résultat de l'alignement des pics robustes sur les 10 répliquas.

La figure 6.2 montre les 50 spectres de masse de ACCA/UF5kD dans la région 2200-2400 Da, réparties en 10 répétions de 5 spectres. Les points bleues matérialisent les pics détectés dans chaque spectre de masse avec un seuil S/N=3.4. Les lignes verticales montrent le résultat de l'alignement des pics

robustes sur les 10 répliquas, chaque ligne montre la position m/z ou au moins un pic robuste a été trouvé dans 1 des 10 répliqua. On peut constater la présence de certains pics dans les spectres de masse autour de 2300Da, mais comme ils ne sont pas robustes (c'est à dire présents dans plus de 3 spectres dans les répliqua), ils ne sont associés à aucun alignement. On remarque également pour le répliqua ACCA/UF5kD/B la présence d'un pic robuste autour de 2225 Da qui explique la présence d'un alignement à cet emplacement malgré qu'il n'y ai aucun autre pic robuste dans les autres répliqua. L'analyse du nombre de pics robuste dans chaque alignement permet de tirer des conclusions sur la reproductibilité inter-répliqua.

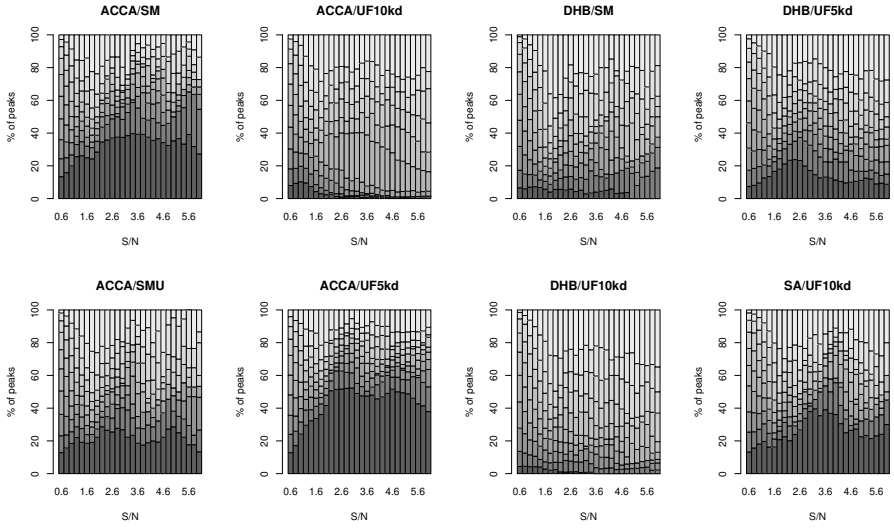


FIGURE 6.3 – Résultats de l'analyse inter-répliqua par protocole de préparation. Les 10 tranches de couleur qui se superposent reflètent la proportion de pics qui sont robustes dans 10/10 répliquas (couleur la plus foncée) jusqu'à 1/10 répliqua (couleur la plus claire), pour chaque seuil de détection S/N.

La figure 6.3 montrent les résultats de l'analyse de reproductibilité inter-répliqua pour chaque protocole. Les tracés indiquent, par des tranches de couleur, la proportion de pics qui sont robustes dans la totalité des 10 répliquas, 9/10 répliquas, ... seulement 1/10 répliqua. ACCA/UF5kD affiche une bonne reproductibilité pour une large plage de S/N (entre 2 et 6) : environ la moitié (50%, voir table 6.4) des pics couvrent la totalité des 10 répliquas, et le pourcentage augmente à 60% pour 9/10. Les performances de ACCA/SM et SA/UF10kD sont légèrement en dessous (entre 40% et 50%), et les bonnes performance de SA/UF10kD sont limités à des valeurs restreintes de S/N (3-4). Le détérioration de la reproductibilité de SA/UF10kD lorsque que S/N est supérieur à 4, signifie que les problèmes reproductibilité affectent également les pics les plus marqués

des spectres de masse avec ce protocole. Par contre les protocoles ACCA/UF5kD et ACCA/SM ne semblent pas souffrir de ce problème car leur reproductibilité reste constante pour S/N>4. Enfin, les protocoles faisant intervenir la matrice DHB ainsi que ACCA/SMU et ACCA/UF10kD montrent une reproductibilité bien inférieure.

S/N	ACCA SM	ACCA SMU	ACCA UF10kD	ACCA UF5kD	DHB SM	DHB UF10kD	DHB UF5kD	SA UF10kD
0.6	13.3%	12.8%	7.9%	12.8%	6.5%	4.3%	7.3%	13.2%
0.8	15.8%	15.6%	9.3%	17.1%	6.0%	4.5%	8.0%	15.6%
1.0	19.8%	18.4%	10.2%	24.1%	7.0%	4.2%	9.5%	18.0%
1.2	25.6%	22.0%	9.5%	29.5%	7.2%	4.2%	12.6%	19.6%
1.4	26.2%	19.1%	7.3%	32.3%	6.7%	4.1%	13.4%	16.2%
1.6	24.9%	18.5%	4.2%	33.6%	5.7%	2.8%	16.6%	16.2%
1.8	24.2%	19.0%	3.0%	36.1%	4.0%	2.0%	17.9%	20.7%
2.0	28.4%	22.1%	2.3%	41.6%	5.5%	2.4%	21.1%	22.5%
2.2	34.5%	28.6%	2.5%	50.6%	5.6%	1.0%	23.3%	20.1%
2.4	35.8%	26.5%	1.2%	51.4%	5.1%	0.9%	23.8%	23.9%
2.6	36.9%	25.4%	0.9%	51.6%	5.7%	1.0%	23.3%	26.6%
2.8	37.5%	27.7%	1.0%	52.1%	4.3%	1.2%	19.8%	33.8%
3.0	37.3%	26.8%	1.1%	52.3%	5.1%	0.7%	17.1%	37.9%
3.2	38.4%	22.2%	1.2%	48.9%	2.9%	0.8%	14.6%	39.6%
3.4	39.6%	23.4%	0.7%	47.5%	3.3%	0.8%	12.7%	35.4%
3.6	39.2%	18.6%	0.7%	47.4%	4%	0%	13.1%	39.0%
3.8	39.2%	17.5%	0.8%	48.0%	4.4%	0%	12.5%	36.8%
4.0	37.5%	18.9%	0.8%	45.6%	5%	0%	11.3%	36.1%
4.2	35.5%	20.0%	0.8%	46.7%	5.8%	0%	10.4%	30.5%
4.4	36.5%	19.3%	0.9%	49.0%	3.1%	0%	9.6%	25%
4.6	34.2%	22.2%	1.0%	51.5%	3.4%	0%	10%	26.4%
4.8	32.4%	27.2%	1.0%	50.0%	3.7%	0%	10.5%	27.2%
5.0	35.2%	28.5%	1.0%	49.4%	0%	0%	11.1%	22.5%
5.2	36.6%	25.0%	1.1%	48.7%	0%	0%	12.5%	23.3%
5.4	33.3%	22.2%	1.2%	46.0%	0%	0%	11.9%	21.4%
5.6	39.1%	17.6%	1.2%	42.6%	0%	0%	8.9%	23.0%
5.8	31.8%	17.6%	1.4%	41.4%	0%	0%	9.4%	24%
6.0	27.2%	13.3%	1.4%	37.8%	0%	0%	8.7%	30%

TABLE 6.4 – Proportion de pic détecté dans la totalité des 10 répétions.

La comparaison des résultats inter-repliqua avec les résultats intra-répliqua (section 6.2.1) est difficile car ils ne font pas intervenir le même nombre d'analyses. Il est effectivement statistiquement beaucoup plus fort d'affirmer qu'on retrouve un pic dans l'ensemble des 10 répliquas que d'affirmer qu'on le trouve dans l'ensemble des 5 spectres. Si on fait abstraction de cela et que l'on compare les valeurs des tables 6.4 et 6.3. On constate que certains protocoles ont une meilleure reproductibilité inter-répliqua que intra-repliqua (ACCA/SM, ACCA/UF5kD, DHB/UF5kD), pour d'autres c'est l'inverse (ACCA/UF10kD, DHB/UF10kD, SA/UF10kD), et il y en a pour lesquels c'est difficile à dire (ACCA/SMU,

DHB/SM). Dans le premier cas, la principale source de variation serait donc introduite lors de l'acquisition des spectres MS plutôt que lors de la préparation des échantillons. Citons à ce sujet [105] qui souligne qu'il s'agit effectivement d'une source de variation importante. Dans le second, c'est au contraire la préparation des échantillons la principale source de variation, et il est intéressant de remarquer que cela correspond à chaque fois à l'emploi de la méthode d'ultrafiltration UF10kD. On en conclu que cette méthode de filtration est inadaptée à la préparation des échantillons d'urine avant de les soumettres à spectrométrie de masse.

Nombre de pics

Comme nous l'avons déjà évoqué, en plus de la reproductibilité, il est également important de comparer les protocoles en terme de nombre de pics qu'ils génèrent afin de prendre en considération l'étendue du protéome qu'ils permettent de couvrir. La table 6.1 fournit déjà une indication, mais elle ne tient pas compte de la reproductibilité des pics à travers les expériences. Afin de compléter cette information, la figure 6.4 fournit pour chaque protocole le nombre de pics robustes que l'on retrouve dans 10/10 répliquas, 9/10, ..., 1/10 lorsque le seuil de détection des pics est fixé à S/N=3.4 (rappelons encore une fois qu'un pic robuste est un pic qui est présent dans plus de 3 spectres sur 5 à l'intérieur d'un répliqua). La même informations est également données sous forme de table dans la table 6.5. Remarquer que nous reprenons ici l'alignement des pics opéré lors de l'analyse inter-répliqua et illustré figure 6.2. Les nombres qui sont données correspondent aux lignes verticales de la figure 6.2.

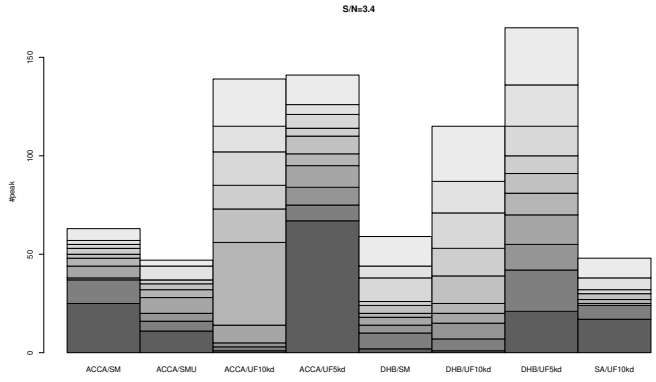


FIGURE 6.4 – Nombre de pics robustes présents dans 10/10 répliquas (couleur la plus foncé), jusqu'à 1/10 répliqua (couleur la plus claire) pour les différents protocoles.

ACCA/UF5kD est le protocole avec lequel on détecte le nombre de pics ro-

k	ACCA SM	ACCA SMU	ACCA UF10kD	ACCA UF5kD	DHB SM	DHB UF10kD	DHB UF5kD	SA UF10kD
1	63	47	139	141	59	115	165	48
2	57	44	115	126	44	87	136	38
3	55	37	102	121	38	71	115	32
4	53	37	85	114	26	53	100	30
5	50	35	73	110	24	39	91	30
6	48	32	56	101	20	25	81	27
7	44	28	14	95	18	20	70	25
8	38	20	5	84	14	15	55	24
9	37	16	3	75	10	7	42	24
10	25	11	1	67	2	1	21	17

TABLE 6.5 – Nombre de pics détecté par chaque méthode de préparation des échantillons lorsque le rapport signal/bruit de la détection de pic est fixé à S/N = 3.4. Chaque ligne donne le nombre de pics détecté dans au moins $k/10$ répétions.

buste le plus important : dans les 10 répliquas le nombre médian de pics détectés est 107 (avec +/- 8, de déviation standard). C'est également le protocole qui affiche le plus grand nombre de pics robuste entièrement reproductible 67/141 (47.5%) sur les 10 répliquas. DHB/UF5kD (90 +/- 24) et ACCA/UF10kD (80 +/- 36) permettent également de détecter un nombre de pics robuste important mais ils sont nettement moins reproductibles. Enfin, DHB/UF10kD (50 +/- 25), ACCA/SM (46 +/- 6) et SA/UF10kD (30 +/- 2) mènent à un nombre de pics inférieur, mais certains sont assez reproductibles.

Reproductibilité de l'intensité des pics

Jusqu'ici, nous avons seulement tenu compte de la position m/z des pics pour tirer des conclusions sur la reproductibilité des expériences en ignorant les intensités. Nos analyses ne sont effectivement fondée que sur la présence/absence des pics dans les spectres, et lors de l'alignement on ne tient compte que de leur m/z. C'est avantageux de se limiter seulement à l'information m/z des pics car, contrairement aux intensités, elle est robuste aux imprécisions liées à l'estimation de la ligne de base et à la normalisation des spectres qui pourrait être des sources de variabilité. Signalons néanmoins, que lorsque l'on fait varier le seuil S/N de la détection des pics, on introduit certaines contraintes sur l'intensité des pics, si bien que nous en tenons compte dans une certaine mesure. Pour compléter nos résultats, nous nous consacrons maintenant à l'étude de la variabilité de l'intensité des pics détectées dans les spectres de masse.

Pour cela, nous reprenons encore une fois l'alignement des pics réalisé lors de l'analyse inter-réplicat, avec un seuil de détection S/N=3.4, et illustré figure 6.2. De cet alignement on tire un certain nombre de position m/z ou apparaissent des pics robustes (les lignes verticales sur la figure 6.2), et on peut associer à chacune des positions jusqu'à 50 pics (les points bleues). On s'intéresse au coefficient de

variation (CV) de l'intensité des pics associés à chaque position. Le coefficient de variation mesure l'incertitude sur l'intensité des pics et il est égale au ratio de la déviation standard de l'intensité des pics par la moyenne des intensités. C'est une mesure couramment employé en spectrométrie de masse pour refléter que l'incertitude sur l'intensité d'un pic augmente linéairement avec son intensité.

La figure 6.5 donne les détails des CV sous forme graphique, et la table 6.6 résume les CV moyens. Pour les CV moyens, deux méthodes de calcul sont considéré : 1) la moyenne arithmétique classique ; et 2) la moyenne pondéré par le nombre de pics associé à chaque position. Cette deuxième alternative est introduite pour tenir compte que l'erreur sur la valeur d'un CV augmente lorsque l'on possède peu de valeurs pour le calculer.

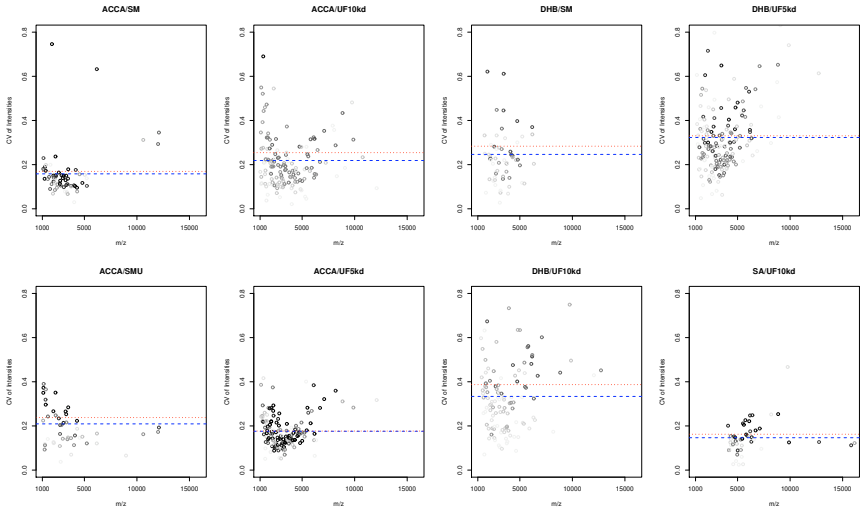


FIGURE 6.5 – CV de l'intensité des pics détectés à une position m/z donnée, pour une détection réalisée avec S/N=3.4. La couleur des points indique le nombre de pics associés cette position dans les 50 spectres (plus elle est foncée, plus il y a de pics associés à la position). Les lignes horizontales bleue et rouge indiquent respectivement la moyenne et la moyenne pondéré des CV.

Les résultats montrent que les méthodes qui ont eut les meilleures reproductibilités sont également celles qui ont les CV les plus faibles. En particulier, ACCA/UF5kD, ACCA/SM et SA/UF10kD ont un CV moyen inférieure à 20%. Sinon, tous les CV moyens sont compris entre 14% et 38% et correspondent aux valeurs couramment mentionnées dans les études SELDI et MALDI (voir section 2.3). La figure 6.5 suggère également qu'il n'y a pas de relation entre la position des pics en m/z et le CV de leurs intensités. Pour les trois protocoles mentionnés nous n'avons pas non plus établi de relation entre le CV de l'intensité des pics à une position donnée et l'intensité moyenne des pics à cette position (table 6.7, $p>0.05$ pour ces protocoles). C'est un résultat espéré car le

	moyenne	moyenne pondérée
ACCA/SM	15.8%	16.9%
ACCA/SMU	20.9%	23.7%
ACCA/UF10kD	21.8%	25.4%
ACCA/UF5kD	17.5%	17.6%
DHB/SM	24.6%	28.3%
DHB/UF10kD	33.3%	38.6%
DHB/UF5kD	32.2%	33.0%
SA/UF10kD	14.6%	16.2%

TABLE 6.6 – Coefficients de variations moyens pour l'intensité des pics.

CV tient déjà compte que l'incertitude augmente avec l'intensité des pics. Pour d'autres protocoles moins reproductibles, il peut y avoir une corrélation entre le CV et l'intensité des pics mais elle reste assez faible (en dessous de 41%), et elle peut s'expliquer par un nombre important de positions qui sont associées à peu de pics dans les spectres. Cela mène à des incertitudes sur le CV qui peut perturber le calcul de la corrélation par la suite.

méthode	correlation CV / Intensity
ACCA/SM	0.10(p=0.4444)
ACCA/SMU	0.41 (p=0.004)
ACCA/UF10kD	0.28 (p=0.001)
ACCA/UF5kD	-0.10 (p=0.242)
DHB/SM	0.41 (p=0.001)
DHB/UF10kD	0.24 (p=0.011)
DHB/UF5kD	0.06 (p=0.409)
SA/UF10kD	0.18 (p=0.212)

TABLE 6.7 – Coefficients de corrélation entre l'intensité moyenne d'un pic et le coefficients de variance de ses intensités pour les répliquas d'une méthode donnée. La valeur p entre parenthèses est le résultat d'un test statistique pour vérifier si le coefficient de corrélation est significativement différent de 0 (p ¡ 0.05 dans ce cas là).

6.2.3 Complémentarité des méthodes

Après avoir étudié la reproductibilité intra-répliqua et inter-répliqua de chaque protocole de préparation des échantillons, nous nous intéressons dans cette section à la complémentarité des protocoles. Notre but cette fois est de déterminer quels sont les protocoles qui couvrent des parties similaires et ceux qui couvrent des parties distinctes du protéome. Pour cela, il est nécessaire d'extraire une liste de pics qui caractérise chaque protocole, et de les aligner pour déterminer les pics qu'ils ont en communs et ceux qui diffères. Les listes de pics extraites pour

chaque protocole doivent être robustes aux variations intra-réplicat et inter-réplicat, sans quoi elle ne caractériserait pas vraiment le protocole. Pour cette raison, nous avons repris la procédure suivie lors de l'analyse inter-réplicat et qui nous à permis de déterminer la position des pics présents dans le protocole. Pour rappel, cette procédure consiste 1) en une détection des pics dans chaque spectre avec un seuil S/N=3.4 ; 2) en un premier alignement des pics intra-répliqua duquel on ne conserve que les pics robustes qui apparaissent dans au moins 3 des 5 spectres ; 3) un second alignement des pics robustes sur les 10 répliquas qui fourni la position des pics observées dans le protocole. Pour la comparaison des protocoles, on opère finalement un troisième alignement entre les pics observés dans les différents protocoles.

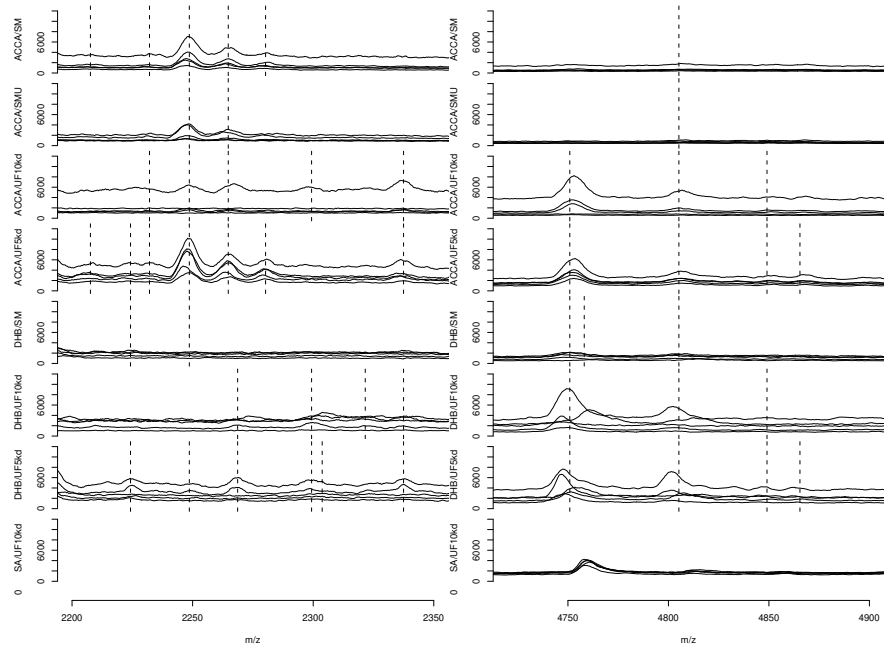


FIGURE 6.6 – Résultat de l'alignement des protocoles sur deux régions m/z différente. Chaque méthode est illustré par 5 spectres de masse pris au hasard parmi les 50. Les lignes verticales signalent la présence (ou l'absence) de pic caractéristique du protocole à la position ou elles apparaissent.

La figure 6.6 illustre notre propos en montrant, pour chaque protocole, 5 spectres de masse pris au hasard parmi les 50 réalisés selon ce protocole. Les lignes verticales montrent la positions des pics caractéristiques qui résultent du troisième alignement entre les protocoles. L'absence de ligne dans un des protocole indique qu'il n'y a pas de pic caractéristique à cette position. On constate également sur la figure qu'il n'y a pas de problèmes de décalage sérieux en m/z entre les différents protocoles, et que SA/UF10kD doit être traité séparément

en raison de sa plage d'acquisition différente (il est exclut de nos analyses). L'analyse du nombre de pics que chaque couple de protocole ont en commun permet de tirer des conclusions sur leur complémentarité. Ces informations sont consignées dans la table 6.8, et en utilisant la distance de Tanimoto pour mesurer la distance entre deux ensembles de pics, il est possible de construire, à partir de cette table, un arbre de clustering hiérarchique pour visualiser l'information (figure 6.7).

	# pics	ACCA SM	ACCA SMU	ACCA UF10kD	ACCA UF5kD	DHB SM	DHB UF10kD	DHB UF5kD
# pics		63	47	139	141	59	115	165
ACCA/SM	63	63	41	43	55	24	20	32
ACCA/SMU	47	41	47	36	37	19	13	24
ACCA/UF10kD	139	43	36	139	96	34	61	79
ACCA/UF5kD	141	55	37	96	141	43	61	94
DHB/SM	59	24	19	34	43	59	32	53
DHB/UF10kD	115	20	13	61	61	32	115	89
DHB/UF5kD	165	32	24	79	94	53	89	165

TABLE 6.8 – Tableau croisé du nombre de pics en commun pour chaque couple de protocole. La première ligne et la première colonne sont identiques à la diagonale et elles indiquent le nombre de pics détecté par cette protocole. Ces résultats correspondent à un seuil de détection S/N=3.4.

Le résultat de clustering hiérarchique avec *complete linkage*[3], [45], de la figure 6.7 regroupe, comme espéré, les protocoles similaires entre eux. L'arbre est très utile pour repérer les protocoles similaires et les protocoles complémentaires : par exemple ACCA/SM et DHB/UF sont très complémentaires car situés dans des branches bien distincts de l'arbre, alors que ACCA/UF10kD et ACCA/UF5kD sont très proches l'un de l'autre. De manière générale, au plus haut niveau de l'arbre, la distinction majeure se fait par rapport au protocole de préparation des échantillons : à gauche les protocoles "UF" (UF5kD et UF10kD), à droite les protocoles "SM" (SM et SMU). Aux deuxièmes niveaux, la distinction se fait en revanche par rapport à la matrice employée : d'une part ACCA, et d'autre par DHB. Enfin au troisième niveau, la distinction se fait par rapport aux paramètres des protocoles : UF5kD/UF10kD, ou SM/SMU.

6.3 Bilan de l'évaluation des protocoles

Les outils de pré-traitement des spectres de masse qui ont fait l'objet des chapitres précédant, nous ont permis de mener une étude sur la reproductibilité des protocoles de préparation des échantillons biologiques et l'acquisition MS des spectres de masse MALDI-TOF. Le nombre relativement important de spectres

3. Selon la méthode *complete linkage*, la distance entre deux clusters est égale à la distance entre les deux éléments les plus éloignés de chaque cluster

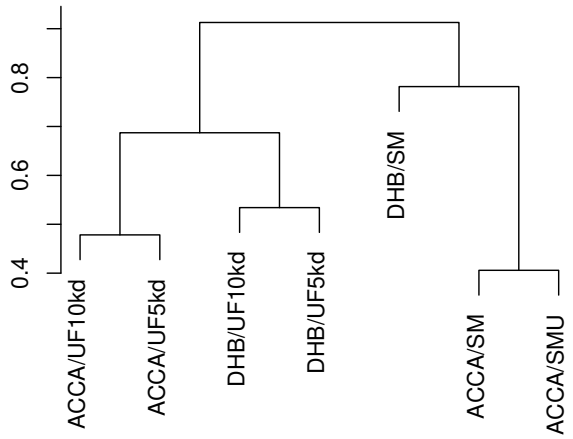


FIGURE 6.7 – Résultat du clustering hiérarchique des protocoles avec *complete linkage*

de masse à notre disposition (50 spectres par protocole) nous permet de mener une étude assez complète sur la position et l'intensité des pics détectés dans les spectres, et de tirer des conclusions fiables. Cela a rendu possible non seulement l'analyse de la reproductibilité de l'acquisition des spectres de masse (intra-répliqua), mais également de la reproductibilité des manipulations impliquées dans la préparation des échantillons biologiques (inter-répliqua), et aussi de la complémentarité des protocoles de préparation.

Ce que nous retenons de l'étude, c'est que des protocoles couramment employés pour la préparation des échantillons mènent à des reproductibilités très différentes. Il arrive même que certains protocoles, comme ACCA/UF10kD ou DHB/UF5kD, qui paraissent donner des spectres de masse de bonne qualité avec des pics bien marqués montrent en réalité une reproductibilité assez faible. Le protocole ACCA/UF5kD s'est en revanche distingué pour la bonne reproductibilité des expériences qu'il permet d'obtenir et pour sa large couverture du protéome. Avec SA/UF10kD, ils semblent bien adaptés à l'analyse des échantillons d'urine, l'un étant un bon complément de l'autre car ils couvrent des poids moléculaires différents.

Il résulte également de cette étude un certain nombre de bonnes pratiques

à respecter pour rendre plus robuste les données et augmenter les chances de succès des analyses bioinformatiques en vue de la recherche de bio-marqueurs. Avoir plusieurs spectres de masse pour le même échantillon (3 ou 4), permet en particulier d'identifier les pics robustes. Avoir plusieurs spectres de masse de référence réalisés à partir d'un échantillon blanc, permet de déterminer un seuil S/N pour la détection des pics de manière à ne détecter aucun pics dans le bruit. Ces approches sont susceptibles d'éliminer des pics dans les analyses, mais elles garantissent une certaine stabilité des données qui sont utilisées lors de l'apprentissage. Ce critère de stabilité est une notion importante que l'on utilise pour la première fois ici, et que nous utiliserons à nouveau dans la section 7.3.2 pour comparer des méthodes de pré-traitement, mais également dans la seconde partie sur l'apprentissage automatique.

Chapitre 7

Évaluation du pré-traitement

Dans les chapitres qui précédent, nous avons défini un pipeline de pré-traitement des spectres de masse en plusieurs étapes qui nous permet de construire la matrice d'expression. Bien que nous ayons essayé de limiter au maximum le nombre de paramètre qui contrôle ce pré-traitement, il reste certains paramètre à ajuster. Le plus important est sans doute le paramètre *vd* qui contrôle la sensibilité de la détection des pics car sa valeur affecte directement la dimension de la matrice d'expression. Se pose alors la question de savoir comment ajuster ce paramètre au mieux. Nous sommes également dans l'incertitude vis-à-vis des différentes options de pré-traitement qui nous sont offertes dans le pipeline : quel est la meilleure manière de remplir la matrice d'expression ? Quel est la robustesse de notre approche vis à vis du bruit ? La détection des pics fonctionn-elle mieux si le signal spectre est lissé ? Enfin, on s'interroge aussi sur les améliorations qu'apporte l'approche proposée par rapport aux approches déjà existantes.

Le présent chapitre essais de répondre à ces questions en expérimentant le pipeline de pré-traitement sur plusieurs jeux de données. Il est structuré en trois sections. La première (section 7.1) propose de se pencher sur les mesures qui permettent d'évaluer la qualité d'un pré-traitement. Ces mesures sont indispensables pour pouvoir déterminer si un pré-traitement est meilleure qu'un autre. Nous en avons identifié trois, et elles sont présentées dans cette section. Les deux sections qui suivent (section 7.2 et section 7.3) utilisent deux de ces mesures pour évaluer la qualité de notre pipeline de pré-tratiement. La section 7.2 s'intéresse plus particulièrement à une mesure de performance de classification, et essai de déterminer les meilleures paramètres et options de pré-traitement de notre pipeline. La section 7.3 s'intéresse davantage à une mesure de stabilité du pré-traitement dans le but de comparer la qualité de notre pipeline de pré-traitement à un pipeline de pré-traitement. Nous avons choisi de le comparer au pipeline de pré-traitement proposé Li et al.[35], qui porte le nom "PROcess",

car il offre plusieurs points de comparaison intéressant avec le notre. A signaler enfin que dans ce chapitre, nous nous référons à notre pipeline de pré-tratiement en utilisant les terminologies "pipeline *ms*" ou "pacquage *ms*" pour faciliter la notation.

7.1 Mesures pour l'évaluation du pré-traitement

L'évaluation de la qualité des algorithmes de pré-traitement et leur comparaison sont des problèmes difficiles à aborder dans la pratique [20]. Il faudrait idéalement disposer de spectres de masses réalisés sur des échantillons de composition connue (masse et concentration de tous leurs constituants) et vérifier dans quelle mesure la matrice de concentration obtenue à la fin de leur pré-traitement se rapproche de la matrice d'expression espérée en théorie. Il faudrait également utiliser des échantillons de composition complexes qui reflètent le contenu des échantillons traditionnellement analysés, comme par exemple de l'urine ou du sérum de patient, pour se placer dans les conditions réelles où les interactions entre constituants influent sur les spectres de masse produits (effet de suppression, compétition pour l'attachement à la surface, ...). De plus, comme la reproductibilité des expériences est un problème majeur des plates-formes SELDI-TOF, il faudrait également considérer des conditions expérimentales variées et vérifier la robustesse des algorithmes de pré-traitement à ces changements (acquisition des spectres à des moments différents dans le temps, en des lieux différents, sur des instruments différents, etc.). Malheureusement, il est difficile d'obtenir ce type de données, même si des efforts récents ont été fait en ce sens pour montrer la reproductibilité des plates-formes SELDI-TOF [40, 105, 131]. Pour évaluer et comparer la qualité de plusieurs pré-traitements il faut donc s'en remettre à des méthodes moins directes. Parmi celles-ci, on distingue trois approches qui sont chacune présentés dans une des sous-section qui suit.

7.1.1 Simulateur de spectrométre

La première approche propose de palier au manque de données expérimentales en générant artificiellement des spectres de masse. Coombes et al. [20] propose pour cela un simulateur qui imite les phénomènes physiques qui se produisent à l'intérieur d'un spectromètre de masse SELDI-TOF. Le simulateur prend en entrée les masses et les concentrations des molécules que l'on souhaite faire figurer dans l'échantillon biologique et génère des spectres de masse qui correspondent à cette mixture. Plusieurs paramètres de l'instrument sont ajustables par l'utilisateur comme la fréquence d'échantillonage des acquisitions, la longueur du tube où volent les ions, les tensions d'alimentations, et le temps de latence entre le shoot laser et l'application du champs électrique. La seule variable stochastique dans cette simulation est la vitesse initiale des ions (après le shoot laser) qui influe sur la résolution des spectres produits et accessoirement sur leur bruit. Ce simulateur a été justement développé dans l'optique de comparer des algorithmes de pré-traitement, et les auteurs l'utilisent dans un autre

article ([79]) pour évaluer leur pipeline de pré-traitement. Citons également le travail de Noy et al. [84], qui a également implémenté un générateur de spectre de masse, mais celui-ci n'est pas basée sur la simulation des phénomènes physique qui se produisent dans le spectromètre de masse.

Cependant, le problème des simulateurs est qu'ils ne prennent en compte qu'une partie des phénomènes physiques qui se produisent en réalité. Ainsi, par exemple, les spectres produits par le simulateur de Coombes et al. [20] n'ont pas de ligne de base ; leur bruit est concentré aux endroits ou se trouve les pics et est quasi inexistant dans les zones sans pics ; enfin, il ne prend pas non plus en compte les interactions entre molécules (effet de suppression, compétitions pour l'attachement à la matrice), etc. En conséquence, si l'on développe des algorithmes de pré-traitement en fonction des performances qu'ils affichent pour les spectres de masse générés par le simulateur, on court le risque que ces algorithmes soient trop spécifiques au simulateur et ne se généralisent pas à la réalité.

7.1.2 Stabilité du pré-traitement

La deuxième possibilité pour évaluer les algorithmes de pré-traitement repose sur des données expérimentales. Ces données consistent en plusieurs spectres de masse réalisés sur le même échantillon de départ dans des conditions expérimentales identiques. L'information que ces spectres contiennent est donc théoriquement la même, et l'on peut vérifier si les pré-traitements réalisés un à un sur les spectres (ou par petits groupes) produisent des résultats identiques. Les avantages de ce procédé sont multiples. Tout d'abord, il n'est pas nécessaire de connaître par avance la composition de l'échantillon analysé pour pouvoir l'appliquer. De plus, l'échantillon peut être complexe et provenir d'une expérience de spectrométrie classique (comme du sérum de patients par exemple) ce qui permet de travailler avec des spectres produits dans des conditions réelles. Enfin, ce procédé permet d'évaluer la stabilité de l'ensemble du processus expérimental depuis l'acquisition des l'échantillons jusqu'au pré-traitement numérique des spectres.

Dans Zerefos et al. [131], nous avons utilisé cette méthode non pas pour évaluer des algorithmes de pré-traitement, mais pour comparer la robustesse de différents protocoles biologiques de préparation des échantillons : les spectres qui sont produits en suivant le même protocole de préparation permettent d'estimer sa stabilité ; puis les stabilités des différents protocoles sont comparées entre elles et permettent d'élire le protocole le plus stable. De la même manière, en faisant varier les algorithmes de pré-traitement utilisés pour traiter les données, nous pouvons élire l'algorithme de pré-traitement le plus stable : les spectres sont traités en utilisant différents algorithmes de pré-traitement, et celui qui identifie le plus de pics aux mêmes positions dans touts les spectres est élue comme le plus prometteur. Dans la section 7.3 qui suit, nous utiliserons ce procédé pour comparer notre pipeline de pré-traitement à celui de PROcess (un autre pipeline de pré-traitement plus répandu [35]).

Il faut cependant noter que la stabilité des algorithmes de pré-traitements n'est pas une preuve de leur bonne performance. En effet, il est par exemple fa-

cile de définir un algorithme de détection de pics très stable, mais qui ne trouve pas les pics des spectres. Il suffit que celui-ci soit programmé pour détecter des pics à des positions fixes, même si il n'y a rien à cet endroit. L'algorithme paraît ainsi très stable puisque il extrait toujours les mêmes positions dans les spectres, mais ces positions ne correspondent pas forcement à des pics dans les spectres. Toutefois, la stabilité est quand même une indication de la bonne qualité d'un algorithme de pré-traitement car les mêmes pics sont censées être présents dans tous les spectres qui proviennent d'une même source, et un algorithme qui détecterait des pics qui sont en réalité du bruit a tendance à être moins stable car la position des pics dans le bruit est aléatoire. Ce qu'il faut retenir de cela, c'est que la mesure de stabilité n'est pas suffisante à elle seule pour affirmer qu'un pré-tratiement est de meilleure qualité qu'un autre. Il faut en parallèle s'assurer que le pré-tratiement soit performants, c'est à dire qu'il réalise bien la tache pour laquelle il a été programmé.

Dans la second partie de ce document, consacré à l'apprentissage automatique, nous serons amené a discuter à nouveau des rapports entre stabilité et performance pour les algorithmes de sélection d'attributs. Le parallèle avec le pré-traitement, et en particulier avec la détection des pics, n'est pas très surprenant dans la mesure ou un algorithme de détection de pics peut être vu comme un modèle qui prédit la position des pics.

7.1.3 Performance de classification

Enfin, la troisième façon d'évaluer la qualité d'un pré-traitement utilise la performance de classification d'un algorithme d'apprentissage. Pour l'appliquer il faut disposer d'un jeu de données qui définit un problème de classification, comme par exemple le diagnostique de patients à partir de spectres de masse. La méthode consiste à réaliser différents pré-traitements des données et utiliser les matrices d'expression qu'ils produisent pour évaluer la performance d'un algorithme d'apprentissage. Le pré-traitement qui obtient les meilleures performances de classification est considéré comme le plus adapté pour les données, car c'est celui à partir duquel il est possible d'extraire l'information la plus pertinente avec l'algorithme d'apprentissage utilisé.

L'avantage de ce procédé est qu'il est très général : il ne se limite pas aux données de spectrométrie de masse, et peut s'appliquer à l'évaluation du pré-traitement d'autres données. Il est également attractif car il prend en considération la totalité du processus, jusqu'à l'algorithme d'apprentissage. Par contre, son inconvénient est qu'il risque de favoriser les algorithmes de pré-traitement qui sont spécifiques à l'algorithme d'apprentissage utilisé et aux données utilisées. Idéalement, il faut donc vérifier la supériorité du pré-traitement choisi sur plusieurs jeux de données et en faisant varier l'algorithme d'apprentissage.

Noter aussi qu'en spectrométrie de masse, l'information pertinente est censée se trouver dans les pics des spectres, et on peut s'attendre à ce que le pré-traitement qui fournit les performances de classification les plus élevées extrait les pics le mieux possible. Cependant, ceci suppose aussi que les spectres de masse sont générés en suivant un protocole expérimentale rigoureux. Les études

de Sorace et al. [111] et Baggerly et al. [5] ont effectivement mis en évidences des structures dans le bruit des spectres de masse qui permettent de distinguer des patientes atteintes d'un cancer des ovaires des autres patientes. Ces structures n'ont pas de sens biologique, et paraissent être la conséquence d'un changement de protocole expérimental lors de l'analyse des deux groupes de patientes.

Dans les sections 7.2 et 7.3 qui suivent, nous utilisons ce procédé pour deux objectifs. D'une part, pour ajuster optimalement les paramètres de notre pipeline de pré-traitement et déterminer la représentation des données la plus adaptée, et d'autre part pour comparer la qualité de notre pré-traitement à celle d'un autre pipeline couramment employé (PROcess).

7.2 Ajustement des paramètres de pré-traitement

L'un des problèmes qui se pose lors de l'analyse des données de spectrométrie de masse est de fixer les paramètres et les options de pré-traitement de manière à obtenir une matrice d'expression qui reflète le mieux possible la réalité de l'expression des protéines dans les échantillons analysés. Dans les chapitres qui précédent, pour fixer ces paramètres et comparer la qualité des différentes méthodes de pré-traitement, nous nous sommes essentiellement basé sur des mesures empiriques. Par exemple, dans la section 4.2.2, nous utilisons le nombre de pics détectés dans un spectre de masse blanc (qui en théorie n'en contient aucun) pour déterminer le seuil signal/bruit de l'algorithme de détection de pics ; autre exemple, dans la section 3.2, nous estimons le paramètre contrôlant le nombre de segments dans la ligne de base à partir de la forme de la courbe d'erreur entre le spectre original et la ligne de base. L'objet de cette section est de poursuivre ces études sur l'ajustement des paramètres et des options de pré-traitement en adoptant une approche différente, plus globale. Ce que l'on entend par là, c'est que l'on ne va pas chercher à évaluer individuellement la qualité de chaque étape de pré-traitement comme auparavant, mais on va plutôt chercher à mesurer la qualité de la matrice d'expression générée par l'ensemble du processus de pré-traitement. Pour cela, nous allons employer la mesure de qualité du pré-tratiement basée sur les performances de classification (évoqué dans la section 7.1.3). Avec cette mesure, nous pouvons étudier l'impact d'un paramètre de pré-traitement sur la qualité de l'ensemble du processus.

Pour réaliser cette étude, nous nous plaçons dans un contexte de diagnostique de patients, où l'on cherche à distinguer les patients des différents groupes à partir des données de la matrice d'expression générée par le pré-traitement. Comme les paramètres et les options de pré-traitement ont un impact direct sur le contenu de la matrice, ils affectent son pouvoir de discrimination ; c'est à dire notre faculté à distinguer les groupes de patients à partir des données qu'elle contient. Or, on s'attend à ce que le pouvoir de discrimination de la matrice d'expression soit d'autant meilleure que le pré-traitement est de bonne qualité. Effectivement, plus un pré-traitement est de bonne qualité, et plus il arrive à extraire l'information pertinente contenue dans les données (c'est à dire, dans notre cas, la concentration des protéines à partir des spectre de masse). Se pose

alors le problème d'estimer le pouvoir de discrimination de la matrice d'expression, et pour cela, nous proposons de mesurer les performances de classification que des algorithmes d'apprentissage obtiennent lorsque ils sont entraînés avec les données de la matrice d'expression. Plus la matrice d'expression à un pouvoir de discrimination fort, meilleures devrait être les performances des algorithmes d'apprentissage.

Ce procédé, où l'ensemble du processus de pré-traitement est considéré comme une boite noire, et où l'on évalue la qualité du pré-traitement avec les performances qu'obtiennent des algorithmes d'apprentissage est assez universelle. D'ailleurs, nous l'avons appliqué à plusieurs reprises pour comparer et ajuster les paramètres de plusieurs pipelines de pré-traitement assez différents : dans Prados et al. [89], nous utilisons ce procédé pour ajuster les paramètres de la détection de pics du logiciel Ciphergen ; dans Kalousis et al. [60], pour ajuster les paramètres d'algorithmes de pré-traitement à base d'ondelettes. Dans ce dernier, nous proposons même un moyen d'automatiser le choix des paramètres de pré-traitement en opérant des validations croisées internes dans le processus d'apprentissage, et en choisissant ceux qui maximisent les performances de classification – ce qui peut-être utile pour des utilisateurs non-expert. L'étude qui suit applique encore une fois ce procédé pour analyser le pipeline de pré-traitement *ms* que l'on propose dans les chapitres précédents. Les résultats présentés reprennent en grande partie ceux de l'article Prados et al. [90], et nous nous intéressons essentiellement aux trois aspects suivant :

1. Est-il mieux d'employer les aires des pics ou leurs intensités pour représenter les concentrations des constituants de l'échantillon ?
2. Quel stratégie faut-il utiliser pour remplir les valeurs manquantes de l'alignement ?
3. Quel est le seuil signal/bruit le plus adapté à la détection des pics ?

Mais avant d'aborder ces thèmes, passons à la description des données utilisées.

7.2.1 Données

Notre étude sur l'ajustement des paramètres de pré-traitement a porté sur trois jeux de données de spectrométrie de masse SELDI-TOF : prostate, ovarian et stroke. Chacun définit un problème de classification binaire pour le diagnostique de patients. Prostate et ovarian sont des jeux de données relatifs au diagnostique de patients atteints d'un cancer de la prostate et des ovaires ; stroke est relatif au diagnostique de patients qui ont contracté un accident vasculaire cérébral. La table 7.1 résume leurs caractéristiques principales [1].

Plusieurs choses sont à mentionner au sujet de ces données. Il faut tout d'abord signaler que les spectres de stroke n'ont subit aucun pré-traitement, mais il apparaît que les spectres de prostate et ovarian en ont subit un par les

1. Les jeux de données prostate, ovarian et stroke que nous décrivons ici sont identiques à ceux utilisés dans la deuxième partie de la thèse sous le nom de sous le nom de prostateRawGt2000, ovarianRawGt2000, et strokeRawGt2000, et qui sont décrit dans la section 10.4.1.

	#Control	#Malade	#Attribut > 2000Da	
			brut	$vd = 2.5$
prostate	226	96	10363	390(83%)
ovarian	91	162	10363	385(63%)
stroke	101	107	23797	172(73%)

TABLE 7.1 – Caractéristiques des trois jeux de données utilisés dans l'étude sur l'ajustement des paramètres de pré-traitement. Les colonnes '#Control' & '#Malade' indiquent le nombre d'instances dans chaque classe ; la colonne 'brut' est le nombre de valeurs d'échantillonage dans les spectres de masse qui ont un m/z > 2000Da(données brutes, sans pré-traitement) ; la colonne '$vd = 2.5$' est la dimension de la matrice d'expression qui résulte du pré-traitement lorsqu'on applique une détection de pics avec un seuil signal/bruit = 2.5, et le nombre entre parenthèse indique le pourcentage de valeurs manquantes dans cette matrice.

fournisseurs de données. Une inspection visuel de ces spectres laisse effectivement apparaître qu'ils ont été lissé, et que la ligne de base a été éliminée des spectres de prostate. Le fait que l'on ne connaisse pas les algorithmes employés pour ces traitements nous empêche de retrouver les données originales, et complique l'étude de ces pré-traitement. C'est pourquoi dans cette section, nous laissons temporairement de coté l'étude des paramètres qui contrôlent le calcul de la ligne de base et le calcul du niveau de bruit, et nous nous concentrons seulement sur les paramètres qui contrôle la détection des pics et la construction de la matrice d'expression. Nous reviendrons sur les aspects du lissage du signal dans la section suivante (section 7.3) où des données brutes sont disponibles. Pour l'heure nous avons tenté de gommer les différences entre les jeux de données en éliminant la ligne de base de chacun des spectres à l'aide de l'algorithme top-hat décrit dans la section 3.2 [2]. Les spectres de prostate, qui n'ont déjà plus de ligne de base, sont tout de même traités à nouveau pour tenter de simuler une élimination de ligne de base selon le même algorithme que les autres. Par contre les spectres de stroke ne sont pas lissé comme le sont ceux de prostate et ovarian afin de pouvoir comparer le comportement de l'algorithme de détection de pics dans des conditions de bruitage différentes.

Une autre information importante au sujet des données est que nous avons coupé les masses inférieures à 2000Da dans tous les spectres. Le problème avec ces parties de spectres est que certains y ont trouvé des motifs pour la discrimination des patients que d'autres considèrent comme du bruit. Or, comme dans cette section nous utilisons le pouvoir discriminatoire de la matrice d'expression comme mesure de la pertinence des pré-traitements, nous avons préféré éviter le risque de biaiser nos résultats avec ces données. La troisième colonne de la table 7.1 indique ainsi le nombre de valeurs d'échantillonage des spectres

2. Comme paramètre pour la taille de la fenêtre glissante de l'algorithme top-hat, nous avons utilisé 1/20ème du nombre de points dans les spectres de masse, en vérifiant visuellement la cohérence de ce paramètre.

qui ont une masse supérieure à 2000Da, et non pas le nombre initial de valeurs d'échantillonage. De plus, les données restantes, au dessus de 2000Da, sont étonnement assez bien calibré, si bien que l'on peut envisager de construire une matrice d'expression sans effectuer la détection et l'alignement de pics. Par exemple avec prostate, nous pouvons construire une matrice d'expression contenant 226 instances positives, 96 instances négatives et 10363 attributs simplement avec les intensités des spectres. Les exemples d'apprentissage ainsi obtenus serviront à établir un résultat de référence pour voir si la détection des pics et la manière dont est remplie la matrice d'expression améliore les performances de classification.

7.2.2 Représentation des expressions & stratégie de remplissage

Dans cette partie, nous cherchons à étudier l'effet des options de pré-traitement qui s'offrent à nous une fois les pics des spectres détectés. On souhaite notamment déterminer quel est la meilleure représentation des expressions, et la stratégie la plus adaptée au remplissage des valeurs manquantes de la matrice d'expression. Dans le premier cas, deux choix s'offrent à nous : soit l'on utilise l'aire d'un pic, soit sa hauteur pour représenter l'expression de la protéine sous-jacente. Dans le second cas, deux possibilités encore : soit nous remplissons les valeurs manquantes de la matrice d'expression avec des zéros pour signifier l'absence de pics et donc une concentration de protéine nulle, soit nous recherchons l'information manquée par la détection de pics dans le spectre de masse. Donc quatre combinaisons possibles. Pourtant, alors qu'il est très facile de remplir les données manquantes de la matrice d'expression avec des zéros, la tâche se complique lorsque l'on décide de récupérer les informations dans les spectres de masse car il s'agit de déterminer l'aire et la hauteur d'un pic que l'on n'a pas détecté dans le spectre de masse. Pour la hauteur, nous avons simplement récupéré l'intensité du spectre (qui n'a plus de ligne de base) à l'endroit où devait se trouver le pic ; par contre il est plus difficile de récupérer l'aire du pic manquant, si bien que nous avons ignorer ce cas. Au final, seul trois combinaisons sont donc considérées dans ce qui suit :

iz Représentation des concentrations par l'intensité des pics & remplissage des valeurs manquantes avec des zéros.

az Représentation des concentrations par l'aire des pics & remplissage des valeurs manquantes avec des zéros.

is Représentation des concentrations par l'intensité des pics & remplissage des valeurs manquantes avec l'intensité du spectre à l'endroit où devait se trouver le pic

Les trois sous-sections qui suivent décrivent l'expérience qui cherche à déterminer laquelle des trois combinaison est la meilleure. On commence par une sous-section pour décrire comment est construite la matrice d'expression que l'on utilise dans notre apprentissage. La sous-section qui suit décrit les algorithmes

d'apprentissage utilisés. Et enfin on présente nos résultats dans la troisième sous-section.

Génération des matrices d'expression

Les listes de pics avec lesquelles nous travaillons pour construire la matrice d'expression sont obtenues en fixant le seuil signal/bruit de la détection de pics à 2.5. Cette valeur a été choisie suite à l'analyse faite dans la section 4.2.2 portant sur un spectre blanc produits en parallèle des expériences menées pour stroke. Lors de cette analyse, nous avons établi que le seuil de détection des pics devait être d'au moins 2.0 pour éviter que l'on détecte des pics dans un spectre blanc. Pourtant, après inspection visuelle de plusieurs détections, il a été décidé de relever le niveau à 2.5. La raison à cela est que l'on continuait d'observer de nombreux pics dans des régions bruitées des spectres, alors que nous préférons ici nous concentrer sur des pics plus marqués pour mieux étudier les effets de la représentation des expressions et de la stratégie de remplissage sans être gêné par du bruit.

La taille des matrices d'expression que l'on obtient après alignement des pics détectés avec ce seuil (et élimination des pics qui apparaissent dans moins de 5% des expériences) est donnée dans la dernière colonne de la table 7.1. Le contenu des matrices change selon la représentation et la stratégie de remplissage choisie (iz,az,is), et il faut souligner l'impact important de la stratégie de remplissage sur leur contenu. Effectivement, la table 7.1 nous indique qu'entre 63% et 83% des valeurs de la matrice sont manquantes, ce qui signifie que la majorité des pics sont présents dans un petit nombre de spectres (en annexe A, vous trouverez également, pour chaque jeu de donnée, un histogramme qui reflète les valeurs manquantes de la matrice d'expression).

Apprentissage

Les matrices d'expression ainsi obtenues sont utilisées comme ensemble d'apprentissage pour estimer les performances de classification de trois algorithmes d'apprentissage par validation croisée à 10 couches (*10 fold cross validation*). Les trois algorithmes de classification choisis viennent de l'utilitaire WEKA (Witten et Frank, [127]), et ont été sélectionnés pour couvrir un éventail d'approche assez différentes. Il s'agit de :

J48 Un arbre de décision avec les paramètres par défaut de WEKA, c'est à dire un paramètre d'élagage -C 0.25, et au minimum deux instances par feuilles -M 2.

IBk Une classification par voisin les plus proches, avec distance euclidienne comme mesure de distance, et le nombre de voisin déterminé automatiquement par une validation croisée interne de type *leave-one-out*.

SMO Une machine à vecteur support (SVM) avec noyau linéaire et un paramètre de complexité C fixé à 1.0.

		is	*az*	*iz*	brut
stroke	IBk	36.5	35.0(=)	34.1(=)	46.1(=)
	J48	34.6	39.4(=)	35.5(=)	41.8(=)
	SMO	18.2	25.4(-)	24.5(=)	15.8(=)
prostate	IBk	17.0	22.0(=)	27.6(-)	18.9(=)
	J48	20.4	28.5(-)	27.0(-)	22.9(=)
	SMO	14.2	20.8(-)	20.1(-)	12.1(=)
ovarian	IBk	9.4	12.2(=)	9.8(=)	9.4(=)
	J48	5.9	9.4(=)	12.2(-)	13.4(-)
	SMO	0.7	3.9(-)	3.1(=)	0.7(=)

TABLE 7.2 – Erreur de classification et résultat des testes de significativité des trois algorithmes d'apprentissage (IBk, J48, SMO) pour les différentes options de remplissage de la matrice d'expression (is,az,iz). La colonne 'brut' indique les performances sans pré-traitement, pour les autres, le seuil signal/bruit de la détection des pics est fixé à 2.5. Le symbole entre parenthèse indique le résultat du teste de significativité de Mc Nemars' contre la colonne *is*, le signe '-' indique un résultat significativement inférieure, alors qu'un '=' indique que le résultats n'est pas significativement différent (aucun résultat n'est significativement supérieur)

Résultats

La table 7.2 fournit les résultats de l'estimation des performances d'apprentissage. Elle donne l'erreur de classification qu'obtiennent les différents algorithmes d'apprentissage avec les matrices d'expression décrites ci-dessus, et dont les caractéristiques sont résumées dans la table 7.1. La dernière colonne de la table 7.2 indique aussi l'estimation de l'erreur de classification sur l'ensemble d'apprentissage 'brut', où chaque exemple d'apprentissage consiste en l'intensité de chacune des valeurs d'échantillonage des spectres de masse avant détection des pics. Un test statistique de McNemar's, réalisé avec un seuil de 5%, permet en plus de déterminer si chaque erreur est significativement différente de l'erreur obtenue avec la combinaison *is*.

Si l'on compare d'abord les performances de la colonne *is* avec celles de la colonne *iz* il apparaît que la première à un avantage systématique. Effectivement, les tests de significativité montrent que l'erreur de classification obtenue avec une matrice d'expression générée selon la procédure *is* est toujours inférieure ou équivalente à celle obtenue avec une matrice d'expression générée selon la procédure *iz*. Cela est vrai non seulement pour tous les algorithmes d'apprentissage utilisés, mais aussi pour tous les jeux de données, avec un accent spécialement prononcé dans le cas 'prostate' où l'erreur des trois algorithmes de classification est significativement inférieure. Ce résultats s'explique d'une part par le grand nombre de valeurs manquantes dans les matrices d'expression, et tend à montrer que de remplir la matrice d'expression avec des intensités issus des spectres de masse retient plus d'information discriminante que si elle est

remplie avec des zéros. La raison qui pourrait expliquer ce phénomène est que certaines décisions pour classer les spectres de masse se jouent sur des pics dont le rapport signal/bruit est inférieur au seuil de 2.5 que nous avons fixé, ils se trouvent donc dans le bruit et nous n'arrivons pas à les détecter. Effectivement, si de tels pics n'existaient pas dans les spectres de masse, nous aurions les mêmes performances en remplissant les matrices d'expression avec des zéros. Une autre explication possible est la présence de motifs dans les données que nous utilisons pour remplir les valeurs manquantes et qui correspondent en réalité à du bruit, comme les motifs mis en évidence par Baggerly dans le bruit de ovarian, [5]. Enfin, une dernière explication pourrait être que le grand nombre de zéros dans la matrice d'expression perturbe les algorithmes d'apprentissage.

La conclusion que l'on peut tirer de cette expérience par rapport au pré-traitement, c'est que notre algorithme de détection de pics manque certains pics des spectres de masse qui sont importants, et pour compenser ses erreurs, il faut remplir la matrice d'expression avec des données issues des spectres de masse. Signalons également que les performances supérieures de *is* par rapport à *az*, qui remplace aussi les valeurs manquantes par des zéros, tend à renforcer la même conclusion.

De manière similaire, si l'on compare les colonnes *az* et *iz* avec un test de significativité (résultat non fourni dans la table 7.2), on s'aperçoit qu'aucune erreur n'est significativement différente, ce qui tend à prouver que l'on ne tire aucun avantage à représenter les expressions par l'aire des pics plutôt que par leur intensité. Pourtant, si l'on considère les difficultés liées à l'extraction de l'aire des pics, on préférera sans doute utiliser les intensités pour la matrice d'expression. Au final, ces expériences soutiennent donc que la représentation la plus adaptée pour la matrice d'expression est obtenue par *is*, c'est à dire en utilisant l'intensité des pics et en remplissant les valeurs manquantes avec des intensités issus des spectres de masse.

Enfin, si l'on compare les performances de *is* (la représentation que nous venons de déterminer comme la plus adaptée) aux performances du jeu de donnée sans pré-traitement (colonne 'brut'), on observe peu de différences significatives (sauf pour J48 sur ovarian, où la représentation *is* est meilleure). Il faut certes faire preuve de beaucoup de précautions dans l'analyse de ces résultats, car la différence de dimension entre les jeux de données pré-traités et les jeux de données brutes peut influencer l'efficacité de l'apprentissage (la dimension des données 'brut' est effectivement plus de vingt fois supérieure à la dimension des données *is*). Toutefois, le fait d'observer peu de différence avec l'ensemble des jeux de données et des algorithmes de classification tend à montrer qu'un pré-traitement réalisé selon *is* conserve la quasi totalité de l'information pertinente contenue dans les données brutes. Cette remarque est particulièrement pertinente par rapport à l'algorithme SMO car il est réputé moins sensible à la dimension des données, et on peut donc espéré de lui qu'il ne soit pas perturbé par la grande dimension des données brutes. En plus de cela, le pré-traitement *is* réduit considérablement la quantité de donnée et se focalisant sur l'information biologiquement valide (c'est à dire les pics des spectres de masse). Dans une optique d'extraction des biomarqueurs, ce pré-traitement des données est donc

tout à fait recommandé car il permet une première réduction de la dimension des données sans dégrader les performances de classification que l'on peut en tirer. Signalons au passage que nous sommes arrivé à une conclusion similaire dans Kalousis et al. [60] avec un pré-traitement différent à base d'ondelettes.

7.2.3 Ajustement du seuil signal/bruit pour la détection de pics

Dans la section précédente, nous avons choisi une représentation des expressions et une stratégie de remplissage pour la génération de la matrice d'expression (*is*). Pour cela un paramètre clefs du pré-traitement, le seuil signal/bruit qui contrôle la sensibilité de la détection de pics, a été fixé à 2.5 sur la base de résultats empiriques. Ce paramètre est certainement le plus important du processus de pré-traitement car il sert à décider quels sont les régions des spectres qu'il faut prendre en compte, et celles qu'il faut ignorer car il n'y a pas de pics. Plus le seuil signal/bruit est élevé, plus on sélectionne l'information en éliminant les pics les moins marqués des spectres. Au niveau de la matrice d'expression, cela se traduit par une diminution de sa dimension, et/ou une diminution de sa densité (le nombre de valeurs manquantes augmente). La figure 7.1 montre à ce sujet comment la dimension de la matrice d'expression diminue lorsque le seuil signal/bruit de la détection de pics augmente. Comme attendu, on constate une décroissance des courbes, avec une pente plus prononcée de la courbe stroke qui est certainement liée à l'absence de lissage des spectres de ce jeu de donnée.

Dans la suite de cette section, nous allons nous intéresser à l'influence du seuil signal/bruit sur le pouvoir de discrimination de la matrice d'expression afin de déterminer le seuil signal/bruit optimal. On s'attend effectivement à ce que le pouvoir de discrimination de la matrice d'expression et la valeur du seuil signal/bruit soient liées car : si le seuil est trop élevé, on réduit l'information seulement aux pics les plus marqués des spectres et on diminue nos chances d'observer des pics discriminatoires ; à l'inverse, si le seuil est trop bas, certes on possède plus d'informations mais on risque également de "masquer" l'information pertinente dans le bruit et de perturber les algorithmes d'apprentissage peu robuste au bruit. A ce stade, il est sans doute utile de rappeler qu'un des objectif principal du pré-traitement est de réduire la dimension des données en conservant le plus d'informations discriminantes possible. L'avantage que nous gagnons à cela c'est que l'on rend les modèles plus facile à interpréter. Si on ne fait pas une détection des pics, mais que l'on choisie de travailler avec des données brutes, on disperse une information importante sur plusieurs points qui forment les pics et on risque de rendre l'interprétation plus difficile. Au final, le seuil de détection idéal est donc supposé être celui qui maximise le pouvoir discriminatoire de la matrice d'expression.

Pour réaliser notre étude, nous décidons donc de générer plusieurs matrices d'expression en faisant varier le seuil signal/bruit de la détection de pics (en suivant le protocole *is*). Nous estimons ensuite le pouvoir discriminatoire des matrices à l'aide de la performance de l'algorithme SMO, que nous choisissons d'une part car les expériences menées précédemment montre clairement qu'il est

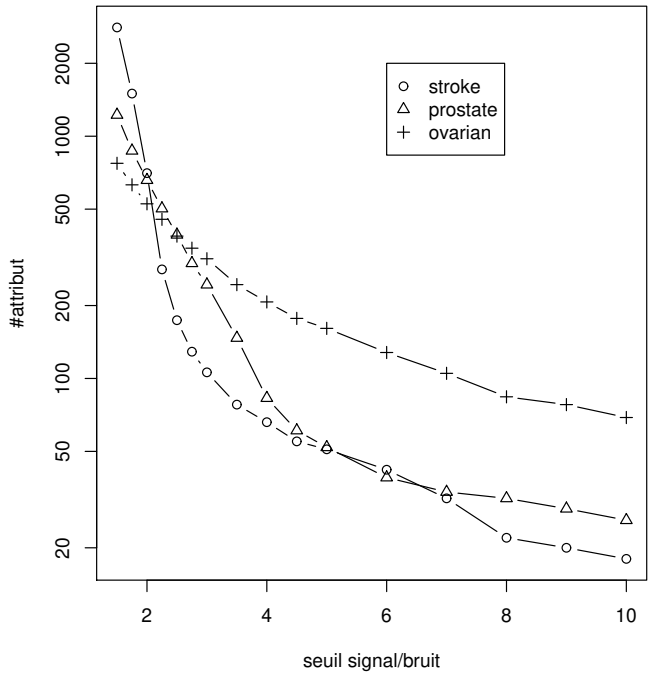


FIGURE 7.1 – Évolution de la dimension de la matrice d'expression lorsque le seuil signal/bruit de la détection de pics varie. Attention, l'axe des ordonnées est en échelle logarithmique.

le plus efficace pour trouver les motifs cachées dans les matrices d'expression (voir résultats de la table 7.2), et d'autre part car il est connu pour être robuste au données fortement bruitées que l'on peut avoir dans les données brutes ou lorsque avec une détection de pics sensible. Les performances de SMO sont étudiées en termes d'erreur de classification comme auparavant, mais aussi en termes de sensibilité [3]/spécificité [4] pour prendre en compte le défaut de balancement [5] des classes dans prostate et ovarian qui pourrait se traduire par une petite erreur de classification avec pourtant une mauvaise sensibilité ou spécificité.

L'ensemble des résultats sont présentés sur la figure 7.2. Elle montre l'évolution de l'erreur de classification, de la sensibilité et de la spécificité de SMO lorsque le seuil signal/bruit de la détection de pics varie. Comme attendu, les résultats se détériorent lorsque le seuil est élevé (au-delà de 4) car des pics informatifs disparaissent alors de la matrice d'expression. Pour prostate, cette détérioration se traduit essentiellement par une chute de la sensibilité car SMO préfère prédire

3. sensibilité : pourcentage de patients malade correctement classé par le modèle
4. spécificité : pourcentage de patients control correctement classés par le modèle
5. défaut de balancement : les données n'ont pas un nombre égale d'exemples "control" et d'exemples "malade"

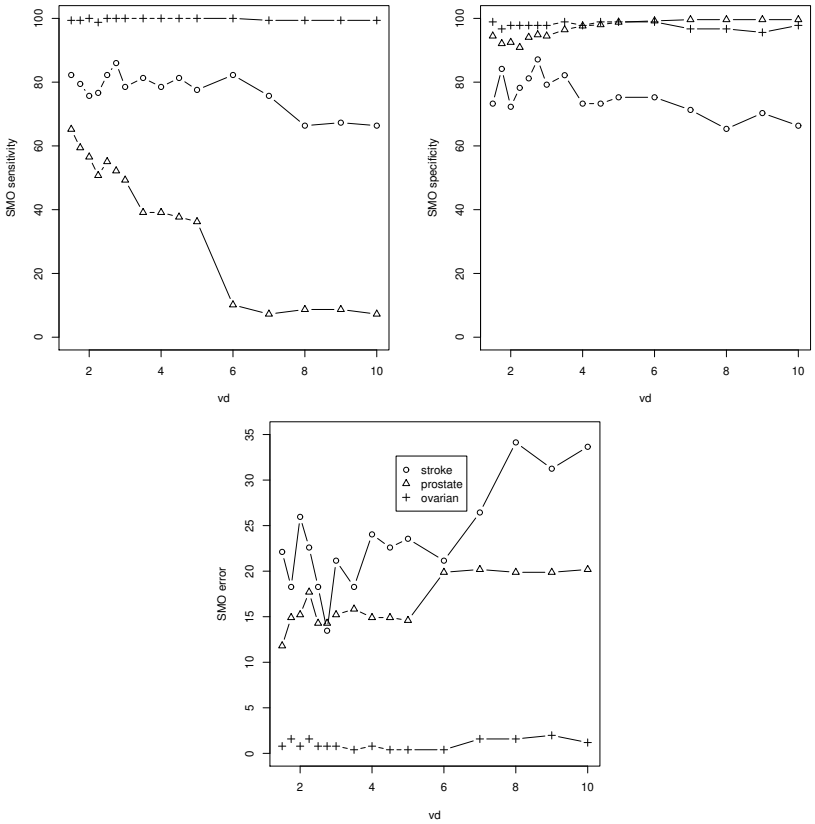


FIGURE 7.2 – Évolution de la sensibilité, de la spécificité, et de l'erreur de classification de SMO lorsque le paramètre qui contrôle le seuil signal/bruit de la détection de pics varie.

la classe majoritaire (les controls). Cette chute montre qu'il existe des pics informatifs qui ont un rapport signal/bruit "faible" (inférieur à 4). Par contre, les performances de classification sur ovarian restent très élevées même pour un seuil signal/bruit de 10. Ce résultat nous indique que pour ce jeu de donnée, l'information discriminante reste présente même si l'on conserve seulement les pics les plus prononcés des spectres (un seuil de 10 génère une matrice d'expression avec 67 attributs). Aussi, afin de vérifier que l'information pertinente se trouve effectivement dans l'intensité des pics les plus élevés et pas dans les intensités qui servent à remplir la matrice d'expression nous vérifions que l'erreur de classification de SMO sur ovarian était faible lorsque la matrice d'expression est rempli avec des zéros (protocole iz a la place de is). Une erreur relativement faible de 7.5% est observé qui confirme notre hypothèse. Avec ce résultat, nous étendons donc les observations que Baggerly et al. [5] font sur ovarian en mon-

trant qu'une bonne classification des données peut être obtenue seulement avec les pics les plus intenses des spectres de ovarian, alors que leur étude se concentre plutôt sur les motifs présents dans les bruits des spectres et les problèmes liées à la préparation des échantillons de ovarian.

Si l'on étudie maintenant les courbes lorsque le seuil signal/bruit est faible, la détérioration attendue est moins évidente, certainement à cause des bonnes facultés de SMO à extraire les motifs dans les données de grande dimension. Pourtant une région intéressante se dessine autour de la valeur seuil 2.75 : stroke obtient sa plus petite erreur de classification avec ce seuil, et l'erreur de prostate montre un minimum local. De plus, le seuil 2.75 est comparable au seuil que l'on a choisi de manière empirique (2.5), et correspond à des matrices d'expression de taille raisonnable qui contiennent entre 150 à 300 attributs (figure 7.1 et table 7.1). Au final, le seuil signal/bruit qui ressort de cette étude pour une détection approprié des pics se situe donc autour de 2.75.

7.2.4 Bilan de l'ajustement des paramètres de pré-traitement

Dans cette section, nous avons présenté des expériences visant à étudier et à ajuster au mieux les paramètres de pré-traitement et les différentes possibilités de représentation des données qui sont à notre disposition dans le pipeline de pré-traitement *ms*. Afin de pouvoir évaluer et comparer objectivement les différentes options qui s'offre à nous, nous avons défini un "framework" basé sur les performances de classification que l'on peut tirer des matrices d'expressions produites par le pré-traitement. Ce "framework" a permis de constater que les paramètres affectent parfois significativement les résultats des expériences d'apprentissage, et a permis de choisir objectivement la représentation et les paramètres les plus adaptées. Potentiellement, il est possible d'étudier l'impact de n'importe lequel des paramètres de pré-traitement, toutefois, nous nous sommes limité à l'étude de la représentation des données et au seuil signal/bruit qui contrôle la détection des pics. D'autres paramètres comme l'effet du lissage des spectres seront étudier plus en détail dans la section suivante.

Plus généralement, ce que nous montre ces travaux c'est un besoin de s'abstraire des paramètres de pré-traitement, soit en les fixant à priori, soit en les estimant au mieux. C'est d'ailleurs la raison pour laquelle dans les chapitres antérieures, le design des algorithmes est motivé par la réduction au minimum du nombre de paramètres à ajuster. Le framework que nous proposons dans cette section, nous a justement permis d'ajuster automatiquement ces paramètres en combinant le pré-traitement à une phase d'apprentissage plutôt que de le dissocier entièrement. Le problème c'est que le processus engendre un coût computationelle conséquent car le pré-traitement et l'apprentissage doivent être ré-exécuté à chaque changement de paramètre. Cette observation suggère toutefois que nous devons nous orienté vers des méthodes ou le pré-traitement est davantage intégré à l'algorithme d'apprentissage de manière à ce que l'ensemble soit réalisé en une seule opération [6].

6. Le travail de [101] est d'ailleurs intéressant à ce sujet concernant l'alignement des pics.

Le travail présenté dans la deuxième partie de cette thèse est notamment motivé par ces observations : nous y présentons une méthode d'apprentissage qui nécessite un pré-traitement des spectres et la construction d'une matrice d'expression, mais en revanche elle rend superflue l'étape de normalisation des spectres de masse. Pour cela, nous définissons une méthode d'apprentissage dont la prédiction est indépendante de la normalisation des spectres effectué, et qui en quelque sorte intègre ce pré-traitement en son sain. Mais avant de passer à ce sujet, la section qui suit montre une comparaison du pipeline de pré-traitement *ms* avec le pipeline de pré-traitement PROcess afin de vérifier si notre approche apporte un bénéfice par rapport à un autre pipeline de pré-traitement disponible.

7.3 Comparaison des pipelines de pré-traitement *ms* et PROcess

Cette section propose une comparaison de la qualité de notre pipeline de pré-traitement (pipeline *ms*) avec un autre pipeline de pré-traitement, plus répandu, du nom de PROcess [35]. L'étude présentée ici se base d'une part sur la stabilité des pics détectés par les deux approches (section 7.3.2), et d'autre part sur la comparaison de leur performances de classification respectives (section 7.3.3). Mais avant tout, nous décrivons plus en détails dans la section qui suit (section 7.3.1) le fonctionnement du pipeline PROcess.

7.3.1 PROcess

PROcess est un pacquage du langage R spécialisé dans le pré-traitement des spectres de masse SELDI-TOF. Il fait partie du projet BioConductor, un projet populaire offrant des logiciels libres pour le traitement des données biologiques sous la forme de pacquages pour le langage R [35]. Il est librement téléchargeable depuis l'adresse `http://bioconductor.org/packages/2.0/bioc/html/PROcess.html`. Dans cette partie nous employons PROcess (dans sa version 1.12.0) afin de comparer ses performances aux algorithmes de pré-traitement que nous proposons dans ce document.

Pour extraire les pics d'un spectre de masse, PROcess procède en deux étapes :

Élimination de la ligne de base : Il élimine d'abord la ligne de base du spectre de masse en découpant le signal en 200 morceaux de taille constante selon l'axe m/z (après transformation logarithmique), et y recherche le minimum local [7].

Détection des pics : PROcess opère ensuite une détection de pics. Pour cela, il lisse le signal à l'aide d'un filtre moyenneur à fenêtre glissante de taille *sm.span*, et y recherche les maxima locaux dans une fenêtre glissante (de taille 101 points, nous avons vérifié visuellement que cette valeurs était cohérente avec

7. dans nos expériences, la ligne de base est calculée avec la commande *bslnoff(spectrum,method="approx",breaks=200)*

nos données, c'est à dire qu'elle correspond plus ou moins à l'étendue d'un pic dans les spectres). Les maxima qui satisfont les trois contraintes suivantes sont considérés comme des pics : a) le rapport signal bruit est supérieur au seuil SoN ; b) l'intensité du signal au point maximum est supérieure à un seuil, que nous avons fixé à 0 ; c) l'aire sous la courbe autour du maximum (dans une région dont la taille est liée à la résolution du spectre) est supérieure à un seuil, que nous avons fixé à 0 pour éliminer cette contrainte.

Ces trois critères ont été définis par les concepteurs de l'algorithme pour éliminer les pics qui sont en réalité du bruit. Les valeurs seuils que nous avons employés dans les contraintes b) et c) les rendent cependant superflues (elles ne filtrent aucun maxima), mais ainsi la détection de pics que réalise PROcess avec ces paramètres se résume à une élimination de la ligne de base, suivie d'un lissage du signal, et de la recherche des maxima locaux qui ont un rapport signal/bruit minimal. Ces simplifications le rendent très similaire à de nombreux autres algorithmes que l'on retrouve dans la littérature tel que ceux de Yasui et al [128], Tibshirani et al. [116], Morris et al. [79], et rendent les résultats plus facile à interpréter.

Deux paramètres restent libres dans ce processus, il s'agit de *sm.span* et SoN qui contrôlent respectivement le degré de lissage du signal, et le rapport signal/bruit minimum pour qu'un maximum local soit considéré comme un pic. Ceux sont les deux paramètres qui influencent le plus le résultat de la détection de PROcess, et nous souhaitons étudier leur influence sur la stabilité des résultats produits. Signalons aussi que pour faciliter la comparaison entre PROcess et le pipeline de pré-traitement que nous proposons, nous avons unifié la manière d'estimer le niveau de bruit dans les deux approches. En conséquence, la méthode utilisée dans PROcess pour estimer le niveau de bruit a été remplacée par la méthode proposée dans la section 3.3. Au final donc PROcess diffère de notre approche par la manière dont il détecte les pics et parce qu'il nécessite une élimination de la ligne de base alors que *ms* travaille sur le signal original.

PROcess implémente également un algorithme d'alignement des pics. Cette algorithme passe par la construction d'un graphe ou chaque noeud est associé à un pic, et ou les arcs relient les pics qui ont des m/z similaires (la tolérance à l'erreur est ajusté par un paramètre). L'alignement est basée sur la recherche des cliques maximales du graphe, ce problème est NP-complet mais PROcess implémente une alternative probabiliste de type maximum likelihood pour approché la solution. Néanmoins, même si l'idée est intéressante, l'algorithme est mal adapté à l'alignement des pics car il n'interdit pas que deux pics d'un même spectre soit associé à une seule protéine, et se pose alors le problème de savoir comment combiner l'information associé à ces pics. De plus, si il n'y a pas de telles contraintes, le résultat de l'alignement devient très sensible au paramètre qui contrôle la tolérance à l'erreur. Effectivement, Dans tous les cas de figure, si une valeur de tolérance extrême est choisie, on peut aboutir à une matrice d'expression à une seule colonne. C'est notamment pour pour ces raisons que

dans la suite nous préférerons utiliser MZCL qui inclut cette contrainte.

7.3.2 Comparaison des stabilités de pré-traitement

Comme cela a été évoqué en introduction de ce chapitre, pour pouvoir comparer la stabilité des pré-traitements de PROcess à ceux de *ms*, il faut disposer de plusieurs spectres de masse en provenance du même échantillon, y appliquer les deux pré-traitements, et vérifier quel est celui qui identifie le plus grand nombre de pics aux mêmes positions. Nous en rediscuterons, mais à priori, la meilleure méthode est celle qui montre la plus grande stabilité car c'est celle qui identifie le moins de pics bruités des spectres.

Les données utilisées pour la comparaison des stabilités des pré-traitements sont identiques à celles que nous utilisons dans le chapitre 6 et dans Zerefos et al. [131]. Elles totalisent 400 spectres de masse MALDI-TOF réalisés sur un même échantillon d'urine en suivant 8 protocoles biologiques différents. Chaque protocole expérimental a donné lieu à 50 spectres de masse, parmi lesquels on distingue encore 10x5 spectres qui correspondent à 5 acquisitions de spectres sur 10 aliquots différents de l'échantillon de départ. Les différences entre les 8 protocoles s'obtiennent en faisant varier trois facteurs expérimentaux : il s'agit 1) de la matrice qui permet l'ionisation des molécules (3 types de matrices ont été testé ACCA, DHB, Sinapinic) ; 2) de la méthode de filtration des protéines qui permet de sélectionner les protéines selon leur poids (deux seuils ont été considéré : UF5kD et UF10kD) ; 3) de la méthode de dilution de l'échantillon d'urine (avec ou sans adjonction d'urée, SMU ou SM). Le lecteur intéressé peut se référer au chapitre 6 et à l'article [131] pour plus de détails sur les différences entre les protocoles. Pour l'étude qui suit, retenons simplement que les 50 spectres qui sont réalisés en suivant l'un des 8 protocole sont censés contenir les mêmes pics et ils vont nous permettre de comparer la stabilité des pipelines de pré-traitement. Il faut aussi savoir concernant ces données que les spectres sont d'assez bonne qualité : l'acquisition des données se limite à la plage 1000Da-15000Da avec un échantillonnage élevé des valeurs, la ligne de base des spectres est relativement peu élevé, et les spectres ne montrent pas de problème de saturation dans les régions de faible m/z si bien que l'on peu utiliser la totalité des données dans nos expériences.

Les deux sous-sections qui suivent décrivent la méthodologie et les résultats de notre étude de stabilité. Il y a deux volets dans ces résultats, un premier où nous fixons les paramètres de pré-traitement des pipelines *ms* et PROcess pour pouvoir comparer leur stabilité, et un second ou nous étudier l'influence des paramètres de pré-traitement sur la stabilité du pré-traitement.

Méthodologie

L'expérience que nous avons mise en oeuvre cherche à mesurer, pour chacun des 8 protocoles expérimentaux, si le pré-traitement tel que le pipeline *ms* le fait apporte un gain de reproductibilité/stabilité par rapport au pré-traitement tel que PROcess le fait. Ceci signifierai qu'il est capable de mieux s'abstraire des

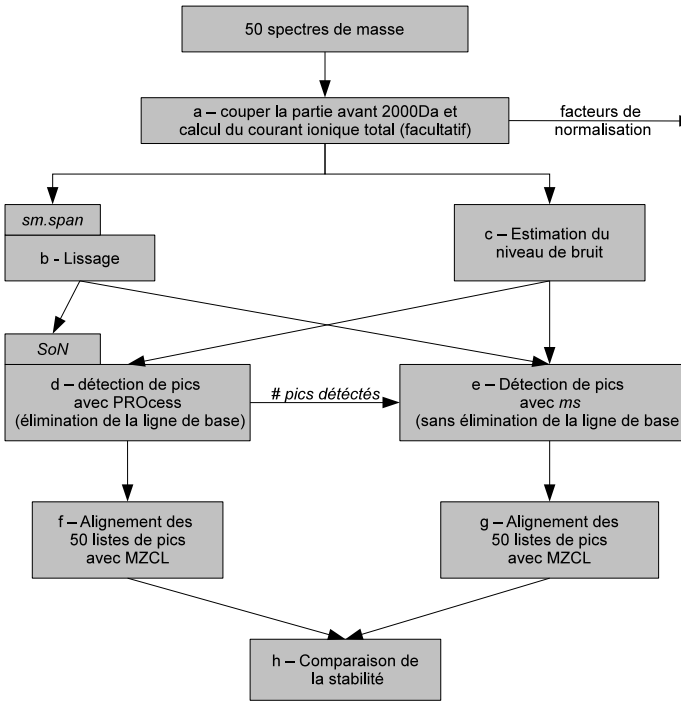


FIGURE 7.3 – Schéma du processus de comparaison des stabilités.

variabilités inhérentes à la préparation et l'acquisition des spectres de masse. Rappelons que pour cette comparaison, nous disposons au départ de 50 spectres de masse réalisés selon un même protocole expérimental et censés contenir des pics à des positions identiques. Le chemin qu'ils suivent pour nous permettre au final de comparer la stabilité de PROcess à celle de *ms*, est schématisé sur la figure 7.3. Nous donnons maintenant les détails de chaque étape de ce processus :

a, tronquer : Nous commençons dans ce processus par tronquer les spectres de masse, en éliminant la partie inférieure à 2000Da et par calculer le courant ionique total qui servira à la normalisation des spectres de masse. Certains experts considèrent effectivement que les pics située avant 2000Da ont biologiquement peu de sens car cette région est souvent bruitée par les molécules de la matrice [26]. Ces étapes sont cependant facultatives pour nos expériences sur la stabilité car les données sont d'excellentes qualité et que la normalisation des spectres de masse ne changera en rien nos résultats, par contre l'étape est mentionnée dans le diagramme car dans la section 7.3.3 nous utilisons le même schéma avec des données de qualité inférieure pour lesquelles il est plus prudent de prendre cette

précaution. Le courant ionique total est calculée comme l'aire sous la courbe après lissage et élimination de la ligne de base. Par contre ces opérations sont seulement réalisés temporairement pour calculer les facteurs de normalisation, mais elles n'affectent pas les spectres qui sortent de la boite a. En sortie de la boite a nous avons donc des spectres identiques aux spectres d'entré avec la partie avant 2000Da tronquée et des facteurs de normalisation.
b, lissage : Les spectres ainsi réduits sont lissés par le filtre moyenneur qu'utilise PROcess, et pour lequel on peut contrôler le degré de lissage à l'aide du paramètre *sm.span*. Deux valeurs de ce paramètre seront considérées dans les expériences : *sm.span*=0 qui correspond à aucun lissage de signal, et *sm.span*=31.
c, estimation du bruit : Parallèlement, le niveau de bruit dans les spectres est estimé par la méthode proposée dans la section 3.3 et pour laquelle nous avons observé des résultats robustes sur ces mêmes données – Les données utilisée ici ont la même origine que les données utilisées dans la section 3.3 –.
d, détection de pics avec PROcess : A partir des données lissées et du niveau de bruit calculé dans les phases **b** et **c**, une détection de pics est opérée avec PROcess. Comme nous l'avons dit dans la section 7.3.1, celle-ci se résume en une élimination de la ligne de base et une recherche des maxima locaux dont le rapport signal/bruit est supérieur au seuil *SoN*. Les fonctions du pacquage PROcess sont utilisées pour réaliser ces opérations.
e, détection de pics avec *ms* A partir des mêmes données que celles fournies en entrée du module **d**, nous opérons une détection de pics avec notre pacquage *ms*. Aussi, dans le but de faciliter la comparaison ultérieure des résultats fournis par ce dernier avec ceux rendus par PROcess, nous voulons que *ms* détecte le même nombre de pics que ce que PROcess en a détecté avec le seuil SoN. La détection a donc consisté en un calcul de la profondeur des vallées par l'algorithme 5 du chapitre 4, et la sélection des n points du spectre qui ont les vallées les plus profondes par rapport au bruit, n étant le nombre de pics détecté par PROcess pour le spectre en question. Noter la simplicité de ce processus qui n'a pas de paramètre, et s'applique directement sur les données sans élimination de la ligne de base.
f,g,h alignement et comparaison : Enfin, les trois dernières étapes se chargent d'aligner les 50 listes de pics détectées par chacune des deux méthodes, et d'évaluer leur stabilité. Dans les deux cas, l'alignement est fait avec MZCL (algorithme 7) que l'on paramètre avec une tolérance d'erreur de 0.5%. Nous préférons utiliser MZCL à la place de l'algorithme d'alignement de PROcess pour les raisons mentionnées dans la section 7.3.1 et aussi car cela nous permets de nous focaliser sur la comparaison des autres étapes de pré-traitement et enfin car cela facilite l'interprétation des résultats. L'estimation de la stabilité est faite en examinant la densité de la matrice d'expression, et aussi pour chaque pic identifié, le nombre de spectre dans lesquelles il apparaît. Effectivement, idéalement, les listes de pics détectés dans les 50 spectres sont identiques et l'alignement produit dans ce cas une matrice d'expression pleine (sans valeur manquante) ; au contraire, si des pics sont manqués par la détection ou si des pics sont détectés dans le bruit, ils n'apparaissent pas dans tous les spectres de masse et cela introduit des trous dans la matrice d'expression. La densité de

la matrice d'expression représente donc une bonne mesure de la stabilité d'un algorithme, d'autant que celle-ci est aussi égale au nombre moyen de valeurs pleines (non manquantes) dans chaque colonne de la matrice d'expression. Par exemple, une densité de 20% de la matrice d'expression signifie aussi que les colonnes de la matrice d'expression ont en moyenne 20% de valeurs pleines.

Résultats de la comparaison des stabilités

donnée	PROcess	ms
ACCA-SM	9.69 (19.38%)	13.48 (26.96%)
ACCA-SMU	7.72 (15.44%)	11.23 (22.46%)
ACCA-UF5kD	13.95 (27.90%)	18.19 (36.38%)
ACCA-UF10kD	7.46 (14.92%)	11.60 (23.20%)
DHB-SM	7.71 (15.42%)	10.82 (21.64%)
DHB-UF5kD	10.26 (20.52%)	13.47 (26.94%)
DHB-UF10kD	7.37 (14.74%)	11.21 (22.42%)
Sinapinic-UF10kD	7.95(15.90%)	10.99(21.98%)

TABLE 7.3 – Densité des matrice d'expression produite par chacun des pré-traitement. La table reprend les moyennes indiqués dans la figure 7.3.2 et en déduit la densité des matrices d'expression en divisant par 50, le nombre de spectres de masse dans chaque protocole. La densité déduite est indiquée entre parenthèse.

Les premiers résultats découlants de la méthodologie décrite ci-dessus sont obtenus en fixant informellement les paramètres SoN et Sm de manière à ce que la détection de pics de PROcess soit visuellement satisfaisante pour nos spectres de masse. Les valeurs choisies après inspection manuelle sont SoN=2.6, et Sm=31. La densité des matrices d'expression qui, comme nous l'avons vu, permet de quantifier la stabilité du processus d'acquisition des spectres est analysée en détail dans la figure 7.4 et la table 7.3. La figure met en opposition les résultats obtenus avec les deux algorithmes de pré-traitement (à gauche PROcess et à droite *ms*) pour chacun des huit jeux de données (un par ligne), ce qui nous permet de les comparer pour élire le plus stable. Chaque cadre permet de visualiser en abscisse la position m/z des pics détectés dans les 50 spectres, et indique en ordonnée le nombre de spectres dans lequel le pic est apparu – cette figure est construite à partir des résultats d'alignement –. En plus de cela, un histogramme horizontal situé sur la droite résume l'information en affichant le nombre de pics qui sont apparus dans 1 des spectres, 2 des spectres, ..., 50 des spectres. La ligne bleue indique la moyenne des valeurs et elle est étroitement liée à la densité de la matrice d'expression : il suffit de diviser cette valeur par le nombre de spectres dans chaque protocole (50) pour obtenir la densité de la matrice d'expression, c'est à dire le pourcentage de valeurs non-manquantes. Les densités des matrices d'expression sont reprisent dans la table 7.3.

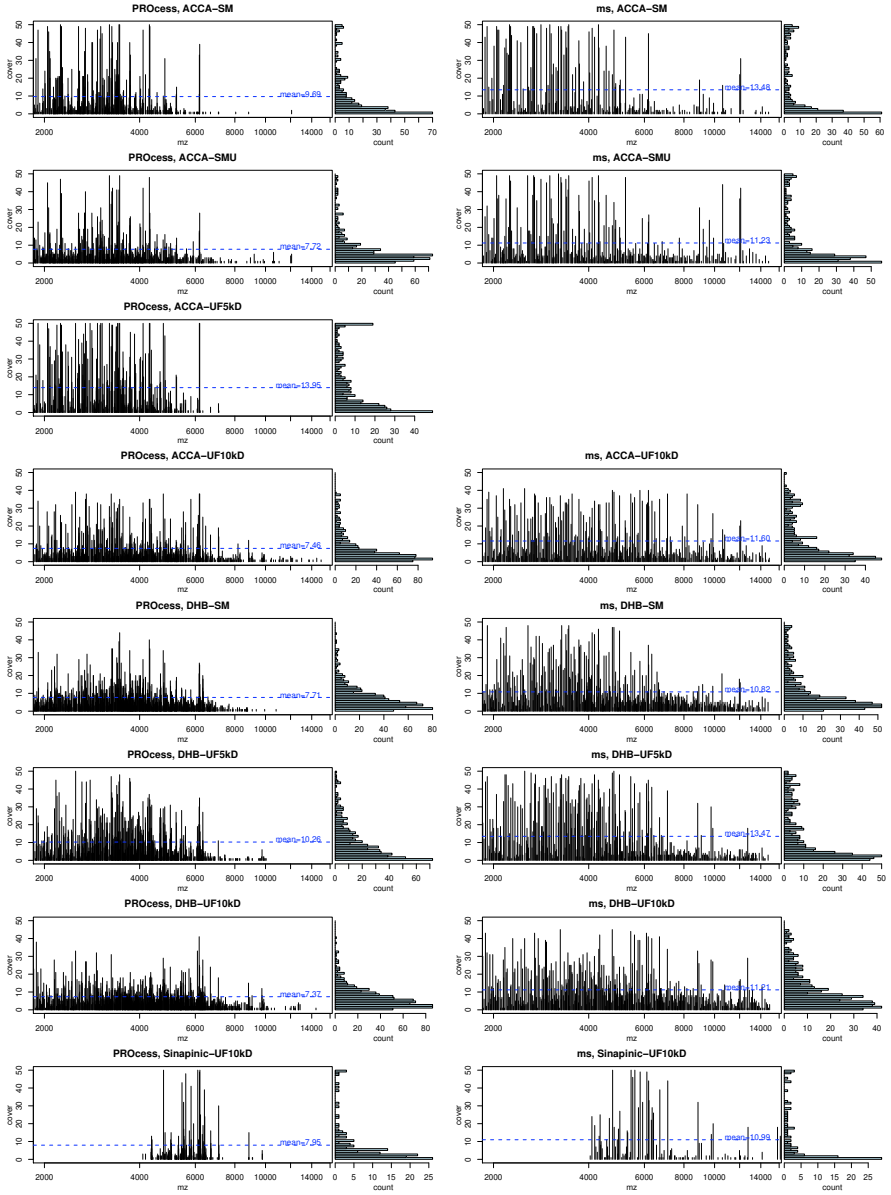


FIGURE 7.4 – Détails des résultats de stabilité pour tous les jeux de données en utilisant un seuil SoN=2.6, et un lissage Sm=31. Les résultats de PROcess sont situées dans la colonne de gauche, ce de *ms*, dans la colonne de droite. L'histogramme orienté verticalement (à gauche), indique le nombre de spectre qui contiennent un pic situé à un m/z donné. L'histogramme orienté horizontalement (à droite) est une projection des ces informations sur l'axe vertical et il indique le nombre de pics qui apparaissent exactement dans 1..50 spectres. La ligne bleue horizontale indique la moyenne des valeurs qui est une mesure équivalente à la densité de la matrice d'expression.

Plusieurs conclusions peuvent être tirées de l'analyse de la figure 7.4 et de la table 7.3. Tout d'abord, en ce qui concerne notre but initial de comparer les pré-traitements, la confrontation des deux colonnes montre clairement que *ms* a une stabilité plus élevé que PROcess. Dans les huit cas analysées, la densité des matrices d'expression produites par *ms* est systématiquement au dessus de celles de PROcess. Ceci apparaît clairement visuellement lorsque l'on compare les histogrammes horizontaux où l'on voit que les grandes valeurs sont davantage représentées dans *ms* que dans PROcess (ce qui signifie que les colonnes de la matrice d'expression ont tendance à avoir moins de valeurs manquantes), mais aussi quantitativement lorsque l'on compare les densités des matrices d'expression. PROcess, semble notamment avoir plus de mal que *ms* à identifier des pics stables dans les régions de grandes masses. Ce problème est certainement liée à la façon dont il détecte les pics : il recherche les maxima locaux dans des fenêtres de taille constante, or la largeur des pics augmentes dans les régions de fort m/z de sorte qu'une fenêtre de largeur adéquate pour les petites masses devient trop étriquée pour les grandes masses. Cela met bien en avant la difficulté d'ajuster un tel paramètre, et renforce l'intérêt d'un algorithme sans paramètre qui ne fait aucune hypothèse sur la largeur des pics, tel que celui que nous proposons dans *ms*.

La seconde chose que nous remarquons, sur les histogrammes horizontaux, c'est le grand nombre de pics qui apparaissent dans très peu de spectres de masse. Effectivement, la majorité des pics détectés apparaissent dans moins de 5 spectres de masse parmi les 50. D'ailleurs si l'on fait la comparaison des histogrammes obtenus avec une détection de pics aléatoire (figure 7.5), on constate que les distributions sont assez proches. Deux causes peuvent expliquer ce phénomène : soit il y a un problème de reproductibilité de la technologie utilisée pour acquérir les spectres qui fait que des pics apparaissent et disparaissent quand les expériences se répètent ; soit les algorithmes de pré-traitement détectent des pics qui sont en réalité du bruit. Pour vérifier quelle hypothèse est la plus probable, nous décidons d'étudier la relation entre la couverture d'un pic (i.e le nombre de spectres de masse dans lequel il apparaît) et son rapport signal/bruit. Si le problème vient du pré-traitement des données qui extrait du bruit, les pics que l'on observe dans peu de spectres doivent avoir un rapport signal/bruit faible. La figure 7.6 confronte ces deux informations pour les 50 spectres réalisés selon chacun des 8 protocoles, avec une détection de pic faite par le pipeline *ms*. On observe sur cette figure que plus les pics sont présents dans un grand nombre de spectres, et plus leur rapport signal/bruit est élevé ce qui joue donc plutôt en faveur de la deuxième hypothèse. Ce résultat renforce aussi l'idée que nous défendons dans la section 5.2 quand nous suggérons de supprimer de la matrice d'expression les colonnes qui comporte de nombreuses valeurs manquantes car elles correspondent certainement à du bruit.

Enfin, pour en terminer avec l'analyse de la figure 7.4, nous pouvons aussi comparer les protocoles expérimentaux entre eux (ligne par ligne) afin de déterminer le plus stable. Dans le chapitre 6 et dans Zerefos et al. [131], des expériences proches de celles que nous étudions maintenant nous avaient déjà permis de déterminer ACCA-UF5kD comme le protocole le plus stable. Les résultats de la

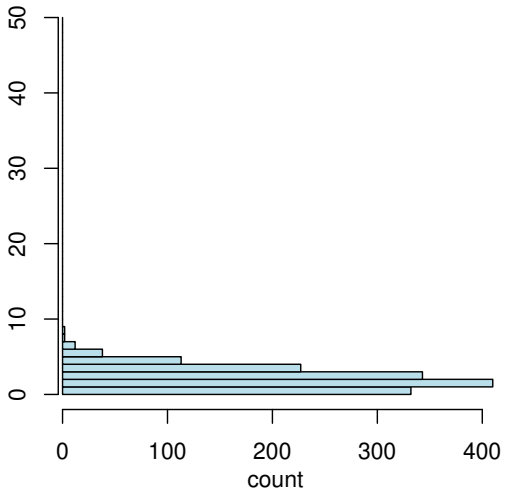


FIGURE 7.5 – Histogramme de couverture obtenu avec une détection de pics aléatoire pour laquelle 100 pics sont positionnés aléatoirement dans chacun des 50 spectres à 1500 postions possibles.

figure 7.4 sont en accord avec les précédents, et montre une nette démarcation de ACCA-UF5kD par rapport aux autres protocoles expérimentaux. Les spectres de masse qui sont produits en suivant ce protocole sont effectivement les seuls à contenir de nombreux pics que l'on retrouve dans la totalité des 50 spectres. L'autre intérêt qu'il y a à comparer les protocoles entre eux, est que cela permet d'identifier leurs ressemblances et leurs différences. C'est souvent important pour un biologiste d'avoir des éléments pour choisir un protocole expérimentale, par exemple pour compléter celui qu'ils utilisent actuellement ou pour valider ses résultats. Dans le chapitre 6, nous utilisons un algorithme de clustering hiérarchique couplé à un calcul de distance de Tanimoto[8] sur les ensembles de pics détectés par les protocoles pour mesurer les ressemblances entre les protocoles et les visualiser. De la même manière, la figure 7.7, montre le clustering hiérarchique obtenus à partir de la figure 7.4 en utilisant une méthode *complete linkage*[9] [45], et la distance de Tanimoto sur l'ensemble des pics trouvés dans plus de la moitié des spectres de masse (c'est a dire 25/50). Afin de déterminer si les pics des différents protocoles sont identiques ou différents, ils sont à nouveau alignés par MZCL que l'on paramètre avec une tolérance d'erreur de 0.5%.

8. La distance de Tanimoto est est mesure de distance entre ensemble, nous en reparlerons dans le chapitre 9.

9. Selon la méthode *complete linkage*, la distance entre deux clusters est égale à la distance entre les deux éléments les plus éloignés de chaque cluster

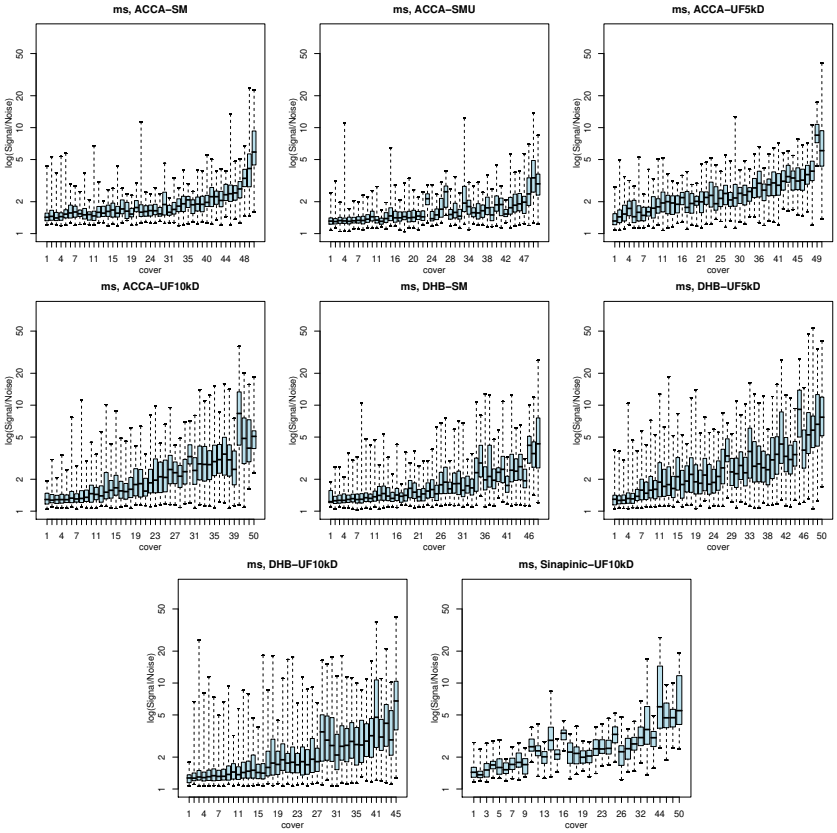


FIGURE 7.6 – Évolution du rapport signal/bruit des pics détectés avec *ms* en fonction du nombre de spectre dans lequel ils apparaissent. Pour chaque couverture, sont indiquées le signal/bruit médian, le 25ème et 75ème pourcentile, et le maximum et le minimum.

On remarque que le résultat de clustering obtenu est très similaire à celui que nous obtenons dans le chapitre 6 malgré des différences de pré-traitement entre les deux cas. Mais, nous ne nous attardons pas d'avantage sur la comparaison des protocoles expérimentaux, car ce sujet est abordé en détail dans le chapitre 6, et ici nous nous intéressons à la comparaison des méthodes de pré-traitements. Pour l'instant, nous nous sommes contenté de comparer les pré-traitements en fixant à priori les paramètres des pipelines. Cependant, certains paramètres ont un impact importants sur le résultats du pré-traitement. C'est en particulier le cas des paramètres *sm.span* et SoN qui contrôlent respectivement le lissage des signaux et le rapport signal/bruit minimum pour un pic. C'est pourquoi dans le paragraphe qui suit nous tachons de faire varier ces paramètres pour observer leur impact sur la stabilité des résultats.

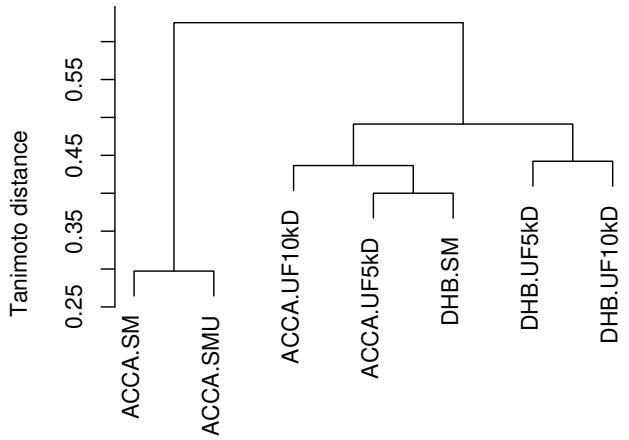


FIGURE 7.7 – Clustering hiérarchique des protocoles expérimentaux en fonction de la distance de Tanimoto sur l'ensemble des pics qu'ils trouvent dans plus de la moitié des spectres.

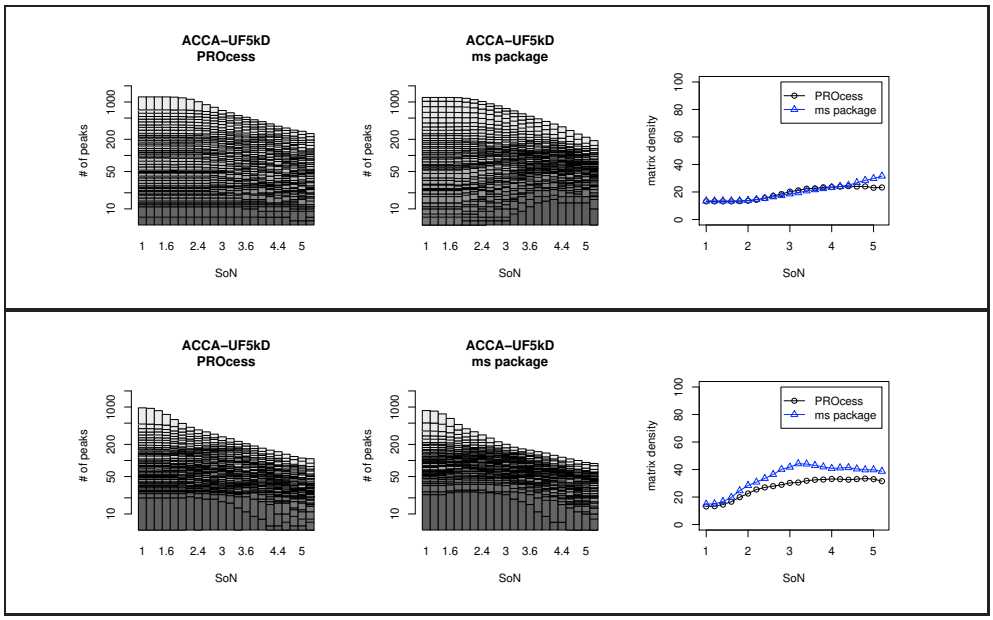


FIGURE 7.8 – Résultats de stabilité pour ACCA-UF5kD, avec en haut *sm.span=0*, et en bas *sm.span=31*.

Impact des paramètres de pré-traitement sur la stabilité

Pour étudier l'influence des paramètres *sm.span* et SoN sur la stabilité du pré-traitement, nous reprenons la même méthodologie employée ci-dessus et faisons varier les paramètres *sm.span* et SoN. Pour *sm.span*, les deux valeurs *sm.span=31* et *sm.span=0* sont considérées, elles correspondent respectivement à un lissage modéré des signaux et pas de lissage du tout. Quand à SoN, il varie entre 1 et 5.2 avec un pas de 0.2. Aussi, afin de simplifier notre propos, nous limitons la discussion aux deux jeux de données ACCA-UF5kD et DHB-UF5kD représentatifs de l'ensemble des résultats (qui peuvent par ailleurs être consultés dans l'annexe B). Les figures 7.8 et 7.9 montrent ce qui est obtenu pour ces données.

Les résultats sans lissage de signal (*sm.span=0*) sont situées en haut et peuvent être comparées aux résultats avec lissage (*sm.span=31*) d'en bas. Les graphiques reprennent l'information des histogrammes horizontaux de la figure 7.6 pour chaque combinaison spécifique de valeurs *sm.span* et SoN. Effectivement les courbes des figures 7.8 et 7.9 (à droite) montrent l'évolution de la densité de la matrice d'expression en fonction du paramètre SoN et sont étroitement liées aux moyennes des distributions indiquées sur la figure 7.6. D'un autre coté, les histogrammes des figures 7.8 et 7.9 sont composées de 50 couchent qui accumulent les informations contenus dans l'histogramme horizon-

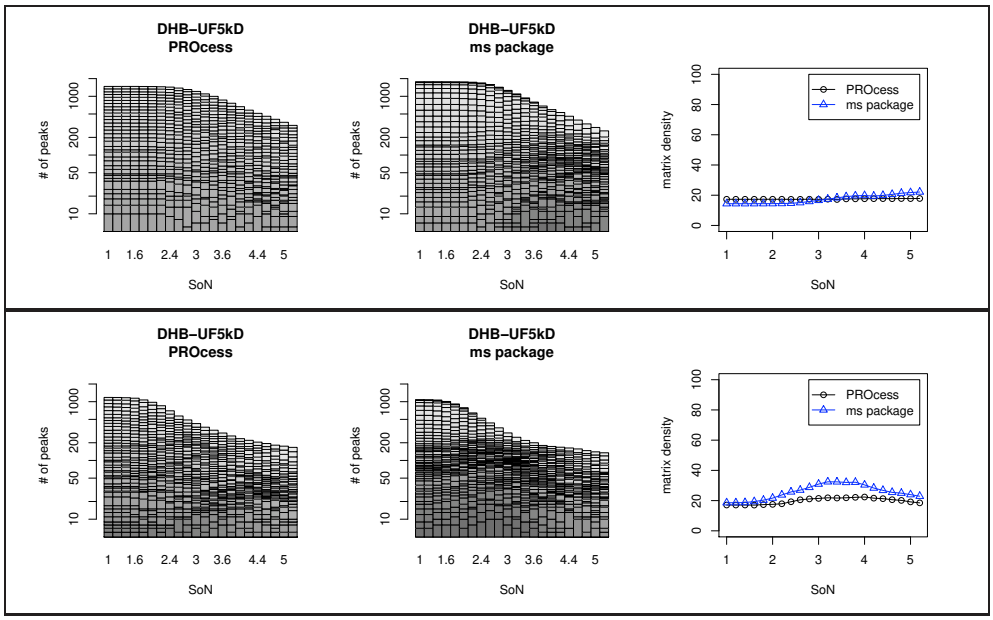


FIGURE 7.9 – Résultats de stabilité pour DHB-UF5kD, avec en haut *sm.span=0*, et en bas *sm.span=31*.

tal de la figure 7.6 pour une valeur SoN spécifique. La première couche indique ainsi le nombre de pics que l'on retrouve dans la totalité des 50 spectres, la seconde le nombre de pics trouvées dans au moins 49 spectres, la troisième dans au moins 48, etc... jusqu'à la dernière qui indique le nombre de pics trouvés dans au moins un des spectres. Donc, plus un processus est stable et plus ses premières couchent sont importantes.

L'analyse des figures 7.8 et 7.9 montrent des résultats bien différents sur les données lissées (*sm.span=31*) par rapport aux données brutes (*sm.span=0*). Tout d'abord, pour un seuil SoN identique, le nombre de pics détectés dans les données lissées sont moins nombreux que le nombre de pics détectés dans les données brutes. Ceci est assez normal car le filtre moyenneur qui permet de lisser les spectres érode les maxima locaux et diminue le rapport signal/bruit des pics. Par contre, le lissage du signal avec *sm.span=31* apporte un gain de stabilité par rapport aux données brutes (*sm.span=0*). C'est surtout visible en comparant les hauteurs des premières couches des histogrammes des figures 7.8 et 7.9. Cela suggère qu'il est bon d'effectuer une telle opération sur les données de spectrométrie de masse à faible résolution pour éliminer une partie du bruit (très abondant dans ces données), même si cela détériore un peu le signal.

Si on se concentre maintenant sur les données lissées et que l'on compare la stabilité de PROcess à celle de *ms*, il est clair que *ms* est plus stable que PROcess, surtout lorsque le paramètre SoN varie entre 2 et 4. Pour les valeurs

extrêmes de ce paramètre, la différence est par contre moins significative. Cela vient du fait que lorsque SoN est petit la détection de pic est très sensible et les deux algorithmes détectent tous les deux beaucoup de bruit, et à l'inverse, lorsque SoN est grand, ils extraient très peu de pics en se focalisant sur les plus évidents. Les qualités de *ms* sont donc d'autant plus appréciables qu'il se montre plus stable que PROcess dans cette plage de valeur intermédiaire où les rapports signaux/bruits des pics permettent difficilement de les classer comme pic ou comme bruit.

En ce qui concerne la comparaison de la stabilité des algorithmes sur les données brutes, les motifs que l'on observe sont un peu différents (la plupart des cas ressemblent à DHB-UF5kD) : la densité de la matrice d'expression générées par PROcess est au départ supérieure à celle de *ms* et à tendance à devenir inférieure lorsque le paramètre SoN augmente. La raison pour laquelle PROcess est plus stable que *ms* pour les petites valeurs de SoN tient à sa façon de détecter les pics : lorsque la détection est très sensible (SoN faible), PROcess sature le spectres de masse de pics, ils se retrouvent alors plus ou moins à la même position dans les différents spectres de masse, et cela produit une bonne stabilité apparente. Or PROcess arrive beaucoup plus vite à saturation que *ms*. Effectivement, PROcess opère une recherche des maxima locaux en considérant 101 points autour d'eux c'est à dire que les pics détectés par PROcess sont espacés d'au moins 50 points. A l'inverse, dans le pacquage *ms*, la position des pics est beaucoup plus libre et il n'y a pas cette contrainte implicite sur l'espacement minimum entre deux pics. En conséquence quand nous détectons avec *ms* autant de pics que PROcess quand il sature, ils sont positionnés moins uniformément et paraissent moins stable. Mais la réalité c'est qu'il y a une saturation et que du bruit est détecté. Du fait de la meilleures stabilité de *ms* lorsque le rapport signal/bruit est au delà de 3.5 et que le phénomène de saturation est atténué, il semble donc encore une fois préférable de privilégier *ms* à PROcess. D'autant que si l'on compare les premières couches des histogrammes (toujours pour *sm.span=0*), celles de *ms* sont souvent plus élevées que celles de PROcess.

7.3.3 Comparaison des performances de classification

Comme nous l'avons mentionné en début de chapitre la bonne stabilité d'un algorithme de pré-traitement n'est pas à elle seule une garanti de bonne performance. Nous l'avons d'ailleurs observé avec la saturation de PROcess qui produit une stabilité supérieure à celle de *ms* mais qui n'est pas forcement synonyme de meilleur résultat. Il faut donc également s'assurer que le pré-traitement fait ce que l'on attend de lui. Pour cela, cette section propose de comparer les performances de classification obtenues avec un pré-traitement réalisé selon PROcess à celles obtenues avec le pré-traitement de *ms*, car plus les performances sont bonnes et plus les pré-traitements sont supposés extraire l'information pertinente.

Pour réaliser cette étude, les trois jeux de données utilisés, stroke, ovarian et prostate, sont identiques à ceux employés dans Prados et al. [90] et dans la section 7.2. Rappelons que chacun définit un problème de classification bi-

naire dont le but est de prédire l'état clinique (sain/malade) de patients à partir de spectres de masse réalisés avec leur sérum. Rappelons aussi que les données prostate et ovarian sont publiques et que nous les avons télé-chargées depuis internet (`http://home.ccr.cancer.gov/ncifdaproteomics/ppatterns.asp`), malheureusement, il semble qu'elles aient déjà subit un pré-traitement (élimination de la ligne de base et lissage), mais dans la suite nous les traitons comme si il s'agissait de données brutes.

Méthodologie

La méthodologie suivie pour obtenir les matrices d'expression qui servent à réaliser les apprentissages est à peu de chose prêt identique à celle suivie pour comparer la stabilité de PROcess et *ms* (voir section 7.3.2). Ce qui change, c'est d'une part que nous ignorons la partie des spectre située avant 2000Da, et d'autre part que nous normalisons par le courant ionique total (après lissage des spectres et élimination de la ligne de base) pour uniformiser les intensités des pics qui sont utilisées pour générer les valeurs de la matrice d'expression. Au final, nous suivons donc le schéma de la figure 7.3 en incluant les étapes facultatives. Le paramètre SoN qui contrôle la sensibilité de la détection est fixé à 2.0, c'est à dire une valeur relativement sensible pour ne pas perdre d'information et car nous envisageons d'éliminer le bruit d'une autre manière.

Ainsi pour chaque jeux de données une première matrice d'expression est obtenu en appliquant le pipeline PROcess (paramétré par SoN), et une deuxième en demandant à *ms* de détecter autant de pics que PROcess dans chaque spectre de masse. Les deux matrices d'expression générées contiennent donc le même nombre d'information et aucune des deux méthode n'est ainsi privilégiée. Aussi, toujours dans le but de préserver cette équité, les valeurs manquantes sont remplacées par 0 plutôt que recherchées dans les spectres de masse. Les deux matrices d'expressions obtenus sont utilisées pour évaluer la performance de classification d'une machine à vecteurs supports linéaire par validation croisée à 10 couches. Rappelons que ce classificateur est celui qui s'est montré le plus performant pour traiter les données de spectrométrie de masse [89, 90], et nous avons fixé son paramètre C qui contrôle la complexité de la machine à vecteurs supports à $C = 1$.

Enfin, nous essayons également d'étudier l'impact des approches d'élimination de bruit sur les performances de classification. Ce qui nous intéresse notamment, c'est d'observer l'effet du lissage des spectres sur les performances de classification, et de voir si effectivement la détérioration du signal dans le lissage diminue les performances de classification. Nous évaluons les performances de la machine a vecteur support linéaire sur des matrices obtenus avec et sans lissage des spectres (*sm.span=31* et *sm.span=0*). L'autre manière que nous avons pour éliminer le bruit c'est d'ignorer les pics qui apparaissent dans peu de spectres de masse. Nous essayons donc également de comparer les résultats lorsque tout les pics sont considérés (Cov=0) et lorsque l'on élimine ceux qui sont présents dans moins de 5 spectres (Cov=5). Au total nous testons quatre combinaisons de paramètres de pré-traitement pour chaque jeu de donnée.

	PROcess param.			ms			PROcess			
donnée	SoN [a]	sm [b]	Cov [c]	#att [d]	densité [e]	err [f]	#att [d]	densité [e]	err [f]	sig [g]
stroke	2.0	0	0	669	9.59	25.4	678	9.46	31.2	0.134
	2.0	0	5	463	13.30	25.4	549	11.39	33.6	**0.034**
	2.0	31	0	140	11.98	38.9	141	11.90	42.7	0.433
	2.0	31	5	74	21.89	35.5	77	21.08	43.2	0.101
ovarian	2.0	0	0	309	26.07	3.1	277	29.08	5.9	0.121
	2.0	0	5	233	34.30	3.1	230	34.87	5.9	0.121
	2.0	31	0	252	26.54	3.5	242	27.63	3.5	1
	2.0	31	5	198	33.57	3.1	192	34.61	3.9	0.802
prostate	2.0	0	0	533	13.67	18.6	428	17.02	17.0	0.602
	2.0	0	5	462	15.65	19.8	360	20.09	18.6	0.707
	2.0	31	0	366	13.25	13.9	317	15.30	16.1	0.442
	2.0	31	5	300	16.01	13.0	246	19.51	17.3	0.098

TABLE 7.4 – Propriétés des matrices d'expression générées par PROcess et ms, et comparaison de leurs performances de classification. Chaque ligne du tableau correspond à différentes valeurs des paramètres controlant le pipeline de pré-traitement de PROcess, le pipeline ms est configuré pour détécter autant de pics que PROcess.

a. Rapport signal/bruit minimum d'un pic

b. Valeur du paramètre *sm.span* égale à la largeur de la fenêtre utilisée pour lisser les spectres de masse

c. Nombre minimum de valeur non-manquante dans les colonnes de la matrice d'expression

d. Nombre de colonne (d'attributs) dans la matrice d'expression

e. Densité de la matrice d'expression (pourcentage de valeurs non-manquante)

f. Erreur de classification du SVM linéaire (dont le paramètre C est fixé à 1) estimée par validation croisée à 10 couches

g. Résultat du test de significativité de Mc Nemars comparant les 2 erreurs de classification

Résultats de la comparaison de performance

Les résultats des expériences décrites ci-dessus sont résumés dans le tableau 7.4. Chaque ligne donne, pour une combinaison de paramètre possible, les caractéristiques des matrices d'expressions générées par PROcess et *ms* et l'évaluation de l'erreur de classification sur ces données. La première constatation est que les erreurs de classification de du pipeline *ms* ne sont jamais significativement supérieures à celles de PROcess. Dans un des cas (stroke avec *sm.span=0* et Cov=5) son erreur est même significativement inférieure à celle de PROcess, et dans la plupart des autres elle est légèrement inférieure. La seule fois où l'erreur de *ms* est plus élevée que celle de PROcess (non significativement) c'est sur prostate sans lissage des spectres (*sm.span=0*). Nous notons aussi que, sur ces résultats, il ne semble pas y avoir de lien entre d'une part la densité et la dimension de la matrice d'expression, et d'autre part l'erreur de classification. A priori, nous aurions pourtant pu penser que plus une matrice est dense et plus les performances de classification sont élevées, car une forte densité signifie à la fois que l'information est concentrée ce qui devrait faciliter l'apprentissage, et aussi que le pré-traitement est stable, ce que l'on attend d'un pré-traitement efficace.

En ce qui concerne l'impact des méthodes de suppression de bruit sur les performances de classification, il apparaît tout d'abord que l'élimination des

pics présents dans moins de 5 spectres (Cov = 5) n'a pas d'impact sur l'erreur de classification alors que cela réduit le nombre d'attribut d'environ 20%. Ce résultat n'est pas étonnant, car il y a peu de chance que l'algorithme de classification base ses prédictions sur les attributs ainsi éliminées puisqu'ils ne lui permettent, au mieux, que de prédire seulement 5 instances sur les centaines que contiennent les jeux de donnée. En revanche, le lissage du signal semble influencer d'avantage les erreurs de classification : sur stroke il induit une augmentation significative de l'erreur de classification, et sur prostate il la diminue.

7.3.4 Bilan de la comparaison des pré-traitements

La comparaison du pré-traitement de *ms* par rapport au pré-traitement de PROcess joue clairement en faveur de *ms*. Effectivement, celui-ci est capable d'extraire une matrice d'expression qui conserve le même pouvoir de discrimination que le pré-traitement de PROcess mais avec une stabilité bien supérieure. Cette supériorité et terme de stabilité s'explique essentiellement par notre manière simple et élégante de détecter les pics directement à partir des données brutes. Signalons à ce sujet que l'algorithme de détection des pics qu'implémente PROcess, et qui consiste en une recherche des maxima locaux, est largement répandu dans la littérature ce qui ouvre de nombreuses perspectives d'amélioration en intégrant les méthodes que l'on propose dans *ms*.

De plus les expériences que nous avons mené dans ce cadre nous ont également permis de compléter notre analyse sur l'influence des paramètres de pré-traitement : d'une part nous avons pu observé l'utilité du lissage du signal qui améliore la stabilité du pré-traitement mais qui perturbe aussi les performances de classification ; et d'autre part, nous avons pu constater que l'on pouvait éliminer du bruit en supprimant les colonnes de la matrice d'expression qui ont de nombreuses valeurs manquantes sans affecter les performances de classification.

7.4 Bilan de l'évaluation du pré-traitement

Dans ce chapitre, nous avons mené de nombreuses expériences pour analyser le comportement du pipeline de pré-traitement présenté dans les chapitres précédents. Par des expériences objectives qui font intervenir d'une part les performances de classification et d'autre part la stabilité des structures générées, nous avons pu étudier le comportement de nos algorithmes et choisir la représentation des données la plus adaptée. L'effort qui est fait dans ce chapitre pour montrer la qualité de notre pré-traitement est certainement ce qui met notre travail le plus en valeur, car nous ne nous contentons pas de proposer des algorithmes pour le pré-traitement des spectres de masse, mais nous montrons également qu'ils sont efficaces et stables face aux variations des données. De plus les résultats ont montré que le pipeline *ms*, que nous proposons, obtient des résultats de meilleures qualité que d'autres pipelines existants qui sont fondés sur des approches répandues (nous faisons ici références à notre comparaison avec le pipeline PROcess et sa détection de pics par maxima locaux).

Il est également important de souligner que toutes ces expériences sont basées sur des données expérimentales qui reflètent toute la complexité du protocole biologique, ce qui rend nos résultats d'autant plus intéressants. Effectivement, comme tous nos spectres de masse proviennent d'échantillons complexes, qu'on ne connaît pas leur composition, et qu'aucune donnée n'est synthétique, nos conclusions correspondent à des conditions expérimentales réelles et non pas à des simplifications qui pourrais affecter les observations.

Malgré nos efforts pour couvrir les différentes étapes du pré-tratiement, un des aspect est resté un peu de coté dans nos analyses. Il s'agit de la normalisation des données qui est pourtant un point important dans la génération de la matrice d'expression. Effectivement, selon la manière dont est réalisée la normalisation des spectres de masse, des conclusions différentes sont tirés sur l'abondance des protéines dans les échantillons, et une mauvaise normalisation des données rend difficile la comparaison des valeurs de la matrice d'expression entre elles. Jusqu'à présent, nous nous sommes contentés d'appliquer une normalisation des spectres de masse par "courant ionique total" lorsque cela était nécessaire car c'est une approche largement acceptée, néanmoins, on peut se poser la question de savoir si cette normalisation est appropriée.

La seconde partie de cette thèse aborde plus spécifiquement cet aspect de la normalisation des données et tache d'apporter des solutions. Dans cette seconde partie, on ne se limite cependant pas au domaine de la spectrométrie de masse, mais on se place dans un cadre plus général car, comme nous allons le voir, le problème de la normalisation est très largement répandu. Ce qui va spécialement retenir notre attention, c'est l'effet de la normalisation des données sur le processus de sélection des attributs qui nous permet d'extraire de la matrice d'expression des bio-marqueurs potentielles. Rappelons à ce sujet que le pré-traitement a déjà permit de réduire fortement la dimension des données en structurant et en focalisant l'information sur ce qui a un sens biologique (les pics des spectres), mais la sélection des attributs permet d'aller encore plus loin en exploitant les corrélations entre les données de la matrice d'expression et la classe associée à chaque spectre. Le but de la sélection des attributs est effectivement de réduire la dimension de la matrice d'expression en préservant son pouvoir discriminatif vis à vis des patients. Il s'agit d'extraire un petit nombre de protéines dont l'activité permet de distinguer les spectres provenants de patients "contrôles" des spectres provenants de patients "malades", c'est à dire des bio-marqueurs potentiels de la maladie. Comme nous allons le voir dans la deuxième partie, la normalisation des données peut avoir une influence importante sur la sélection des attributs et nous allons tacher de proposer une méthode de sélection qui soit robuste à la normalisation des données, et qui en même temps s'inspire des concepts importants de la biologie comme l'interaction entre les protéines.

Deuxième partie

Data mining : Sélection d'attributs en vu de l'extraction des biomarqueurs

Dans la première partie nous avons vu des méthodes de pré-traitement qui permettent de structurer et réduire la quantité d'information dans les expériences de spectrométrie de masse. Ces pré-traitements utilisent les connaissances du domaine afin d'extraire une matrice d'expression qui a un sens biologique fort. Dans cette seconde partie, on s'intéresse au problème de la recherche de bio-marqueurs potentiels dans cette matrice d'expression. Le but ici est encore une fois de réduire la quantité d'information mais en exploitant l'information de classe associée à chaque profil d'expression dans la réduction. Effectivement, on cherche maintenant à identifier un petit nombre de protéines de la matrice d'expression dont l'activité permet de distinguer si le spectre de masse provient d'un patient contrôle ou d'un patient malade. Cela nous conduit à nous intéresser aux méthodes de sélection d'attributs car les objectifs visés sont exactement les mêmes que ceux que l'on cherche à atteindre lors de extraction des bio-marqueurs.

Etant donnée que la sélection d'attribut a une portée plus générale que l'extraction des bio-marqueurs en spectrométrie de masse et qu'elle est applicable à de nombreux domaines, nous avons préféré consacrer une partie distinct à ce thème. C'est également pourquoi notre propos et les expériences menées dans cette partie incluront d'autres domaines que la spectrométrie de masse. Nous resterons cependant avec des données relatives au domaine de la biologie, car c'est un domaine producteurs de nombreuses données de grande dimension (puces à ADN, spectrométrie de masse, article scientifiques PUBMed), et les biologistes ressentent un besoin grandissant pour des outils capables d'analyser et d'extraire de la connaissance dans ces informations. Signalons tous de même que le design des approches qui sont proposées dans cette partie est largement inspiré des problématiques que l'on rencontre spécifiquement pour l'extraction des bio-marqueurs en spectrométrie de masse.

Lorsque l'on s'intéresse comme nous aux méthodes de sélection d'attributs, se pose la question de leur comparaison. Le critère le plus couramment utilisé dans ces cas là consiste à mesurer la qualité de l'information contenue dans les sous-ensembles d'attributs sélectionnés par les algorithmes. Nous proposons dans le chapitre 9 d'utiliser également la stabilité comme critère de qualité des algorithmes de sélection d'attributs. La problématique ici est en réalité très proche de celle de l'évaluation du pré-traitement en spectrométrie de masse. Effectivement, lors de la détection des pics dans un spectre de masse, on souhaite d'une part que les pics détectés correspondent à des pics réelles et non pas à des artefacts ; et d'autre part que la détection soit reproductible (ou stable), c'est à dire que l'on détecte les pics aux même positions dans tous les spectres issus d'un même échantillon.

Dans cette partie, nous présentons aussi un système de sélection d'attributs capable d'extraire des sous-ensembles d'attributs en tenant compte de la redondance d'information, des interactions attributs-attributs, et des problèmes de normalisation des données qui sont des problématiques que l'on rencontre souvent en biologie (chapitre 10). Ce système est basé sur l'étude d'une fonction noyau (logRatio), au départ choisie pour son adéquation avec ces réalités du domaine de l'extraction des bio-marqueurs en spectrométrie de masse, puis l'on

s'aperçoit que les propriétés et l'interprétation des modèles qui en découle nous permettent, en plus, de définir un algorithme de sélection d'attributs similaire à SVMRFE avec quelques avantages supplémentaires.

Pour présenter notre travail, la partie est structurée en quatre chapitres : dans le chapitre 8 nous introduisons la problématique de la sélection des attributs en montrant les avantages et les lacunes des approches existantes pour l'extraction des bio-marqueurs ; puis dans le chapitre 9, nous nous intéressons à la stabilité des méthodes de sélection d'attributs comme mesure de qualité ; le chapitre 10 développe notre approche de sélection d'attributs et la teste en autre à l'aide des outils du chapitre précédant ; enfin nous concluons dans le chapitre 11. Signalons qu' une bonne partie de ce travail a fait l'objet de plusieurs publications [58, 59, 92, 91].

Dans cette partie, nous utiliserons les conventions de notations suivantes : des caractères alphabétiques minuscules avec une police standard pour désigner les nombres scalaires de $\Re$ comme par exemple α, y_i, x_i ; des caractères alphabétiques minuscules avec des caractères gras pour désigner des vecteurs de $\Re^n$ comme par exemple $\mathbf{x}$ où $\mathbf{x}_i$.

Chapitre 8

Introduction à la sélection des attributs

A propos de la sélection des attributs, nous avons déjà mentionné en introduction, que ceux sont des techniques qui ont été développées au départ pour répondre aux problèmes de grande dimensionalité des données qui apparaissent de plus en plus fréquemment en apprentissage automatique. Effectivement, après une phase intensive de pré-traitement des données, la grande dimension des ensembles d'apprentissage obtenus pose encore des problèmes d'analyse, car on observe un phénomène de sur-apprentissage des méthodes de classification classiques sur ce type de données [45]. Les algorithmes de sélection d'attributs offrent heureusement des solutions pour contourner ces problèmes en focalisant l'apprentissage sur un sous-ensemble de variables pertinentes. En particulier, l'algorithme SVMRFE (Support Vector Machine Recursive Feature Elimination) s'est distingué pour le traitement des puces à ADN [43]. Cet algorithme tire profit de l'excellent comportement des l'algorithmes de classification de type "machine a vecteurs supports (SVM) sur les données de grande dimension en offrant une interprétation des modèles SVM à noyau linéaire qui permet d'ordonner les attributs [1] en fonction de leur utilité dans la prise de décision. Cependant, SVMRFE montre des problèmes face à la normalisation des données et vis à vis de la redondance d'information (voir 10.3). Nous aborderons le fonctionnement de SVM et SVMRFE dans les sections 8.3, 8.3.

Dans le contexte particulier de l'extraction des bio-marqueurs, de nombreuses méthodes de sélection d'attributs ont été testés dont SVMRFE (pour une revue, voir [47] et [103]). Pourtant, plusieurs points sont mis en avant par les experts du domaine qui vont à l'encontre des problèmes dont souffre SVMRFE, et que plus généralement, peu de méthodes d'apprentissage considèrent. Par exemple, les biologistes ont des problèmes pour normaliser correctement leurs données de spectrométrie de masse et de puces à ADN : une même expérience répétée plusieurs fois ne produit pas les mêmes instances d'apprentissage ce

1. plus exactement des sous-ensembles d'attributs

qui perturbe de nombreux algorithmes et perturbe notamment les résultats de SVMRFE (voir section 8.3). Les biologistes s'intéressent également de prêt aux protéines qui montrent des similitudes dans leurs expressions car celles-ci sont susceptibles d'interagir entre elles, mais malheureusement cela se traduit par de la redondance d'information dans les jeux de données que SVMRFE à des problèmes à traiter (voir section 10.3). Dans la même idée, les biologistes s'intéressent de prêt aux interactions entre protéines/gènes et leur impact sur l'état des spécimens étudiés (c'est à dire leur classe contrôle/malade), car le changement d'état d'un spécimen est rarement liée à l'activité d'une protéine isolée, mais il est plus souvent causés par des mécanismes qui activent et/ou inhibent l'interaction entre les protéines. Ces observations suggèrent donc de focaliser l'apprentissage sur les interactions protéine-protéine plutôt que sur l'activité isolée des protéines pour mettre l'accent sur l'information recherchée par les utilisateurs.

Dans ce chapitre, nous allons voir plus en détails le fonctionnent de ces approches et mettre en lumière les lacunes dont elles souffrent afin de pouvoir étudier par la suite une solution qui essaie de résoudre ces problèmes.

8.1 Objectifs de la sélection d'attributs

Dans l'apprentissage supervisé moderne, les algorithmes de sélection d'attributs sont devenus des outils indispensables pour traiter les jeux de données de grande dimension qui se répandent de plus en plus. La tâche de ces algorithmes est de trouver de petits sous-ensembles G de m attributs (parmi les n attributs de l'ensemble F de départ) qui minimise l'erreur de généralisation des algorithmes d'apprentissage que l'on y applique [64]. Par l'emploi de ces méthodes, le *data-miner* vise trois objectifs : 1) améliorer la vitesse d'exécution des algorithmes et diminuer l'espace mémoire nécessaire au stockage des données ; 2) simplifier et améliorer l'interprétation des modèles ; 3) améliorer les performances d'apprentissage en évitant le problème posée par la grande dimension des données ("curse of dimensionality") qui expose les modèles au risque du sur-apprentissage qui les rends trop spécifiques aux données d'entraînement [45]. Ce dernier point est souvent considéré comme le plus important, mais si on se place d'un point de vu bio-informatique et en particulier pour la recherche de bio-marqueurs, il ne faut pas non plus négliger le point numéro 2), car dans ce domaine on s'intéresse avant tout à comprendre les interactions entre attributs ce qui nécessite de produire des modèles simples et très compréhensibles. Dans la suite, nous allons voir les concepts importants lorsque l'on s'intéresse à la sélection des attributs.

8.1.1 Pertinence et Redondance

Les notions de pertinence et de redondance pour les attributs sont deux concepts importants pour la sélection d'attributs [64, 130]. En effet, idéalement, un algorithme de sélection d'attributs doit pouvoir éliminer les attributs non-pertinents (qui n'apportent pas d'information à l'attribut de classe), mais aussi

traiter les attributs redondants de manière consistante. Ces attributs redondants peuvent par exemple apparaître lorsqu'un jeu de donnée contient des informations exprimées dans des unités de mesure différentes, ou plus généralement, si il est possible de déduire la valeur d'un attribut à partir de celles d'autres. Dans ce cas, les attributs redondants n'apportent pas une réelle information, et certains peuvent être éliminé. Par contre, comme le souligne [42], la redondance d'information n'est pas quelque chose de nécessairement néfaste et qu'il faut absolument éliminer. En particulier, lorsque les attributs sont bruités, il peut être intéressant d'exploiter les variables redondantes. Considérons par exemple n attributs correspondants à n mesures indépendantes de la même valeur (avec erreur/bruit). L'information qu'ils contiennent peut paraître redondante, et pourtant ils sont tous importants car si on en fait la moyenne, on améliore la précision sur la valeur mesurée d'un facteur $\sqrt{n}$ par rapport aux erreurs originales. Or la précision accrue que l'on obtient peut aider à distinguer les classes du problème, et dans ce cas, il faut conserver tous les attributs.

Au final donc, plutôt que d'éliminer les attributs redondants, il est certainement préférable de proposer des approches qui structurent l'espace des variables pour fournir des informations sur la redondance entre les attributs. L'approche de [129] est intéressante à ce sujet, elle consiste a former des groupes de variables denses qui mettent ensemble les variables qui apportent une information similaire sans tenir compte de la classe des instances. L'information de classe intervient dans un deuxième temps pour choisir les groupes de variables pertinents. De manière similaire, d'autres alternatives peuvent être envisagé qui reposent sur des regroupements de variables redondantes et qui inclut ou exclut la totalité d'un groupe. On pense par exemple aux approches de type "discriminant vector quantization" [54] et "information bottleneck" [117].

Se pose également la question de distinguer la redondance utile (qui peut améliorer les résultats de classification) de la redondance inutile (qui n'apporte pas d'information supplémentaire aux informations déjà à notre disposition) ? Pour une réponse théorique, il faut se reporter aux récents travaux de [130], qui complètent ceux de [65, 64], et dans lesquelles une définition formelle est donnée pour identifier les attributs pertinent et les attributs redondants. De ces travaux il ressort que l'on peut classer les attributs d'un sous-ensemble en trois catégories : I) *irrelevant features*, ceux sont les attributs qui n'apportent aucune information vis à vis de la classe ; II) *weakly relevant*, ceux sont les attributs qui apportent une information vis à vis de la classe, mais celle-ci est similaire à celle contenue dans d'autres attributs ; III) *strongly relevant features*, ceux sont les attributs qui apportent vis à vis de la classe une information qui n'est contenue dans aucun autre attribut. Un algorithme de sélection d'attributs doit être en mesure d'identifier les attributs de la catégorie II et III, car ceux sont les seuls qui apportent une information vis à vis de la classe. De plus, il est clair que les attributs qui apportent une redondance d'information fond partie de la catégorie II, et [130] propose d'étendre cette catégorie en distinguant deux sous-catégories en son sain. Il distingue d'une part, II.1) les attributs *weakly relevant and redundant* et d'autre part, II.2) les attributs *weakly relevant but non-redundant*. Il fait donc clairement apparaître la notion de redondance entre

attributs et il suggère que l'objectif d'un algorithme de sélection d'attributs est d'extraire un sous-ensemble qui contient seulement des attributs des catégories II.2 et III. La définition de la redondance selon [130] est basée sur la notion de *Markov blanket* : un attribut *weakly relevant* est considéré comme redondant si il existe un sous-ensemble d'attributs qui contient la même information que lui non seulement vis à vis de la classe, mais aussi vis à vis des autres attributs. Cela permet de partir d'un ensemble d'attributs et d'éliminer récursivement des attributs redondants avec la garantie que l'on conserve un maximum d'informations sur les données. Le problème des *Markov blanket* c'est que les sous-ensembles d'attributs sont difficiles à trouver en pratique, même si [65] et [130] proposent des algorithmes pour s'en approcher en se basant sur les corrélations entre paires attributs (Pearson Correlation pour le premier et *symmetrical uncertainty* pour le second).

Tous les algorithmes de sélection d'attributs ne sont pas capables de traiter la redondance. Une condition nécessaire pour qu'ils aient cette faculté est qu'ils soient *multivariés* [42], c'est à dire qu'ils doivent considérer l'ensemble des variables du jeu de donnée pour décider si un attribut doit être éliminé ou non. Par opposition, les approches *univariées* qui considèrent les attributs indépendamment les uns des autres, ne peuvent que capturer la pertinence et ne peuvent pas déterminer les variables qui sont redondantes à d'autres. L'algorithme de sélection d'attributs SVMRFE (section 8.3), sur lequel se base la méthode que nous proposons dans cette partie, est multivariée. Il permet donc potentiellement d'éliminer les attributs redondants, mes les résultats expérimentaux obtenus avec SVMRFE (voir section 10.3) montrent qu'il échoue pour distinguer les attributs qui apportent une redondance d'information utile de ceux qui apportent une redondance inutile. Par contre, l'extension de SVMRFE que nous proposons dans cette partie focalise davantage l'apprentissage sur les interactions entre attributs, ce qui la rend très efficace pour distinguer entre les deux types de redondance.

8.1.2 Stabilité

En lien avec la présence de redondances dans les données, se pose la question de la stabilité des algorithmes de sélection d'attributs. En effet, comme le suggère [129], si deux attributs sont très redondants, de petites variations dans les données risquent de privilégier tantôt un attribut, tantôt un autre et de se traduire par une instabilité de l'algorithme de sélection. Dans une tel situation, des approches qui cherchent à grouper/combiner des attributs redondants auront tendance à être plus stable que des approches qui éliminent simplement les attributs. Effectivement, de petites variations des données risque de faire choisir à ces dernières des attributs différents, alors que les premières pourrons exploiter les informations de redondance pour produire un résultat plus stable. Nous reviendrons plus en détail sur la stabilité des algorithmes de sélection d'attributs dans le chapitre 9.

8.2 Les Machines à Vecteurs Supports (SVM)

Les machines à vecteurs supports (SVM) sont des classificateurs linéaires pour problèmes binaires (à deux classes) qui recherchent l'hyperplan de marge maximale séparant les données de la classe positive de celles de la classe négative et qui sont réputés très efficaces pour traiter des données de grande dimension [22]. Ces machines autorisent l'emploi de fonctions noyaux ($K(\mathbf{x},\mathbf{z}) = \langle\phi(\mathbf{x}).\phi(\mathbf{z})\rangle$) pour implicitement projeter les instances dans un espace de caractéristiques (*feature space*) où est réalisée la séparation linéaire [107]. Suivant la nature de la projection ϕ, différentes séparations de l'espace initial, pas forcement linéaires, sont obtenues. Par exemple, un noyau polynomial de degré 2 convertit implicitement un vecteur $\mathbf{x}$ de dimension n en un vecteur $\phi(\mathbf{x})$, de dimension n^2 contenant l'ensemble des produits des attributs de x pris deux à deux. La fonction de projection ϕ associée à ce noyau est donc donnée par Eq.(8.1), de plus, grâce à l'Eq. (8.2), le noyau polynomial de degré 2 se calcul avec une complexité de $\mathcal{O}(n)$ alors que l'espace de caractéristiques contient n^2 dimensions :

$$\phi(\mathbf{x}) = (x_i x_j)_{(i,j)=(1,1)}^{(n,n)} \tag{8.1}$$

$$K(\mathbf{x},\mathbf{z}) = \langle\phi(\mathbf{x}).\phi(\mathbf{z})\rangle = \sum_{(i,j)=(1,1)}^{(n,n)} (x_i x_j)(z_i z_j) = \left(\sum_{i=1}^{n} x_i z_i\right)^2 \tag{8.2}$$

La fonction de décision ($f(\mathbf{x})$) du modèle appris par un SVM correspond à l'équation de l'hyperplan séparateur de marge maximal dans l'espace de caractéristiques et s'exprime au moyen d'une combinaison linéaire de la fonction noyau (K) entre les instances d'entraînement ($\mathcal{T} = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \Re^n, y_i \in \{-1, 1\}\}$) d'une part et l'instance de test ($\mathbf{x} \in \Re^n$) d'autre part. Les coefficients (α_i) et l'intercepte (b) de l'hyperplan sont déterminés durant la phase d'apprentissage, et on a donc :

$$f(\mathbf{x}) = \text{sgn}(\sum_i y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b) \tag{8.3}$$

Dans le cas du noyau linéaire qui n'est autre que le produit scalaire standard, l'espace de caractéristiques et l'espace initial coïncident. Les propriétés algébriques du produit scalaire permettent alors de simplifier la fonction de décision pour laisser apparaître le vecteur $\mathbf{w}$ normal à l'hyperplan séparateur. On a en effet :

$$f(\mathbf{x}) = \text{sgn}(\sum_i y_i \alpha_i \langle\mathbf{x}_i.\mathbf{x}\rangle + b) = \text{sgn}(\langle\mathbf{w}.\mathbf{x}\rangle + b) \tag{8.4}$$

$$\text{avec : } \mathbf{w} = \sum_i y_i \alpha_i \mathbf{x}_i \tag{8.5}$$

Cette simplification facilite l'interprétation des modèles SVM puisque un poids (w_i) est associé à chaque attribut de l'espace initial, et permet de déterminer

ceux qui participent le plus dans les prédictions du modèle. Nous allons voir dans la section qui suit comment l'algorithme SVMRFE tire profit de cette interprétation pour réaliser une sélection d'attributs, mais avant, voyons comment trouver l'hyperplan de marge maximal.

8.2.1 Recherche de l'hyperplan à marge maximale

Problème Primal

L'équation de l'hyperplan de marge maximale séparant les données de la classe positive de celles de la classe négative dans l'espace de caractéristiques (les figures 8.1a et 8.1c en montrent deux exemples simples) s'obtient en résolvant le problème de minimisation (8.6) [22] :

$$\begin{cases} \text{minimise}_{\xi,\mathbf{w},b} : & \langle \mathbf{w}.\mathbf{w} \rangle + C \sum_{i=1}^{l} \xi_i \\ \text{subject to :} & y_i(\langle \mathbf{w}.\mathbf{x}_i \rangle + b) \geq 1 - \xi_i, i = 1..l \end{cases} \tag{8.6}$$

Les contraintes imposent que les instances d'entraînement soient situés de part et d'autre des marges, et les variables de relâchement (*slack variables*) ξ_i permettent que des erreurs soient commises. Le paramètre C, fixé par l'utilisateur, pondère les variables de relâchement dans la fonction objective ce qui permet de contrôler la "gravité" de l'erreur. Ce paramètre peut varier entre 0 et $+\infty$, et plus sa valeur augmente, plus la solution est "rigide" dans le sens où moins d'erreurs sont permises. L'une des difficultés dans l'utilisation de ce type de SVM est justement de régler C dont la plage de variation n'est pas borné, et dont l'interprétation est non triviale.

Problème Dual

Plutôt que de résoudre le problème de minimisation primal, l'on préfère souvent résoudre son dual (8.7) dont la solution est équivalente mais qui offre l'avantage 1) de limiter le nombre d'inconnus (α_i) au nombre d'instances dans l'ensemble d'entraînement plutôt qu'au nombre de variables dans l'espace des caractéristiques (w_i), 2) d'introduire une fonction noyau (K) pour le calcul des produits scalaires, 3) d'avoir des contraintes plus simples. Plusieurs algorithmes spécifiques à ce problème de minimisation quadratique ont été proposées, dont le populaire SMO [22].

$$\begin{cases} \text{minimise}_{\alpha} : & \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_i \alpha_i \\ \text{subject to :} & \sum_i \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \end{cases} \tag{8.7}$$

SMO, et certaines de ses variantes comme par exemple LIBSVM [13], permettent donc de résoudre le problème d'optimisation dual avec une complexité quadratique avec le nombre d'instance. Récemment, les avancées dans le domaine de la prédiction de structures, ont cependant permis de proposer des

approches pour résoudre le problème du SVM beaucoup plus rapidement [55, 31, 115]. Ces approches sont basées sur des algorithmes de type *cutting plane* et ont une complexité linéaire avec le nombre d'instances. En revanche, elles résolvent directement le problème d'optimisation primal et non pas le problème dual ce qui rend plus difficile l'introduction de fonctions noyaux. Dans toutes nos expériences, nous avons utilisé SMO d'une part car sont implémentation était beaucoup plus répandue au moment des analyses, et d'autre part pour faciliter l'utilisation de fonctions noyaux. Les caractéristiques particulières de notre noyau nous auraient cependant permis d'employer ces nouvelles approches pour bénéficier d'un temps d'apprentissage beaucoup plus réduit, mais les résultats auraient étaient théoriquement les mêmes.

ν-SVM

Pour résoudre le problème du choix du paramètre C, ν-SVM offre une alternative au C-SVM dans laquelle C est substitué par le paramètre ν dont la valeur est limitée à un intervalle borné. En effet, les solutions du problème (8.7) lorsque C varie entre 0 et $+\infty$ sont équivalentes à celles du problème dual (8.8) lorsque ν varie entre 0 et 1 [22, Remark 6.13] :

$$\begin{cases} \text{minimise}_\alpha : & \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to :} & \sum_i \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq \frac{1}{l}, i = 1, .., l \\ & \nu \leq \sum_i \alpha_i \end{cases} \tag{8.8}$$

Le nouveau paramètre ν offre deux grands avantages par rapport au paramètre C. D'abord, il est plus facile à fixer car sa valeur est bornée, et il représente à la fois une borne supérieure de la proportion d'exemples d'entraînement qui ont une erreur de marge, et une borne inférieure de la proportion de vecteurs supports. Ceci rend le paramètre ν plus robuste aux éventuelles variations des données, c'est à dire que la valeur qui est choisie par l'utilisateur pour une configuration des données reste cohérente avec une autre configuration. Par exemple, dans la suite, nous faisons évoluer les données en éliminant tour à tour des variables du jeu de données. L'avantage offert par ν-SVM par rapport au C-SVM est que le paramètre ν qui est choisie avant la suppression conserve sa signification après la suppression : dans les deux cas ν impose la même contrainte sur le nombre de vecteurs support. A l'inverse, la valeur du paramètre C dans C-SVM est sensible à ces changements introduits dans les données, en conséquence, pour conserver le même nombre de vecteur support avant et après suppression des variables il faut utiliser des valeurs de C différentes.

Mentionnons l'existence d'une extension simple de ν-SVM, du nom de 2ν-SVM, qui permet de s'attaquer au problème du balancement des classes [16, 23, 24]. Cette extension propose d'associer à chacune des deux classes des paramètres ν^+ et ν^- distincts plutôt qu'un seul paramètre ν global. Comme auparavant, ces paramètres représentent à la fois une borne supérieure de la proportion d'exemples d'entraînement qui ont une erreur de marge et une borne

inférieure de la proportion de vecteurs supports, mais cette fois ils sont associés à chacune des classes. Ainsi, l'utilisateur peut contrôler indépendamment les contraintes sur les exemples de classe positive et les exemples de classe négative pour tenir compte de l'importance qu'il souhaite donner à chacune d'elle (controler la sensitivité/spécificité) ou selon la proportion des exemples dans chaque classe (le balancement des classe).

8.3 Séléction d'attributs par SVMRFE

L'algorithme de sélection d'attribut SVMRFE (Support Vector Machine et Recursive Feature Elimination) [43], repose sur la volonté d'éliminer les attributs les moins influents dans un modèle SVM (linéaire). L'idée est de considérer un attribut comme inutile si de grandes variations de sa valeur induisent de petites variations sur la prédiction du modèle SVM. Dans le cas des SVM à noyau linéaire, ces attributs sont faciles à identifier à partir de l'équation 8.4 car il est clair que les attributs qui participent le moins à la décision sont ceux qui sont associés aux facteurs multiplicateurs w_i les plus petits en valeur absolue. Ces scores qui proviennent du vecteur $|\mathbf{w}|$ et qui sont attachés à chaque attribut permettent d'ordonner les attributs du jeu de données selon leur importance en vue d'en éliminer les moins intéressants. Dans SVMRFE, ce processus d'apprentissage du SVM, d'ordonnancement des attributs, et d'élimination des moins pertinents est répété récursivement afin d'affiner l'ordonnancement des attributs : à chaque récursion, SVMRFE place les attributs qui ont les scores les plus faibles dans le modèle SVM en fin de classement, puis il ordonne les attributs restants par un appel récursif qui implique des ré-apprentissages du SVM en se limitant au sous-ensemble d'attribut restant. Remarquez que ce processus d'élimination récursive des attributs (Recursive Feature Elimination, RFE), n'est pas spécifique à SVMRFE, mais il peut être appliqué pour affiner l'ordre de n'importe quel méthode capable d'ordonner les attributs.

Une extension de SVMRFE aux noyaux non-linéaires est possible et vise toujours l'objectif de supprimer les attributs de manière à ce que les prédictions post-sélection soient le plus similaires possible aux prédictions pré-sélection [43]. Dans cette variante, les coefficients w_i pondèrent les caractéristiques de l'espace projeté et non pas les attributs de l'espace initial, de plus les w_i sont très nombreux et ils ne sont pas accessibles. La proposition de [43] pour déterminer l'attribut à éliminer est de les ordonner selon l'impact que leur suppression induit sur la valeur de la fonction objective dans le problème de minimisation (8.7). Ceci est réalisé en retenant le minimum v_0 de la fonction objective dans le problème d'optimisation 8.7, et α_0 comme vecteur solution, puis chaque attribut k est éliminé tour à tour du jeu de donnée afin d'évaluer l'impact de son élimination sur la valeur, v_k, de la fonction objective. Dans le calcul de v_k, α est supposé constant pour éviter d'avoir à résoudre le problème (8.7) à chaque suppression d'attribut, par contre les valeurs $K(\mathbf{x}_i, \mathbf{x}_j)$ changent et doivent être re-calculées. On obtient au final pour chaque attribut k un score $|v_k - v_0|$ qui permet d'ordonner l'attribut selon son impact sur la fonction objective. Cet ordre

est utilisé pour identifier les attributs les moins influents du jeu de donnée qu'il faut éliminer. Toutes ces opérations sont répétées dans des itérations RFE, c'est à dire qu'elles sont répétées plusieurs fois afin d'éliminer un à un les attributs et en déduire l'ordonnancement final.

Mentionnons également que dans le cas du noyau linéaire, plutôt que de minimiser la norme L2 de $\mathbf{w}$ dans le problème primal ($\mathbf{w}^2$) du SVM (équation 8.6), une alternative est de considérer la norme L1 de $\mathbf{w}$ ($|\mathbf{w}|$). Dans ce cas, le problème primal est linéaire et le vecteur $\mathbf{w}$ solution est généralement moins dense que la version L2 (il contient plus de valeurs zéro). Lorsque des composantes du vecteur $\mathbf{w}$ sont à zéro, elles n'influencent pas les prédictions des modèles SVM, et les variables correspondantes ne sont pas sélectionnés par SVMRFE (voir par exemple [9]). Les modèles de la version L1 sont donc généralement plus compact et influencés par moins de variables que les modèles de la version L2. Dans notre travail, nous utilisons la version L2, mais serait également intéressant de considérer les modèles L1 dans la mesure ou nous nous intéressons aux méthodes de sélection des attributs et à la stabilité des modèles.

SVMRFE à plusieurs avantages sur la plupart des autres algorithmes de sélection d'attributs : il est multivarié, et peut donc potentiellement prendre en compte la combinaison de plusieurs variables pour traiter les attributs redondant (à l'inverse des méthodes univariées) ; il est également rapide grâce à la recherche "gloutonne" (par *backward elimination*) du meilleur sous-ensemble (celle-ci peut, de plus, être accéléré en éliminant plus d'un attribut à la fois) ; Enfin, dans sa version linéaire, il est simple et produit des modèles facile à interpréter [2]. D'un autre coté la version non-linéaire perd certains avantages : les modèles sont moins facile à interpréter, et elle est moins rapide car elle à besoin de calculer plusieurs fois la fonction noyau. Le méthode que nous présentons dans cette partie (plus précisément dans le chapitre 10) est une extension de SVMRFE par un noyau non-linéaire (logRatio) dont les propriétés permettent de conserver les avantages de la version linéaire, c'est à dire que l'algorithme est rapide et qu'il génère des modèles facile à interpréter. En plus de cela, la nouvelle méthode se montre insensible à la normalisation des données qui est une des difficulté dans SVMRFE comme nous allons le voir dans la section 8.3.1 et plus généralement en apprentissage automatique (voir section 8.5).

8.3.1 Problème de la normalisation dans SVMRFE

Le résultat de la sélection d'attribut par SVMRFE dépend en partie du pré-traitement des données et notamment de la normalisation des attributs. C'est pourquoi, avant d'appliquer SVMRFE, il est recommandé d'opérer une standardisation des attributs pour les harmoniser et permettre la comparaison directe des valeurs d'un attribut à l'autre. Si aucune normalisation n'est faite, les poids du vecteur $\mathbf{w}$ résultant de l'apprentissage d'un SVM-linéaire sur les données sont affectés de manière à ce que chaque composante w_i renferme à

2. Avec les avancées récentes des algorithmes de type "plan de coupe" pour résoudre le problème primal de SVM, on pourrait même envisager d'implémenté SVMRFE avec une complexité linéaire avec la taille des données [55, 31, 115].

la fois un facteur correctif de la normalisation et un autre facteur relatif à l'importance de l'attribut dans la prise de décision du classificateur. Il s'en suit que l'interprétation des poids w_i faite par SVMRFE est biaisée par les facteurs correctifs de la normalisation, c'est pourquoi une standardisation des attributs est requise pour annuler cette composante.

La standardisation des attributs consiste essentiellement à appliquer à chaque attribut un facteur d'homothétie de manière à ce que sa déviation standard soit de 1 unité. L'ennui c'est que la standardisation ne garantit en rien de fournir la normalisation optimale attendue pour annuler la composante relative à la normalisation dans les poids des SVM-linéaires. De plus le résultat de la standardisation est dépendant de la normalisation des instances qui joue un rôle important en général, et particulièrement en bio-informatique où elle est difficile à réaliser. En effet, dans ce domaine, une instance est la plupart du temps le résultat d'une expérience dans laquelle les expérimentateurs sont susceptibles d'introduire des erreurs (par exemple sur les quantités d'échantillon déposées dans les instruments de mesure) qui affectent les valeurs des instances (e.g. spectrométrie de masse et puce à ADN). Afin qu'une même expérience répétée plusieurs fois produise des instances avec des valeurs similaires, l'usage est d'appliquer un facteur multiplicateur des instances de manière à ce que chacune ait une norme de 1 (e.g. Normalisation par le courant ionique total en spectrométrie de masse). En fournissant ainsi des instances normalisées aux algorithmes de classification, on cherche à obtenir des prédictions identiques pour un échantillon, peu importe la quantité déposée par les expérimentateurs. Mais encore une fois ceci ne garantit en rien qu'il s'agisse du coefficient idéal.

La sensibilité des méthodes SVM à la normalisation des instances a été assez bien étudié par Grapf et al. [37, 38], et les résultats ont révélés qu'une normalisation adaptée des instances pouvait améliorer significativement les performances de classification de SVM. Ces travaux comparent les performances de classification obtenues par un SVM lorsque l'on normalise les instances de l'espace initiale par rapport à lorsque l'on normalise les instances de l'espace projeté. Il est possible de normaliser les instances de l'espace projeté par leur L2-norme pour les placer sur une hyper-sphère de rayon 1 grâce à une transformation simple du noyau utilisé pour l'apprentissage. Par contre signalons que certaine caractéristiques de l'espace projeté sont parfois difficile à calculer, et il est par exemple plus compliqué de normaliser les instances de l'espace projeté par leur L1-norme. En ce qui concerne le noyau linéaire, la normalisation des instances dans l'espace projeté est équivalente à la normalisation de l'espace initiale, car les deux espaces coïncident. Les performances de classification de SVM restent donc les mêmes pour ce noyau, mais pour le noyau polynomial, il apparaît beaucoup plus efficace de normaliser l'espace projeté plutôt que l'espace initial. Ce résultat semble naturel car la normalisation de l'espace initiale ne garantie pas que l'espace projeté, où SVM opère la séparation des données, soit normalisé. Les articles de Graf et al. montrent un second résultat intéressant : on peut effectivement constater que plus le degré du noyau polynomial, et plus la normalisation de l'espace projeté améliore les performances par rapport à la normalisation de l'espace initiale. Comme le fait remarquer l'auteur, cela est

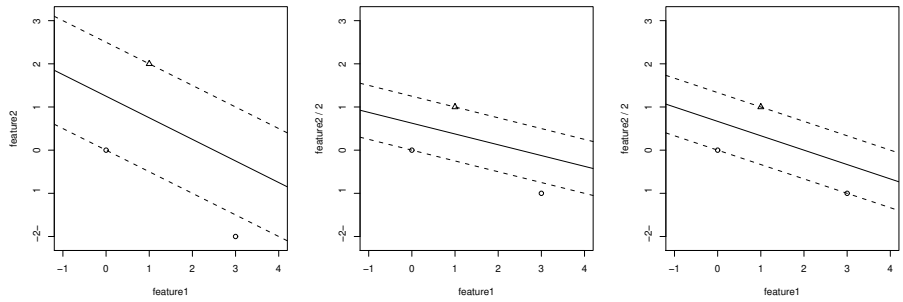


FIGURE 8.1 – Preuve de la sensibilité d'un SVM-linéraire à la normalisation des attributs : la figure a montre un jeu de donnée de trois instances et deux attributs avec sa séparation par l'hyperplan de marge maximale. La figure b montre la même chose que la figure a après que l'attribut en ordonnée (feature2) ait subit un facteur de normalisation de $\frac{1}{2}$. Hélas, la séparation obtenue n'est pas l'hyperplan de marge maximal car celui-ci est donnée par la figure c, ce qui prouve que SVM est sensible à la normalisation des attributs.

liée au fait que de petites différences sur la normalisation des instances dans l'espace initiale sont amplifiées par le noyau. Cette amplification est d'autant plus grande que le degré du noyau polynomial est élevée car on combine alors un plus grand nombre d'attributs. Cela souligne également l'importance d'une normalisation correcte de l'espace initial.

8.4 Autres approches de sélection d'attributs à base de SVM

Ces dernières années, de nombreuses méthodes de sélection d'attributs se sont développées sur la base du classificateur SVM à l'image de SVMRFE [43]. Notre objectif dans cette section est de présenter les méthodes qui ont un lien avec la méthode de sélection d'attribut que nous proposons dans le chapitre 10, et qui se focalise en particulier sur l'influence de la normalisation des données sur les méthodes de sélection d'attributs. Toutes les méthodes qui sont présentées ici souffrent effectivement d'une sensibilité à la normalisation des données appliquées en amont de l'apprentissage, et nous allons cherché dans notre travail de s'abstraire de cette sensibilité. Nous avons distinguer trois type d'approches parmi ces méthodes. Elles sont présentées dans chacune des sous-sections qui viennent. La section qui suit s'attachera ensuite a démontrer plus en détail la problématique de la normalisation des données en apprentissage automatique.

8.4.1 Les extensions de SVMRFE

Parmi les approches de sélection d'attribut à base de SVM, on trouve d'abord plusieurs extensions de SVMRFE. Certaines 1) cherchent à en améliorer la rapidité de l'algorithme en introduisant de l'intelligence pour décider du nombre d'attributs à éliminer à chaque itération RFE [33, 28], et d'autres 2) proposent des critères alternatifs à celui de SVMRFE pour décider des attributs à éliminer à chaque récursion RFE [132, 99] (logRatio-SVMRFE que nous proposons dans la section 10.2 pourrait être vu comme appartenant à la deuxième catégorie). Concernant ce dernier point, rappelons que SVMRFE cherche à éliminer les attributs qui influencent le moins les prédictions des modèles SVM, de manière à ce que le modèle SVM appris après sélection des attributs soit le plus proche possible de celui appris avant sélection. Cette stratégie est discutable car elle sous-entends que le modèle de départ (appris sur l'ensemble des données) est le modèle idéal que l'on cherche à approcher. [132] et [99] remettent en cause cette stratégie et ils proposent de s'intéresser à d'autres objectifs que nous allons décrire maintenant.

Le premier, [132], propose d'éliminer à chaque itération RFE l'attribut qui éloigne le plus possible les données de la classe positive de celles de la classe négative. Pour cela, plutôt que d'ordonner les attributs simplement en fonction du coefficient $|w_i|$ appris par le SVM linéaire comme le fait SVMRFE, [132] les ordonne en fonction du score $s_i = w_i(m_i^+ - m_i^-)$ (où m_i^+ et m_i^- sont les moyennes des valeurs de l'attribut i sur l'ensemble des instances de la classe positive et négative). Cela lui permet de prendre en compte la distance entre le centroïde de chaque classe dans le critère d'ordonnancement.

Quand au second auteur, [99], il cherche plutôt à éliminer l'attribut qui minimise l'erreur de généralisation comme le ferait un algorithme de sélection de type *wrapper* [42]. Par contre, pour éviter les calculs coûteux qui sont nécessaires aux nombreuses phases d'apprentissage des *wrapper*, il approche le résultat en minimisant en réalité une borne supérieure $\mathcal{L}$ sur l'erreur de généralisation *leave one out* qu'il est possible d'obtenir pour les modèles SVM [22]. A chaque itération RFE c'est donc l'attribut dont la suppression minimise la valeur de la borne $\mathcal{L}$ qui est éliminé (ordre zéro). [99] examine également l'élimination de l'attribut qui influence le moins la valeur de la borne supérieure (ordre 1). Pour effectivement déterminer quel est l'attribut qui influence le moins la borne $\mathcal{L}$, il introduit des facteurs multiplicatifs virtuels des attributs (ρ), et calcule le gradient de $\mathcal{L}$ selon ces facteurs. Ainsi le noyau $K(\mathbf{x}, \mathbf{z})$ utilisé dans le modèle SVM devient $K(\rho \otimes \mathbf{x}, \rho \otimes \mathbf{z})$ et le score d'ordonnancement d'un attribut k est donnée par $\left|\frac{\partial \mathcal{L}}{\partial \rho_k}\right|$.

8.4.2 Introduction de facteurs multiplicatifs des attributs

Parallèlement aux travaux de [99], l'introduction de facteurs multiplicatifs dans la fonction noyau pour résoudre le problème de la sélection d'attributs est aussi étudié dans [125] et [14]. Ces auteurs inspectent le problème de la recherche du pré-traitement $\mathbf{x} \rightarrow \rho \otimes \mathbf{x}$ qui minimisent les bornes de l'erreur *leave-one-*

out d'un SVM non linéaire. Avec pour le premier $\rho \in \{0,1\}^n$, et pour le second $\rho \in \Re^n$. Pour cela ils définissent le noyau paramétré $K_\rho(\mathbf{x},\mathbf{y}) = K(\rho \otimes \mathbf{x}_i, \rho \otimes \mathbf{x}_j)$, et cherchent le paramètre ρ qui minimisent la borne *leave-one-out*. Le vecteur ρ solution de ce problème sert à établir un ordonnancement des attributs selon leur pertinence pour le modèle.

Le principal défaut de ces approches c'est qu'elles ne considèrent que ρ pour la sélection des attributs, or une partie de l'information sur la pertinence des attributs est contenue dans les modèles SVM. L'autre point faible c'est que, comme SVMRFE, elles sont très dépendantes de la normalisation des attributs. Enfin la recherche de la solution optimum est faite par descente de gradient et risque d'être piégé dans des minima locaux, et elle ne sont pas applicable au cas du noyau logRatio pour lequel la borne *leave-one-out* est constante pour toutes les valeurs de ρ.

8.4.3 Minimisation de la norme L0 de la marge

Afin de résoudre le problème de la sélection d'attributs à partir de SVM, [124] s'attaque à un problème d'optimisation dans lequel il cherche à minimiser la norme L0 du vecteur $\mathbf{w}$ qui indique la direction de l'hyperplan de marge maximal dans un SVM linéaire. L'avantage d'utiliser la norme L0 est que cela fait explicitement apparaître les contraintes sur le nombre de variables à sélectionner. Plus concrètement, les auteurs cherchent à résoudre le problème d'optimisation (8.9) dont l'objectif est de maximiser la Lp-norme de la marge de l'hyperplan séparateur, avec la contrainte $||\mathbf{w}||_0 \leq r$ qui impose que le nombre de composantes différentes de 0 dans $\mathbf{w}$ soit inférieure à r (un paramètre utilisateur).

$$\begin{cases} \text{minimise}_{\mathbf{w}\in\Re^n} : & ||\mathbf{w}||_p \\ \text{subject to :} & y_i(\langle \mathbf{w}.\mathbf{x}_i \rangle + b) \geq 1 \text{ and } ||\mathbf{w}||_0 \leq r \end{cases} \tag{8.9}$$

Le vecteur $\mathbf{w}$ solution de ce problème contient au maximum r composantes non nulles qui déterminent les attributs à sélectionner. La difficulté est que la résolution du problème est NP-complet, mais l'auteur montre qu'une solution approchée peut être obtenue avec l'algorithme 9 dont le concept est un hybride entre SVMRFE et la recherche des facteurs multiplicatifs idéaux à appliquer aux attributs. Effectivement, on peut constater que cet algorithme revient à

Algorithm 9 Algorithme d'approximation du problème (8.9)

1: $w \leftarrow$ Entraîner un SVM linaire standard
2: Multiplier les variables d'entrée par les composantes du vecteur $\mathbf{w}$
3: Répéter 1 et 2 jusqu'à convergence

appliquer itérativement des corrections sur la normalisation des attributs de l'espace initial jusqu'à obtenir une certaine stabilité du modèle SVM que l'on apprend sur les données. Nous voyons donc déjà bien en quoi cela peu le lier à notre objectif.

8.5 Normalisation et apprentissage automatique

Les problèmes relatifs à la normalisation des données que nous avons évoqué pour SVMRFE (section 8.3.1) sont en réalité universels et nombreux sont les algorithmes d'apprentissage sensibles à la normalisation. Par exemple, la normalisation des attributs influence : 1) une analyse en composante principales (PCA) car elle modifie la variance des attributs ; 2) l'algorithme des voisins proches (IBk) car elle modifie les distances euclidiennes entre les instances ; 3) et aussi les SVM linéaires (voir figure 8.1). Mais, même si une bonne normalisation des attributs à tendance à améliorer les performances de ces algorithmes, les paramètres optimaux restent difficile à régler. Pour illustrer cela, considérons un exemple simple : l'algorithme de classification par voisins proches pour déterminer l'espèce d'une fleur en fonction de la longueur et la largeur de ses pétales. Dans ce problème, il faut déterminer si une fleur dont les pétales ont une dimension de 4cm x 1cm est plus proche d'une fleur de type A dont les pétales mesurent 5cm x 1cm ou d'une autre fleur de type B avec des pétales de 4cm x 2cm ? La réponse dépend des poids attribués à chaque information.

La figure 8.2 illustre cela, en montrant que les poids attribués aux attributs jouent un rôle important dans la prise de décision des classificateurs. Sur cette figure, on peut observer que la surface de décision obtenues avec l'algorithme de classification par le plus proche voisin est différente selon la normalisation effectuée pour les attributs. On constate que les régions noires, qui montrent les différences de prédiction selon la normalisation effectuée, sont beaucoup plus étendues aux endroits où il y a peu d'instances. Cet constatation paraît naturelle dans la mesure ou cela reflète une certaine incertitude des modèles de prédiction aux endroits où nous n'avons pas d'observations. D'ailleurs, dans le cas extrême où l'on aurait un nombre infini d'instances qui couvrent la totalité de l'espace, les incertitudes s'atténueraient et nous n'observerions aucune différence entre les deux cas de normalisation. L'ennui c'est que nous possédons généralement un petit nombre d'instance et que la disparité de ces instances à tendance à s'amplifier dans les données de grande dimension. C'est un phénomène connu sous le nom de *curse of dimensionality* [45, 11] et qui est à l'origine des problèmes que rencontre les algorithmes de classification dans les données de grande dimension. Comme les instances sont donc très dispersées dans des données de grande dimension, on peut s'attendre à ce que les surfaces de prédictions changent beaucoup selon la normalisation effectuée ce qui donne d'autant plus d'importance à ce processus.

Plus problématique encore, il est difficile de définir la normalisation idéale d'un jeu de donnée, car si il est clair qu'elle doit maximiser les performances des algorithmes d'apprentissage que l'on y applique, on ne peut pas se limiter à cette définition. En effet, nous avons par exemple vu que les classificateurs linéaires comme les SVM-linéaire associent à chaque attribut un poids en cherchant à minimiser l'erreur de classification. On pourrait donc être tenter d'utiliser ces poids pour normaliser les attributs des jeux de donnée, mais en réalité ceux-ci renferment deux informations : il contiennent à la fois un facteur correctif correspondant à la normalisation optimale de l'attribut, et à la fois un facteur

qui indique l'importance de l'attribut dans la prise de décision du classificateur. Il est important de dissocier les deux si on souhaite interpréter correctement les modèles.

L'approche classique pour corriger ce problème de normalisation des attributs est de standardiser les attributs. Cela consiste à appliquer un rapport d'homothétie égal à l'inverse de la déviation standard de manière à ce qu'elle soit égale à 1 après standardisation. Cette méthode ne garanti pas que les pondérations appliquées soient optimales, par contre elle à l'avantage d'aboutir à une représentation invariable des données. Effectivement, la déviation standard d'un attribut est une quantité proportionnelle aux valeurs de l'attribut, si bien que la normalisation d'un attribut par une valeur α induit un changement de la déviation standard par le même rapport α. Le quotient des deux reste donc inchangé car les facteurs s'annule. La représentation ainsi obtenu est donc invariable à la normalisation des attributs.

Il faut cependant remarquer que la standardisation des attributs est liée à la normalisation des instances car cette dernière influence la valeur de la déviation standard des attributs. Avant de standardiser les attributs, il est donc recommander de normaliser les instances au préalable. De plus cet ordre est préférable car les instances d'un jeu de données sont supposés indépendantes les unes des autres, alors que ça n'est pas le cas des attributs. L'ennui c'est que ceci rend la normalisation des attributs dépendante des aléa de la normalisation des instances. Dans la suite, le noyau logRatio proposé résout ces soucis en offrant une représentation des données insensible à la fois à la normalisation des instances et à la normalisation des attributs. Pour cela, logRatio emploi une représentation des données insensible à ces deux opérations, si bien que le résultats de la fonction noyau reste inchangé quelque soit les facteurs de normalisation appliquées à la fois aux instances et aux attributs. Avant de présenter notre travail à ce sujet, la sous-section 8.5.1 présente l'algorithme TSP qui possède des similitudes avec notre travail dans la mesure où il exploite également une représentation particulière des données qui le rend insensible à la normalisation.

8.5.1 TSP

TSP (top-scoring pair(s), [34]) est un algorithme de sélection d'attributs qui vise des objectifs similaires à ceux que nous cherchons à atteindre dans notre travail et de nombreux parallèles peuvent être fait entre les deux méthodes : TSP fourni une approche pour la sélection d'attribut facile à interpréter, insensible à certaines variations des données, et focalisant l'apprentissage sur les interactions attribut-attribut. TSP fonctionne en comparant deux à deux les attributs à la recherche des paires d'attributs (i, j) pour lesquels on observe $x_i < x_j$ dans les instances de la classe positive et $x_i > x_j$ dans les instances de la classe négative. En fonction du nombre d'instance satisfaisant cette règle, un score est attribué à chaque paire d'attribut qui permet d'établir un classement des plus pertinentes. La particularité de TSP est donc qu'il produit un ordonnancement de paires d'attributs en fonction de leur pouvoir discriminatoire, plutôt qu'un ordonnancement simple des attributs, et donc un attribut peut apparaître plusieurs fois

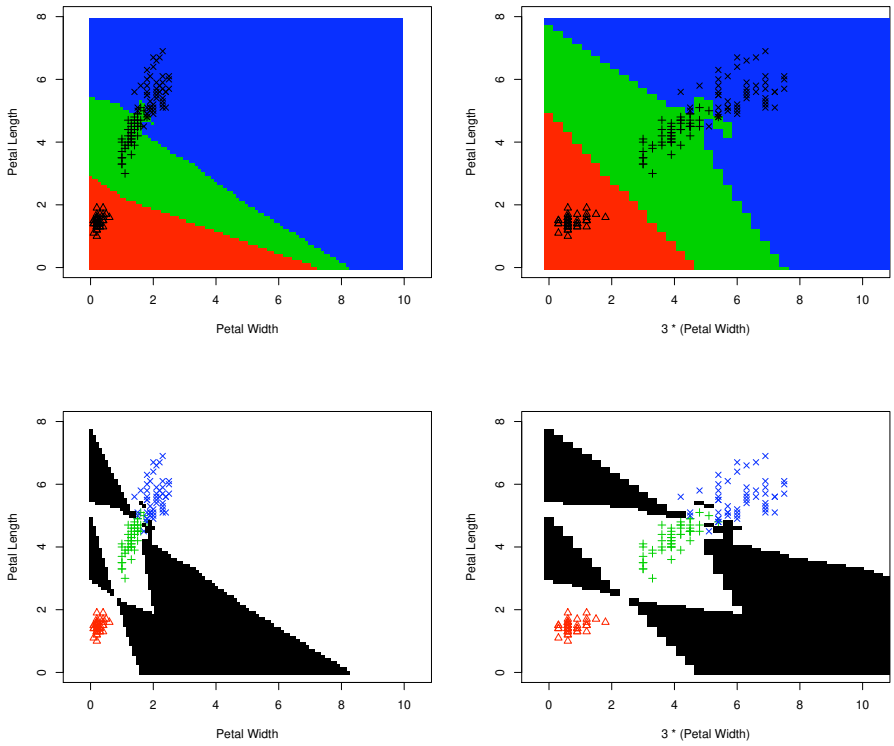


FIGURE 8.2 – Différence entre les surfaces de décision rendues par l'algorithme de classification par le voisin le plus proche sur le jeu de donnée Iris selon la normalisation des attributs : les deux graphes de gauche montrent le résultat sur les données originales sans normalisation, les deux graphes de droite montrent le résultat lorsque l'attribut "Petal Width" est normalisé par un facteur de valeur 3. Les graphiques du haut montrent la surface de décision apprise dans les deux cas, les graphiques du bas montre leurs différences en noir : chaque instance qui serait située dans les régions noires serait prédites différemment selon la normalisation de "Petal Width". On constate que les régions noires qui montrent les différences sont beaucoup plus étendues aux endroits où il y a peu d'instances, mais l'exemple se limite à des données bi-dimensionnelles et la disparité des instances à tendance à augmenté avec la dimension des données.

parmi les paires sélectionnées. A la différence, l'algorithme logRatio-SVMRFE que nous proposons dans la section 10.2 se distingue de TSP en établissant un ordonnancement simple des attributs même si lui aussi se base sur des informations entre paires d'attributs.

Un des avantages de TSP est donc qu'il est insensible à certaines transformations des données souvent observées dans les puces à ADN, ce qui en fait un algorithme adapté pour l'extraction des biomarqueurs dans ces données et par extension il pourrait être appliquer avantageusement à la spectrométrie de masse dans la mesure où les problèmes rencontrés dans ces deux domaines sont similaires. En particulier, TSP est insensible à toute translation, et à toute normalisation des instances par des rapports d'homothétie positifs. En effet, comme ces transformations préservent les ordres, les comparaisons entre les valeurs ne sont pas altérées. Plus précisément, si une instance $\mathbf{x}$ subit un rapport d'homothétie positif $\alpha > 0$, et une translation β, alors on a $\alpha x_i + \beta < \alpha x_j + \beta$ ssi $x_i < x_j$, c'est à dire que le résultat de la comparaison est insensible à ces transformations, c'est pourquoi TSP qui emploi seulement cet opérateur l'est aussi. L'utilité de cette invariabilité de l'algorithme TSP aux deux transformations ci-dessus est importante pour limiter les fluctuations des résultats lorsque des expériences identiques sont répétées plusieurs fois par les expérimentateurs.

Par contre, TSP à plusieurs inconvénients. Tout d'abord, il est sensible à la normalisation des attributs, or celle-ci est difficile à réaliser si les instances ne sont pas normalisées correctement (ce que cherche à éviter la méthode). Pour illustrer ce problème, considérons l'exemple de la classification des fleurs d'Iris en fonction des dimensions de leurs pétales. La largeur des pétales étant toujours inférieure à la longueur, TSP à des difficultés à tirer des conclusions sur ces données sans normalisation des attributs pour corriger les valeurs. Un autre inconvénient de TSP, et qu'il a une complexité quadratique avec le nombre d'attributs car il doit effectuer leur comparaisons deux à deux, or nous travaillons avec des problèmes de forte dimension et cela peux constituer un problème. Enfin, nous allons aussi voir dans les paragraphes qui suivent que TSP à l'inconvénient de se rapprocher d'avantage des méthodes de sélection d'attribut univariées que des méthodes multivariées ce qui l'empêche de prendre en compte des interactions complexes entre les attributs. A titre de comparaison, logRatio-SVMRFE ne souffre pas de tous ces inconvénients : il est multivarié, insensible à la normalisation des attributs, et sa complexité varie en $\mathcal{O}(n \log n)$ avec le nombre d'attributs. De plus, logRatio-SVMRFE à aussi les mêmes avantages que TSP si ce n'est qu'il est sensible à la translation des instances.

TSP équivalent à une projection dans un espace de caractéristiques binaires

Il est intéressant de remarquer que l'on peut obtenir le même résultat que celui produit par TSP en générant un espace de n^2 caractéristiques binaires obtenues en comparant les n attributs du jeu de données initial. Dans cet espace, chaque caractéristique d'une instance $\mathbf{x}$ transformée valant 0 ou 1 selon que la comparaison $x_i < x_j$ est vérifié ou non (ceci pour tous les couples ij possibles).

TSP se réduit alors à un simple algorithme de d'ordonnancement univarié qui doit extraire les caractéristiques binaires qui concordent le plus avec l'étiquette des instances. Ceci est facile à réalisé avec un test individuel de chacune des caractéristiques avec l'étiquette de classe, par exemple avec un test du χ^2. Cette vision de l'algorithme montre que TSP se rapproche d'avantage d'un algorithme de sélection d'attributs univarié qu'un algorithme multivarié, ce qui représente une faiblesse puisque cela le rend incapable de saisir des interactions complexes entre les attributs. A l'inverse l'algorithme logRatio-SVMRFE (section 10.2) établit un ordonnancement des attributs initiaux grâce à des informations sur les paires d'attributs, mais il considère toutes les paires à la fois, et en ce sens logRatio-SVMRFE est réellement multivarié ce qui lui permet de saisir des relations complexes entre les attributs comme les résultats obtenus sur Iris le montrerons (section 10.3).

Notre travail présente des similitude avec la représentation binaire de TSP : nous essayons également de combiner les attributs deux à deux pour former n^2 caractéristiques dans un espace projeté mais à la différence de TSP, nous n'exploiterons pas l'opérateur de comparaison $x_i < x_j$ qui délivre une valeur binaire. A la place, nous calculerons le logarithme du ratio $\log(\frac{x_i}{x_j})$ qui renvoie une valeur réelle dont le signe dépend du résultat de cette comparaison. Dans les deux cas, c'est le choix d'une représentation relative de l'information (représentation des valeurs des instances par rapport aux autres valeurs de cette instance) qui permet d'obtenir des algorithmes insensibles à certaines transformations des données grâce à une description plus "qualitative" de l'information. En effet, alors que classiquement on utilise une représentation absolue des données (par exemple, pour décrire une instance de fleur on utilise des attributs qui représentent les dimensions de ses pétales : 6cm de long et 2cm de large) TSP et logRatio préfèrent se placer à un niveau plus descriptif de l'information (à la place des dimensions des pétales de la fleur, TSP considère simplement que la largeur est inférieure à la longueur et logRatio préfère quand à lui considérer le fait que la longueur des pétales est 3 fois supérieure à leur largeur) ce qui les rend plus robuste aux variations des données.

8.6 Bilan de l'introduction à la sélection d'attributs

Dans ce chapitre, nous avons mis en avant les problématiques qui se posent pour extraire les bio-marqueurs de la matrice d'expression. Nous avons vu que la sélection d'attributs offrait des outils pour résoudre ces problèmes en particulier grâce aux approches SVM qui montrent des facultés pour traiter les données de grande dimension. Néanmoins, nous avons aussi identifié des lacunes dans ces méthodes qui sont gênantes pour traiter les données de la biologie et qui ouvrent des pistes à étudier. Nous avons notamment retenu les aspects liées à la normalisation des données, à la redondance d'information dans les attributs et à la l'interprétation des modèles. Ces aspects sont importants pour les données

biologique, et certains algorithme comme TSP conçu pour ce type de données introduisent des idées intéressantes pour tenter d'aborder ces problèmes.

Dans la suite de cette partie, nous proposons d'étudier un algorithme de sélection d'attribut qui essai mettre ensemble les avantages que présentent les SVM pour traiter les données de grande dimension avec les avantages que présente TSP pour s'abstraire des variabilités des données. Comme nous l'avons déjà signalé, notre approche est basée sur la définition du noyau logRatio qui offre une représentation des données avantageuse : les propriétés et l'interprétation qui découle de l'apprentissage d'un modèle SVM avec ce noyau nous permettront en effet de réaliser une sélection d'attributs par élimination récursive de manière similaire à SVMRFE. La méthode résout dans une certaine mesure les problèmes liées à normalisation des données en offrant un algorithme qui 1) n'est sensible ni à la normalisation des instances, ni à la normalisation des attributs dans l'espace initial ; 2) offre une interprétation intéressante des modèles ; 3) met l'accent sur les interactions attribut-attribut et essai de traiter efficacement la redondance d'information. Nous présentons notre méthode dans le chapitre 10, mais avant cela, nous nous intéressons dans le chapitre 9 à la stabilité des méthodes de sélection d'attributs qui est un critère d'évaluation de leur qualité.

Chapitre 9

Stabilité de la sélection d'attributs

En raison de la banalisation des algorithmes de sélection d'attributs, pour traiter les données de grande dimension qui prolifèrent, il est devenu important de caractériser ces algorithmes et de définir des critères de qualité pour les comparer. Les deux critères de qualité majeurs que l'on considère généralement sont le nombre d'attributs sélectionnés par ces méthodes, et la performance de classification qu'il est possible d'obtenir avec ces attributs. A elles deux, ces informations indiquent les facultés de l'algorithme à concentrer l'information sur les données qui on un pouvoir discriminant. Néanmoins, ces informations sont insuffisantes pour caractériser les algorithmes de sélection d'attributs car il est possible que des sous-ensembles d'attributs distincts et de même taille fournissent des informations similaires. Par exemple, en présence de redondances, le remplacement des attributs d'un sous-ensemble par des attributs redondants peut fournir un autre sous-ensembles d'attributs contenants des informations très proches du premier. Comme en général les algorithmes de sélection d'attributs doivent faire des choix lorsque plusieurs alternatives comme celles-ci s'offrent à eux, se pose la question de la stabilité de ces choix. Ici, nous allons voir comment exploiter ce critère de stabilité pour caractériser les algorithmes de sélection d'attributs.

Dans un contexte de recherche des bio-marqueurs en spectrométrie de masse, la stabilité des résultats est de plus un critère important car il est difficile pour un biologiste d'accepter un résultat expérimental qu'il ne peut répéter. Comme la dimension importante des données et le bruit qu'elles contiennent augmentent les risques d'instabilité des algorithmes de sélection d'attributs, il est important de mesurer ces instabilités pour garantir que l'algorithme employé est stable et pour estimer la reproductibilité de l'expérience si l'on considère de nouvelles données. Ces considérations ont justement été à l'origine de notre travail sur le thème de la stabilité des algorithmes de sélection d'attributs [89, 61]. Dans ces articles, nous fournissons une analyse détaillée de la stabilité d'un algorithme

de sélection d'attributs à base de SVM sur des données de spectrométrie de masse, et en particulier, on y étudie la stabilité des algorithmes aux variations de données engendrées par une validation croisée. En revanche la généralisation de ce concept de stabilité comme caractéristique des algorithmes de sélection d'attributs n'y apparaît pas explicitement. Ce n'est que dans nos travaux suivants, [58, 59], où nous définissons un cadre de travail général et rigoureux pour l'évaluation de la stabilité des algorithmes de sélection d'attributs. Dans nos expériences du chapitre 10, nous exploitons ce travail afin de comparer différentes méthode de sélection d'attributs. Dans la suite, nous allons voir ce dont il s'agit plus précisément. La section 9.1 commence par présenter le travail initié dans [59] sur la stabilité des méthodes de sélection d'attributs, et nous abordons la littérature qui en découle dans la section 9.2.

9.1 La stabilité

La notion de stabilité pour les algorithmes de classification (et de régression) est liée à la décomposition biais/variance de l'erreur [30]. Le biais mesure l'écart moyen entre d'une part les prédictions faites par les modèles de classification générés par l'algorithme analysé et d'autre part les prédictions idéales ; la variance mesure l'instabilité entre les prédictions des modèles et leur prédiction moyenne. L'estimation du biais et de la variance s'obtient en générant plusieurs modèles de classification avec un même algorithme sur des instances d'apprentissage différentes, puis en comparant leurs prédictions pour une même instance.

En revanche, la stabilité des algorithmes de sélection d'attributs est une notion relativement peu abordé dans la littérature jusqu'à récemment [59, 25, 66, 104, 110, 72]. A l'inverse des méthodes de classification, ces algorithmes ne génèrent pas de prédictions pour les instances, et on ne peut pas appliquer la décomposition biais/variance de l'erreur. Par contre, nous proposons dans [59] de s'inspirer des méthodes d'estimation biais/variance pour étudier la stabilité des algorithmes de sélection d'attributs. Cela consiste à comparer les différents résultats générés par un même algorithme de sélection d'attributs entraîné avec des instances d'apprentissage différentes, mais cette fois, il ne s'agit pas de comparer des prédictions sur des instances, mais de comparer les attributs sélectionnés par les algorithmes. Dans la suite nous allons voir ce qui est proposé dans [59] pour quantifier la stabilité des algorithmes de sélection d'attribut (section 9.1.1), établir un résultat de référence théorique pour la stabilité (section 9.1.2) et discuter de quelques résultats expérimentaux (section 9.1.3).

9.1.1 Mesure de la stabilité

Pour quantifier la stabilité des algorithmes de sélection d'attribut, nous introduisons dans [59] trois mesures permettant de comparer deux à deux les résultats rendus par les algorithmes sur différents jeux de données. Chaque mesure s'applique à un type d'algorithme particulier selon que le résultat de celui-ci consiste (1) en une pondération de chaque attribut par une valeur w_i qui reflète

l'importance de l'attribut i pour l'algorithme ; (2) en un ordonnancement de tous les attributs (*ranking*) selon leur importance ; (3) en un sous-ensemble d'attribut identifié comme important par l'algorithme. Remarquez qu'il y a une hiérarchie dans ces types car il est toujours possible de déduire un ordonnancement à partir d'une pondération des attributs, et on peut toujours déduire un sous-ensemble d'attributs à partir des attributs les mieux placés dans un ordonnancement. La suite de cette section présente ces trois mesures.

La première mesure, S_W, s'applique aux algorithmes de sélection d'attributs dont le résultat est une pondération de tous les attributs. Pour cette mesure, la similitude entre deux vecteurs de poids ($\mathbf{w}$ et $\mathbf{w}'$), obtenus avec un même algorithme sur deux ensembles d'instances différents, est calculée au moyen de la corrélation de Pearson :

$$S_W(w, w') = \frac{\sum_i (w_i - \mu_w)(w'_i - \mu_{w'})}{\sqrt{\sum_i (w_i - \mu_w)^2 \sum_i (w'_i - \mu_{w'})^2}}.$$

S_W renvoie une valeur réelle comprise dans [-1,+1] : +1 lorsque les poids sont parfaitement corrélés, 0 lorsqu'il n'y a pas de corrélation et -1 lorsqu'il sont anti-corrélés. La mesure S_W fournit ainsi un résultat global qui prend en compte l'ensemble des poids de tous les attributs. Néanmoins, elle est assez sensible aux valeurs extrêmes, et il suffit qu'un attribut ait un poids extrême dans $\mathbf{w}$ et $\mathbf{w}'$ pour influencer significativement la valeur de S_W même si les autres poids indiquent une influence contraire. De plus, la corrélation de Pearson capture seulement les corrélations linéaires entre les poids, et il peut exister d'autre type de relation.

La deuxième mesure s'applique aux algorithmes de sélection d'attributs dont le résultat est un ordonnancement des attributs. On utilise le coefficient de corrélation de Spearman pour calculer la similitude entre deux ordonnancements $\mathbf{r}$ et $\mathbf{r}'$ (des vecteurs de taille m qui indiquent le rang de chacun des m attributs du jeu de donnée) :

$$S_R(r, r') = 1 - 6 \sum_i^m \frac{(r_i - r'_i)^2}{m(m^2 - 1)}.$$

S_R renvoie une valeur rationnelle comprise dans [-1,+1] : +1 lorsque les poids sont parfaitement corrélés, 0 lorsqu'il n'y a pas de corrélation et -1 lorsqu'il sont anti-corrélés. Comme S_W, S_R fournit un résultat global qui prend en compte l'ensemble des attributs. Par contre,S_R est moins sensible aux valeurs extrêmes que S_W car les rangs r_i sont limités à l'intervalle [1..m], alors que les poids w_i ne sont pas bornés dans S_W. De plus une mesure S_R basée sur un ordonnancement des poids est souvent plus robuste qu'une mesure S_W basée sur les poids eux-même car les hypothèses sont moins fortes (par exemple sur la linéarité des poids w_i).

Enfin, la troisième mesure, S_S, est très générale puisqu'elle mesure la similitude entre deux sous-ensembles d'attributs, et elle peut être appliquée à tous les types d'algorithme. Définissons S comme l'ensemble des attributs du jeu de

donnée, et s, s' deux sous-ensembles de S obtenus par un même algorithme de sélection d'attribut entraîné avec des d'instances d'apprentissage différentes (voir figure 9.1). Le calcul de S_S est basé sur la notion de distance entre deux

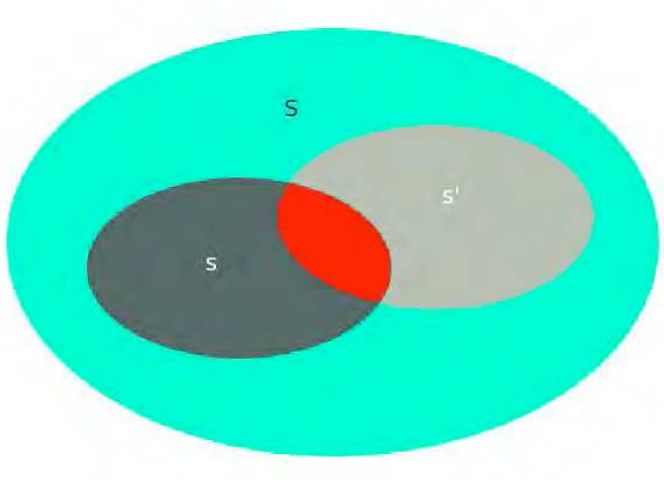


FIGURE 9.1 – Illustration de deux sous-ensembles d'attributs s et s' sélectionné dans l'ensemble S des attributs.

ensembles, et on utilise une adaptation de la distance de Tanimoto dans sa définition [30] :

$$S_S(s, s') = 1 - \frac{|s| + |s'| - 2|s \cap s'|}{|s| + |s'| - |\ s \cap s'|}.$$

S_S renvoi une valeur rationnelle comprise dans [0..1] : 0 lorsque les sous-ensembles s et s' sont disjoints, et 1 lorsqu'ils sont identiques. Cette mesure se concentre seulement sur l'intersection entre les sous-ensembles s et s', et ignore tout ordonnancement des attributs à l'intérieur de ces sous-ensembles comme à l'extérieur. Ainsi, si par exemple un algorithme de sélection d'attribut modifie l'importance qu'il associe aux attributs mais que les même attributs restent sélectionnés au final, alors cette différence ne sera pas reflété dans S_S mais seulement dans S_R et S_W (si on peut appliquer ces mesures). Mentionnons néanmoins qu'il est possible de dresser un profil d'évolution de S_S en fonction du nombre d'attribut dans s et s' qui fournit alors une vision globale de la stabilité (voir figure 9.2, section 9.1.2 pour un exemple). Une autre manière de percevoir cette différence entre S_S et les deux autres mesures (S_R et S_W), c'est de remarquer que S_R et S_W permettent de mesurer le degré d'anti-corrélation entre deux résultats (les valeurs des mesures sont d'autant plus négative que les résultats sont anti-corrélés) alors que S_S ne le permet pas (c'est pourquoi sa valeur ne peut être que positive). Enfin, lorsque les sous-ensembles s et s' ont le même cardinal ($|s| = |s'|$), ce qui est souvent le cas car c'est en général un paramètre de l'algorithme, il est préférable de mesurer le pourcentage d'élément que s et s' ont

en commun ($\frac{|s \cap s'|}{|s|}$) plutôt que de mesurer S_S. Effectivement, les deux mesures délivrent une information similaire, mais le pourcentage d'élément en commun à l'avantage d'être plus facile à interpréter que S_S. Par contre si s et s' n'ont pas le même nombre d'élément, il faut appliquer S_S qui tient compte des différences de taille entre s et s' (voir discussion dans la section 9.1.2).

Les trois mesures définies ci-dessus fournissent des valeurs normalisées (bornées entre [-1..1] ou [0..1] selon la mesure) pour estimer la similitude entre deux résultats d'une sélection d'attributs. Comme il est difficile de tirer des conclusions fiables sur la stabilité des résultats à partir d'une seul comparaison, nous proposons de générer plusieurs ensembles d'instances par tirage aléatoire ou par "validation croisée" pour y entraîner l'algorithme étudié et obtenir plusieurs résultats différents. Par exemple, dans le cas de la validation croisée à 10 couches (*10 folds cross validation*, les mesures sont appliquées aux 10 résultats pris deux à deux et la stabilité de l'algorithme est définie comme la moyenne des 45 valeurs obtenues. Signalons que la validation croisée à 10 couches génère des jeux de données qui ont mutuellement 11% d'instances qui diffèrent, et on mesure donc seulement l'impact de ces 11% sur la stabilité de l'algorithme de sélection d'attribut. Idéalement, il faudrait que les jeux de données soient complètement distincts, mais cela nécessiterait beaucoup d'instances d'apprentissage au départ.

Nous n'avons mentionné dans cette section que les mesures de stabilité que nous avons introduit dans [59]. De nombreuses variantes ont été proposées dans la littérature, notamment pour corriger un effet indésirable qui se caractérisent par un accroissement naturel de la stabilité lorsque de nombreux attributs sont sélectionnés (voir section 9.1.2). Parmi les mesures proposées mentionnons l'index de stabilité de [66], la mesure de consistance de [110], l'index de Jaccard utilisé par [104]. Dans l'état de l'art de la section 9.2, nous donnons plus de détails sur ces articles et la section 9.1.2 va nous permettre de revenir sur cet effet indésirable en étudiant plus en détail le comportement de la mesure de Tanimoto.

9.1.2 Stabilité d'une sélection d'attributs aléatoire

Cette section propose d'établir un résultat de référence théorique pour la stabilité des algorithmes de sélection d'attributs. Pour cela, on va étudier la mesure de Tanimoto que peut espérer obtenir un algorithme de sélection d'attributs qui choisit des attributs aléatoirement avec une distribution uniforme. Définissons S comme l'ensemble des attributs du jeu de donnée, et s, s' deux sous-ensembles aléatoires de S que l'on a obtenu avec cet algorithme aléatoire (voir figure 9.1). De plus, notons respectivement N, n, n' et k le cardinal des ensembles S, s, s' et $(s \cap s')$. Dans un premier temps intéressons-nous à la probabilité que les deux sous-ensemble s et s' aient k éléments en commun, c'est à dire à la distribution $P(k|n, n', N)$. Comme tous les sous-ensembles ont la même probabilité d'apparition, cette distribution est facile à calculer : il suffit de compter le nombre de couple (s, s') qui ont k éléments en commun et de le diviser par le nombre de couples (s, s') possibles. Notons le premier nombre θ_k et le second Θ. Le calcul

de θ_k (équation 9.1) se fait en multipliant : (1) le nombre de façons de choisir les n éléments de s dans l'ensemble S ; (2) par le nombre de façons de choisir les $(n' - k)$ éléments de $(s - s')$ dans l'ensemble $(S - s)$; (3) par le nombre de façons de choisir les k éléments de $(s' \cap s)$ dans l'ensemble s. Pour Θ, il suffit de sommer les θ_k pour tout les k possibles, ou plus simplement de multiplier le nombre de manière de choisir les n attributs de s, par le nombre de manière de choisir les n' attributs de s' (équation 9.2). On obtient donc :

$$\theta_k = \binom{N}{n}\binom{N-n}{n'-k}\binom{n}{k} \tag{9.1}$$

$$\Theta = \sum_{i=0} \theta_i = \binom{N}{n} \sum_{i=0} \binom{N-n}{n'-i}\binom{n}{i} = \binom{N}{n}\binom{N}{n'} \tag{9.2}$$

$$P(k|n, n', N) = \frac{\theta_k}{\Theta} = \frac{\binom{N-n}{n'-k}\binom{n}{k}}{\binom{N}{n'}}. \tag{9.3}$$

Si l'on pose n'=n, la distribution P(k — n,n',N) que l'on obtient est identique à l'équation donnée par [66], et dans ce cas l'auteur fait remarquer que l'espérance de k est égale à $E(k|n, N) = \frac{n^2}{N}$. Rappelons que si n'=n, nous recommandons d'utiliser le pourcentage de recouvrement entre les ensembles pour mesurer la stabilité car c'est une mesure plus facile à interpréter que la mesure de Tanimoto. On voit ici que cette l'espérance de recouvrement pour un algorithme de sélection d'attribut aléatoire est égale à $E(k/n|n, N) = \frac{n}{N}$, c'est à dire que c'est une fonction linéaire de la taille des ensembles sélectionnés qui est égale à 1 lorsque n'=n=N. Dans le chapitre 10 (section 10.4.3), nous utilisons ce résultat comme référence pour la stabilité.

Le calcul de la mesure de Tanimoto entre les deux ensembles s et s' fait intervenir les nombres n,n' et k (équation 9.4). Comme on connaît maintenant la distribution $P(k|n, n', N)$ pour deux ensembles aléatoires s et s' pris dans S (équation 9.3), on peut calculer l'espérance de la mesure de Tanimoto entre s et s' avec l'équation 9.5. La figure 9.2 montre un tracé de $E(S_S(s, s')|n' = n, N = 100)$ en fonction de n. Pour ce tracé, le jeu de donnée contient un total de N=100 attributs, et l'algorithme extrait deux sous-ensembles d'attributs s et s' avec le même nombre n d'éléments. Remarquez cette fois que la courbe n'est pas une droite comme c'est le cas pour l'espérance de recouvrement $E(k|n, N)$.

$$S_S(s, s') = S_S(n, n', k) = 1 - \frac{|s| + |s'| - 2|s \cap s'|}{|s| + |s'| - |\, s \cap s'|} = 1 - \frac{n + n' - 2k}{n + n' - k} \tag{9.4}$$

$$E(S_S(s, s')|n, n', N) = \sum_k P(k|n, n', N) * S_S(n, n', k) \tag{9.5}$$

Il est en tout cas intéressant de remarquer sur la figure 9.2 que la stabilité augmente non-linéairement avec le nombre d'attribut sélectionné. Cela signifie que plus un algorithme sélectionne de petits sous-ensembles d'attributs, et plus il a de chance d'être instable car la probabilité que deux sous-ensembles aléatoire contiennent des éléments identiques est faible. A l'inverse les algorithmes qui sélectionne beaucoup d'attributs de l'ensemble sont très stable car

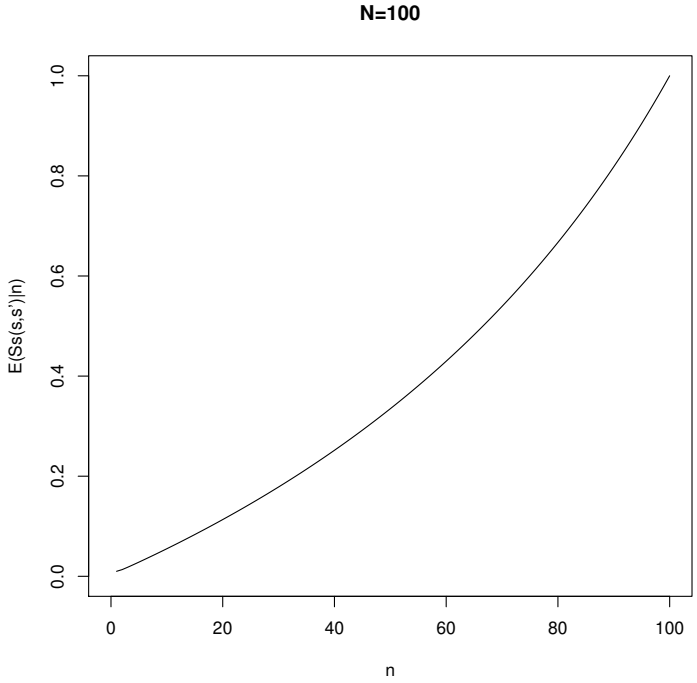


FIGURE 9.2 – Evolution de l'espérance de la distance de Tanimoto lorsque deux sous-ensembles s et s', de n éléments chacun, sont choisis aléatoirement dans un ensemble initial S de $N = 100$ éléments.

il y a de fortes chances que les sous-ensembles se chevauchent. En conséquence, lors de la comparaison des stabilités, il faut tenir compte de cet effet, et faire attention aux conclusions que l'on tire de la comparaison de deux algorithmes qui sélectionnent des sous-ensembles d'attributs de taille différente. Dans les travaux de [25, 66, 110], les auteurs tentent de proposer des mesures de stabilité qui limite cet effet. [25] adopte une mesure proche de S_S et introduit une pénalité qui augmente linéairement avec le nombre d'attributs sélectionnés pour tenir compte de cet effet. [66] discute également de ce problème de la mesure S_S, et apporte une solution théorique élégante basée sur un calcul similaire a celui située ci-dessus pour le contourner. Enfin, [110] propose même une adaptation directe de S_S pour la rendre robuste à cet effet. Nous revenons sur ces travaux dans la section 9.2 En tous cas, ce phénomène illustre bien que le critère de stabilité ne doit pas être considéré seul, mais en complément des autres caractéristiques des algorithmes de sélection d'attributs, c'est à dire le nombre d'attributs sélectionné et la performance que l'on peut en extraire. Signalons enfin que dans [59], nous avons déterminé une courbe proche de celle de la fi-

gure 9.2 par simulation, et le résultat théorique que nous avons établi ci-dessus complète ce travail.

9.1.3 Expériences sur la stabilité

Dans [59], nous menons de nombreuses expériences pour caractériser le comportement de plusieurs algorithmes de sélection d'attributs en nous concentrant sur la comparaison de leur stabilité. Cette section, reprend les résultats majeurs de cette étude. Six méthodes de sélection d'attributs (Information Gain, Chi-Square, Symmetrical Uncertainty, ReliefF, SVMRFE, SVMONE) sont étudiées sur onze jeux de données de grande dimension (table 9.1). Les algorithmes de sélection d'attributs viennent de l'outil Weka [126] et sont choisis pour leur diversité : Information Gain (IG), Chi-Square (CHI), Symmetrical Uncertainty (SYM) sont des approches univariées de pondération des attributs qui discrétisent les valeurs continues ; ReliefF est une approche multivariée qui pondèrent les attributs en fonction de leur aptitude à séparer les instances voisines de classe différente (pour chaque instance K=10 voisins sont considérés) ; SVMRFE, déjà décrit dans la section 8.3, utilise des modèles SVM-linéaire pour pondérer les attributs en en éliminant 10% à chaque itération ; enfin, SVMONE utilise un seul modèle SVM-linéaire pour pondérer les attributs [1]. Les jeux de données définissent tous un problème de classification binaire et ils couvrent trois domaines différents : la spectrométrie de masse, les puces à ADN, le text-mining. Les mesures de stabilité présentées plus haut, dans la section 9.1.1, sont utilisées pour comparer la stabilité des différentes méthodes. Les estimations de stabilité à l'intérieur de chaque ensemble d'entraînement sont obtenues avec une validation croisée interne à dix couches, et les valeurs que l'on rapporte sont les moyennes de S_W, S_R, S_S sur les dix couches de la validation croisée externe.

Résultat de la comparaison des stabilités

Les résultats de l'estimation de stabilité de chaque algorithme de sélection d'attributs, pour chaque jeu de données, et selon chacune des trois mesures sont fournis table 9.2. On remarque tout d'abord que les différentes mesures de stabilité peuvent rendre des résultats en désaccords. Par exemple, la stabilité de IG pour le jeu de donnée "colon" sont respectivement de 0.7606, 0.1138, 0.4865 pour S_W, S_R et S_S, c'est à dire que selon S_R la stabilité est plutôt faible (proche de 0), alors qu'elle relativement élevée selon S_W (0.7606 alors que le maximum est 1), et moyenne selon S_S (proche de 0.5). Ces différences s'expliquent par la manière dont opèrent les trois mesures : S_S ne considère que les (10) attributs avec les meilleurs scores, elle reflète donc une information partielle de la stabilité ; S_R se base sur les rangs attribués à chaque attribut, mais il accorde la même importance à des erreurs situées en haut de classement qu'a des erreurs situées en bas de classement alors que les erreurs en haut de classement sont clairement plus problématiques ; enfin, S_W se base sur les poids

1. Les implémentation de SVMRFE et de SVMONE que nous utilisons ici utilise des modèles C-SVM avec un paramètre C fixé à C=0.5.

dataset	domaine	classe 1	# classe 1	classe 2	# classe 2	# attributs
ovarian	MS	normal	91	diseased	162	824
prostate	MS	normal	253	diseased	69	2200
stroke	MS	normal	101	diseased	107	4928
leukemia	MA	ALL	47	AML	25	7131
nervous	MA	survival	21	failure	39	7131
colon	MA	normal	22	tumor	40	2000
alt	Text	relevant	1425	not	2732	2112
disease	Text	relevant	631	not	2606	2376
function	Text	relevant	818	not	3089	2708
structure	Text	relevant	927	not	2621	2368
subcell	Text	relevant	1502	not	6475	4031

TABLE 9.1 – Description des jeux de données. La colonne "domaine" indique le domaine d'où est issue le jeu de donnée. MS : Mass Spectrometry ; MA : Micro-Array ; Text : Text-mining.

attribués à chaque attribut est très sensible aux valeurs extrêmes. D'une manière générale : S_W indique des stabilités assez élevées pour toutes les méthodes ; S_R indique des valeurs assez faibles pour les algorithmes univariés (IG, CHI, SYM) ; et S_S à des valeurs diversifiées. Ces observations suggèrent la présence fréquente de valeurs extrêmes dans les scores qui biaisent l'estimation de S_W, et la présence d'instabilités en bas des classements rendus par IG, CHI, SYM qui biaisent la mesure S_R sans affecter la mesure S_S. Ce phénomène est en réalité accentué par un "effet de bord" de ces trois méthodes qui vient de la discrétisation des valeurs qu'il est nécessaire d'appliquer aux attributs pour calculer leurs scores. De nombreux attributs se voient attribuer un score extrême de zéro qui les place tous à égalité en fin de classement. Ce score extrême biaise la mesure S_W, et les scores identiques qui sont attribués à de nombreux attributs biaise la mesure S_R qui les ordonne aléatoirement. Les méthodes de sélections d'attributs multivariés n'ont pas ce problème. Au final c'est donc S_S qui délivre l'information la plus fiable.

Si l'on se penche maintenant sur la comparaison des différents algorithmes, on observe tout d'abord des résultats très proches pour les trois algorithmes univariés IG, CHI et SYM. Les trois méthodes obtiennent effectivement des valeurs presque identiques avec les trois mesures pour tous les jeux de données. Selon S_S, qui est la mesure la moins biaisée pour ces méthodes, il apparaît que ces approches sont très stables pour traiter les données textuelles. Pour les données de protéomiques et de génomiques, c'est par contre RELIEF qui montre en moyenne la meilleure stabilité. A la fois en terme de S_S et de S_R, sa stabilité moyenne est systématiquement supérieure à celle des autres méthodes, ce qui traduit non seulement une bonne stabilité des attributs avec les meilleures scores, mais aussi une bonne stabilité globale. Concernant les méthodes à base de SVM, elles obtiennent une stabilité globale S_R correcte souvent autour de

dataset	IG			CHI			SYM		
	$\overline{S_W}$	$\overline{S_R}$	$\overline{S_S}$	$\overline{S_W}$	$\overline{S_R}$	$\overline{S_S}$	$\overline{S_W}$	$\overline{S_R}$	$\overline{S_S}$
ovarian	0.9553	0.9105	0.4948	0.9560	0.9089	0.5614	0.9512	0.9068	0.5038
prostate	0.8247	0.4070	0.4080	0.8249	0.4011	0.4212	0.8196	0.4039	0.4125
stroke	0.8387	0.2042	0.1847	0.8434	0.2032	0.2152	0.8250	0.2023	0.2187
avg	0.8729	0.5072	0.3625	0.8747	0.5044	0.3992	0.8652	0.5043	0.3783
leukemia	0.8507	0.2492	0.7392	0.8467	0.2477	0.7557	0.8397	0.2479	0.7897
nervous	0.4652	0.0177	0.2320	0.4693	0.0166	0.2385	0.4652	0.0182	0.2436
colon	0.7606	0.1138	0.4865	0.7587	0.1157	0.5024	0.7504	0.1159	0.5069
avg	0.6921	0.1269	0.4859	0.6915	0.1266	0.4988	0.6851	0.1273	0.5134
alt	0.9983	0.1254	0.9632	0.9985	0.1267	0.9640	0.9966	0.1268	0.9037
disease	0.9231	0.0900	0.7589	0.9267	0.0853	0.7470	0.9002	0.0853	0.7405
function	0.9151	0.1045	0.8055	0.9255	0.1058	0.7974	0.8972	0.1041	0.7491
structure	0.9771	0.1636	0.8855	0.9807	0.1641	0.8853	0.9643	0.1629	0.8485
subcell	0.9854	0.1136	0.8990	0.9873	0.1139	0.8599	0.9795	0.1135	0.8487
avg	0.9594	0.1194	0.8624	0.9637	0.1191	0.8507	0.9476	0.1185	0.8181

dataset	RELIEF			SVMONE			SMVRFE		
	$\overline{S_W}$	$\overline{S_R}$	$\overline{S_S}$	$\overline{S_W}$	$\overline{S_R}$	$\overline{S_S}$	$\overline{S_W}$	$\overline{S_R}$	$\overline{S_S}$
ovarian	0.9697	0.9537	0.7296	0.9379	0.8476	0.5965	NA	0.8386	0.4680
prostate	0.9572	0.9399	0.5529	0.8685	0.7389	0.5243	NA	0.7323	0.4484
stroke	0.8806	0.8230	0.3410	0.8174	0.7032	0.2721	NA	0.6971	0.1678
avg	0.9358	0.9055	0.5411	0.8746	0.7633	0.4175	NA	0.7560	0.3614
leukemia	0.9157	0.8675	0.5793	0.8757	0.7655	0.4878	NA	0.7632	0.2678
nervous	0.8078	0.7839	0.2873	0.8099	0.6751	0.4568	NA	0.6728	0.1065
colon	0.9063	0.8363	0.6931	0.7782	0.6818	0.3512	NA	0.6799	0.2392
avg	0.8766	0.8292	0.5199	0.8213	0.7074	0.4319	NA	0.7053	0.2045
alt	0.8278	0.6945	0.6908	0.9889	0.7468	0.6676	NA	0.7307	0.6517
disease	0.8270	0.6892	0.5277	0.8366	0.7033	0.5269	-	-	-
function	0.6764	0.6417	0.5444	0.7826	0.7169	0.3255	-	-	-
structure	0.7636	0.6678	0.5072	0.8543	0.7156	0.5325	-	-	-
subcell	0.8016	0.6881	0.6794	0.9328	0.7345	0.7663	-	-	-
avg	0.7793	0.6763	0.5899	0.8790	0.7234	0.5638			

TABLE 9.2 – Résultat de stabilité pour les différentes mesure. $\overline{S_S}$ est calculée à partir de sous-ensembles de 10 attributs.

0.7, et une stabilité S_S plus dépendante des jeux de données de l'ordre de 0.5, ce qui laisse une marge pour des améliorations possibles. Enfin, la comparaison des méthodes à base de SVM, SVMONE et SVMRFE, révèle un comportement global (S_R) assez proche mais SVMONE est souvent plus stable que SVMRFE en terme de S_S. Cette observation s'explique par la répétition des apprentissages SVM qu'il est nécessaire de faire dans SVMRFE. Des erreurs successives se cumulent dans les apprentissages et résultent en une instabilité plus élevée des attributs situées en tête de classement.

Profil de stabilité

La mesure S_S qui fournit l'information la plus utile pour caractériser la stabilité ne fournit cependant qu'une information partielle. Effectivement, on doit fixer le nombre n d'attributs sélectionné pour pouvoir calculer sa valeur. Par exemple la valeur S_S donnée dans la table 9.2 est calculée avec les $n = 10$ attributs ayants obtenus les meilleurs scores. Il est possible que pour un n différent les valeurs de S_S changent beaucoup. Pour avoir une image plus précise de la stabilité des méthodes selon S_S pour différentes valeurs de n, nous avons donc établie le profil de chacune d'elles comme nous l'avons fait pour la sélection d'attributs aléatoire dans la section 9.1.2. Les figures 9.3 et 9.4 donnent ces profils pour les différents jeux de données, et ils sont justement comparés à celui de la sélection d'attributs aléatoire sur chaque tracé.

Les trois méthodes univariées ont des profils de stabilité très proches qui se superposent l'un sur l'autre. A l'exception du jeu de données ovarian, tous ces profils montrent une stabilité acceptable pour les petites valeurs de n qui se dégrade au fur et à mesure que n augmente pour se rapprocher du profil aléatoire. Comme expliqué plus haut, ce comportement est dut au nombreux scores zéros attribués par ces méthodes. Le point à partir duquel les profils commencent à plonger vers le profil aléatoire correspond au n où l'on commence à inclure des attributs avec le score zéro : comme ceux-ci sont inclus dans un ordre aléatoire, la stabilité se dégrade petit à petit.

Les deux méthodes à base de SVM ont également des profils assez proches l'une de l'autre. Pour les petites valeurs de n, il peut y avoir des différences assez importantes, mais plus n augmente et plus leurs profils se rapprochent. La raison à cela est la même que celle donnée plus haut : plus on se déplace vers les n grand, et plus on inclut d'attributs dans les sous-ensembles s et s' qui ont été éliminés tôt dans les itérations de SVMRFE. Ainsi plus n augment, plus les sous-ensembles d'attributs sélectionnés par SVMRFE ressemble à ceux sélectionnés par SVMONE. Lorsque n est égale à 90% les sous-ensembles sont exactement identiques car on élimine 10% des attributs à chaque itération de SVMRFE.

La stabilité des méthodes univariées pour les petites valeurs de n est particulièrement remarquable sur les données textuelles où le profil est toujours au dessus des autres méthodes. Mais c'est RELIEF qui montre les profils les plus stable, d'ailleurs pour tous les jeux de données, il y a toujours une valeur de n pour laquelle RELIEF est le plus stable de tous. Pour les données tex-

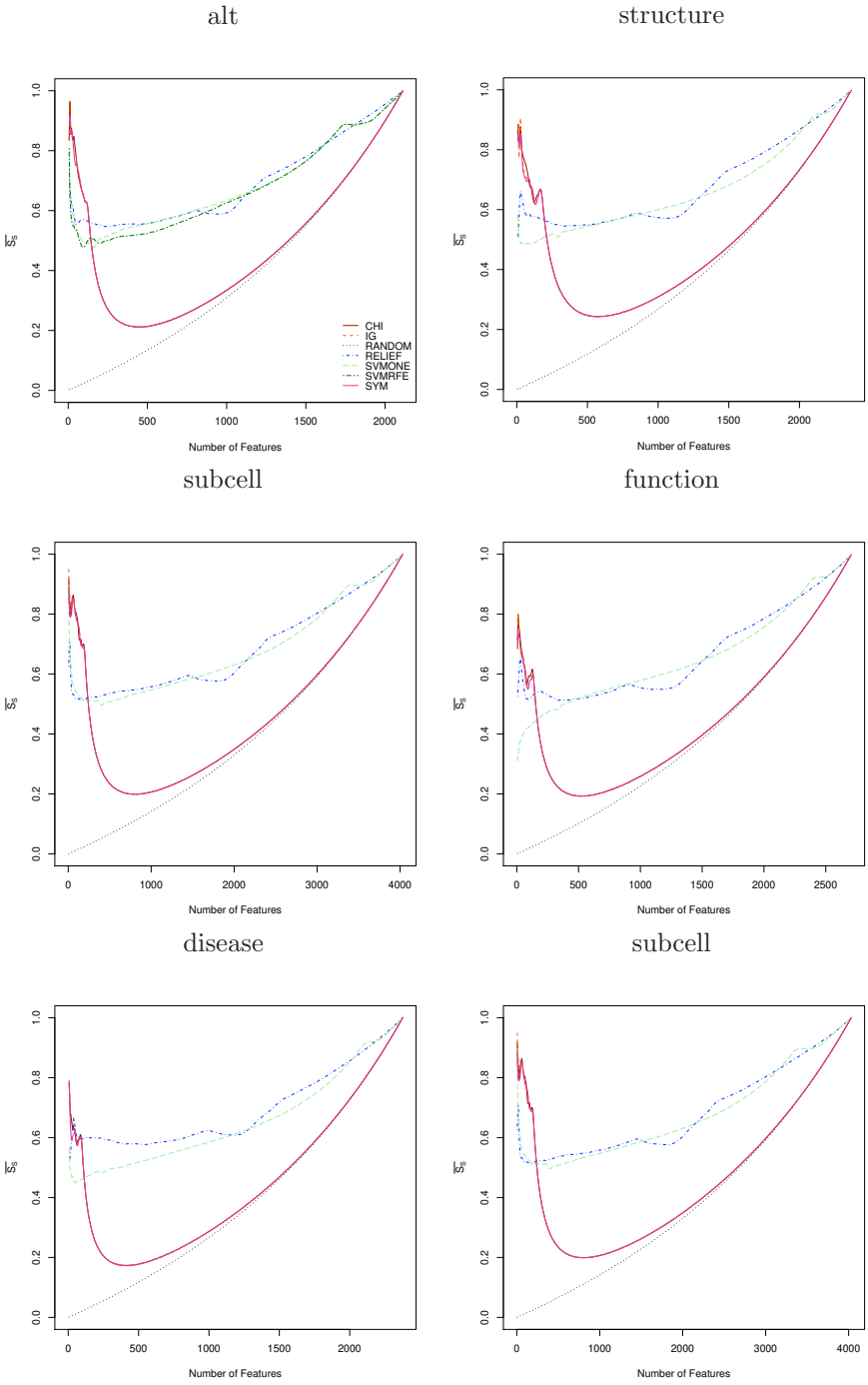


FIGURE 9.3 – Profil S_S des données textuelles.

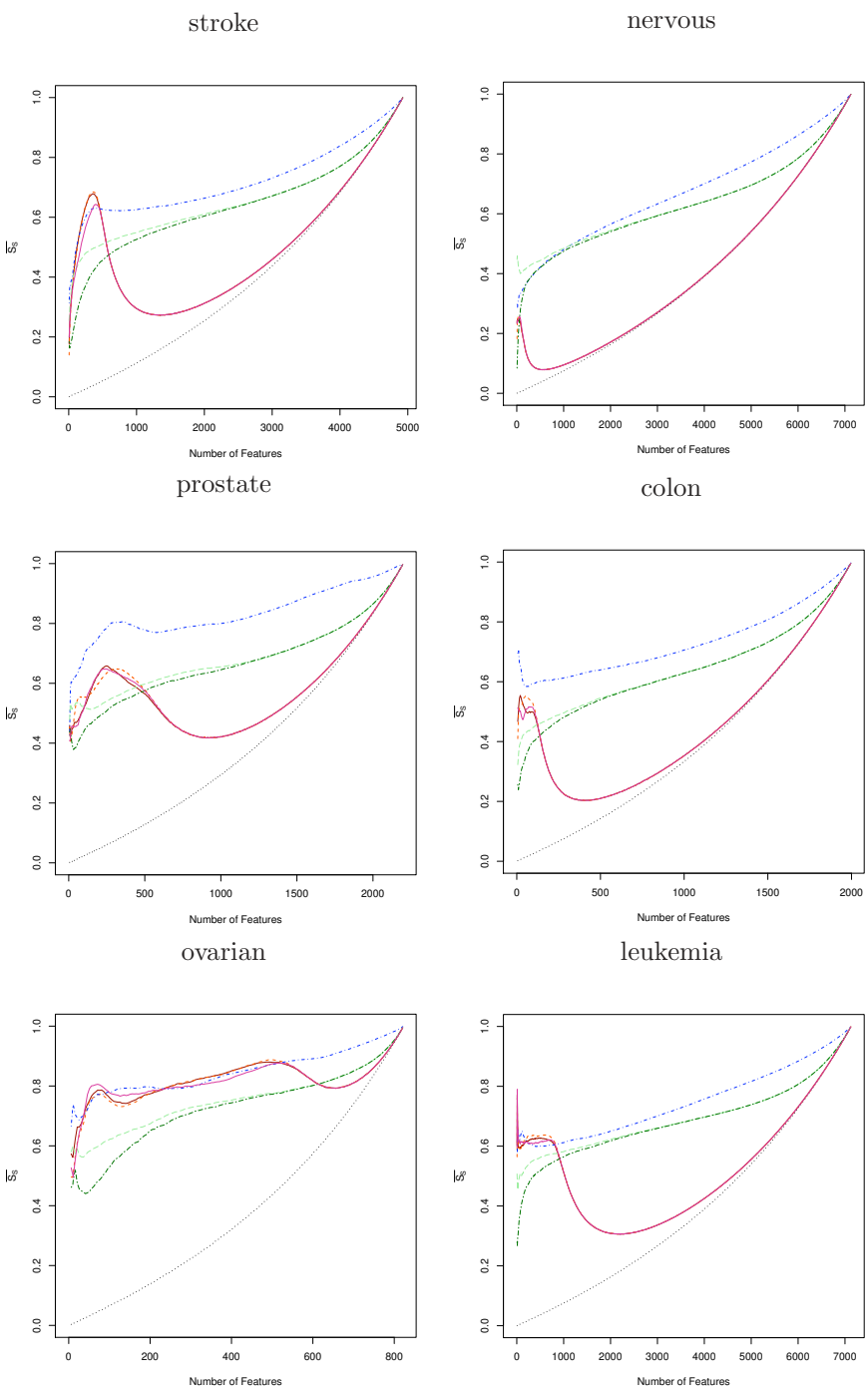


FIGURE 9.4 – Profil S_S des données protéomiques (colonne de gauche) et génomiques (colonne de droite).

tuelles cela est vrai pour des n assez grands (car pour les n petits ceux sont les méthodes univariées qui dominent), mais pour les données de génomique et de protéomique, les profils de RELIEF sont très stables pour toutes les valeurs de n. Enfin les méthodes à base de SVM ont des profils assez proches de ceux de RELIEF pour les données textuelles, mais pour les autres types de données, les profils sont nettement moins stable que ceux de RELIEF. La forme des profils qui se dessinent pour ces méthodes montrent dans de nombreux cas une forme en S : une forte chute de stabilité pour les n assez grand, suivie d'une large période de dégradation lente qui se termine par une chute plus rapide. Cette forme traduit les incertitudes de l'approche à placer les attributs aux extrémités du classement.

Comparaison des performances de classification

Jusqu'ici, les différentes méthodes de sélection d'attributs ont été comparées sur la base de leur stabilité en laissant de coté les performances de classification qu'il est possible d'obtenir avec les attributs qu'elles sélectionnent. Pourtant, ce critère est essentielle pour comparer correctement les méthodes, car il est possible que celles qui se sont montrées les plus stables ne délivrent pas les meilleures performances. Dans cette section les analyses de stabilité sont complétés par une analyse de performance de classification.

Afin d'évaluer la performance de classification, l'erreur de classification d'un SVM-linaire [2] est estimée par validation croisée en limitant l'apprentissage aux n attributs sélectionnées par chaque méthode. Plus exactement, la validation croisée découpe les données en dix ensembles d'entraînement et dix ensembles de test. Sur chaque ensemble d'entraînement, la méthode de sélection d'attribut délivre n attributs qui sont utilisés pour apprendre un modèle SVM. L'ensemble de test correspondant, sert à estimer les performances de classification du modèle. Plusieurs valeurs de n, allant de 10 à 50 avec un pas de 10, sont considérés dans les analyses. Aussi, comme les méthodes univariées ont un comportement similaire, seul les performances de IG sont reportées. Pour un n donné, les performance de classification des quatre algorithmes restants sont comparées deux à deux avec un test statistique de McNemar qui vérifie avec une probabilité de 5% que la différence entre les erreurs de classification est statistiquement significative. Selon le résultat des testes entre deux méthodes A et B, une note est attribuée à chaque méthode de la manière suivante : 1 point pour la méthode A si son erreur de classification est statistiquement plus faible que celle de la méthode B ; 0.5 points si les erreurs de A et B ne sont pas statistiquement différentes ; et 0 points si A à une erreur de classification inférieure à celle de B. La note finale pour une méthode est donnée par la somme de des poids. Cette note est au maximum de 3.0 si elle sur-performe les trois autres méthode, et au minimum de 0.0 si elle sous-performe les trois autres. L'ensemble des résultats de ces expériences sont données dans les tables 9.3, 9.4 et 9.5.

Les résultats montrent tout d'abord un certains nombre de cas ou les per-

2. On utilise ici un C-SVM avec le paramètre C fixé à C=0.5.

Stroke				
n	IG	RELIEF	SVMONE	SVMRFE
10	1.5-32.22-0.1847	1.5-30.29-0.3410	1.0-37.02-0.2721	*2.0-26.45-0.1678*
20	1.0-31.73-0.2612	1.0-28.85-0.3670	1.0-35.10-0.3101	*3.0-21.64-0.1679*
30	*1.5-27.89-0.2944*	*1.5-27.41-0.3830*	*1.5-28.37-0.3390*	*1.5-23.56-0.1802*
40	*1.5-29.81-0.3261*	*1.5-25.97-0.3887*	*1.5-25.00-0.3583*	*1.5-25.49-0.1886*
50	*1.5-27.89-0.3576*	*1.5-28.37-0.4013*	*1.5-26.45-0.3801*	*1.5-25.49-0.1997*
Ovarian				
n	IG	RELIEF	SVMONE	SVMRFE
10	1.0-10.28-0.4948	1.0-10.28-0.7296	1.0-07.11-0.5965	*3.0-01.19-0.4680*
20	1.0-05.53-0.6111	1.0-05.93-0.6933	1.5-03.95-0.5897	*2.5-01.19-0.4749*
30	0.0-04.74-0.6567	*2.0-01.58-0.6966*	*2.0-01.19-0.5631*	*2.0-00.40-0.4498*
40	0.5-03.16-0.7011	1.5-01.58-0.7080	*2.0-00.40-0.5682*	*2.0-00.40-0.4401*
50	*1.5-02.77-0.7496*	*1.5-01.58-0.7368*	*1.5-00.40-0.5825*	*1.5-00.40-0.4473*
Prostate				
n	IG	RELIEF	SVMONE	SVMRFE
10	1.0-18.64-0.4073	1.0-18.95-0.5842	1.0-18.02-0.5308	*3.0-13.05-0.4417*
20	1.0-17.71-0.4299	1.0-17.09-0.6044	1.0-16.46-0.5131	*3.0-11.50-0.4006*
30	1.0-16.46-0.4639	1.0-15.84-0.6170	1.0-14.91-0.5193	*3.0-10.87-0.3786*
40	1.0-16.15-0.5044	1.0-14.91-0.6214	1.0-13.36-0.5280	*3.0-09.01-0.3848*
50	1.0-14.60-0.5374	1.0-13.36-0.6304	1.0-13.05-0.5343	*3.0-09.32-0.3890*

TABLE 9.3 – Performance sur les données de protéomique. Chaque triplet $x-y-z$ indique la note de la méthode x, son erreur de classification y, et la valeur de la mesure de stabilité $\overline{S_S}$ z. Les algorithmes de sélection d'attributs avec les meilleures notes sont indiqués en *italique*.

leukemia				
n	IG	RELIEF	SVMONE	SVMRFE
10	*1.5-05.55-0.7392*	*1.5-06.94-0.5793*	*1.5-05.55-0.4878*	*1.5-05.55-0.2678*
20	*1.5-05.55-0.6570*	*1.5-04.16-0.6553*	*1.5-04.16-0.4544*	*1.5-01.38-0.2979*
30	*1.5-05.55-0.6294*	*1.5-02.77-0.6338*	*1.5-02.77-0.4681*	*1.5-01.38-0.3108*
40	*1.5-05.55-0.5958*	*1.5-02.77-0.6360*	*1.5-02.77-0.4852*	*1.5-01.38-0.3336*
50	*1.5-04.16-0.5938*	*1.5-02.77-0.6255*	*1.5-02.77-0.4921*	*1.5-01.38-0.3526*
nervous				
n	IG	RELIEF	SVMONE	SVMRFE
10	*1.5-40.00-0.2320*	*1.5-30.00-0.2873*	*1.5-35.00-0.4568*	*1.5-36.66-0.1065*
20	*1.5-38.33-0.2491*	*1.5-30.00-0.2973*	*1.5-30.00-0.4469*	*1.5-40.00-0.1498*
30	*1.5-35.00-0.2506*	*1.5-36.66-0.3124*	*1.5-36.66-0.4288*	*1.5-28.33-0.1909*
40	1.5-35.00-0.2488	1.0-40.00-0.3158	1.0-36.66-0.4174	*2.5-23.33-0.2129*
50	1.5-31.66-0.2501	1.0-41.66-0.3283	1.0-38.33-0.4127	*2.5-23.33-0.2349*
colon				
n	IG	RELIEF	SVMONE	SVMRFE
10	*1.5-17.74-0.4856*	*1.5-16.12-0.6931*	*1.5-25.80-0.3512*	*1.5-16.12-0.2392*
20	*1.5-17.74-0.5143*	*1.5-14.51-0.6530*	*1.5-22.58-0.3950*	*1.5-19.35-0.2810*
30	*1.5-14.51-0.5224*	*1.5-14.51-0.6174*	*1.5-16.12-0.4121*	*1.5-19.35-0.3115*
40	1.5-14.51-0.5459	*2-12.90-0.5937*	1.5-16.12-0.4229	1.0-22.58-0.3261
50	1.5-14.51-0.5519	*2-12.90-0.5837*	1.5-14.51-0.4311	1.0-22.58-0.3470

TABLE 9.4 – Results Performance sur les données de génomique. Chaque triplet $x - y - z$ indique la note de la méthode x, son erreur de classification y, et la valeur de la mesure de stabilité $\overline{S_S}$ z. Les algorithmes de sélection d'attributs avec les meilleures notes sont indiqués en *italique*.

alt			
n	IG	RELIEF	SVMONE
10	*1.0-10.77-0.9623*	*1.0-10.87-0.6908*	*1.0-10.89-0.6676*
20	*1.5-10.56-0.8631*	1.0-10.84-0.6187	0.5-11.01-0.6191
30	*1.0-10.68-0.8209*	*1.0-10.80-0.6103*	*1.0-10.58-0.5832*
40	*1.0-10.58-0.7996*	*1.0-10.65-0.5719*	*1.0-10.65-0.5549*
50	*1.0-10.46-0.7733*	*1.0-10.58-0.5545*	*1.0-10.51-0.5311*
disease			
n	IG	RELIEF	SVMONE
10	*1.0-19.67-0.7589*	*1.0-19.46-0.5277*	*1.0-19.64-0.5269*
20	*1.0-19.74-0.6778*	*1.0-19.36-0.5766*	*1.0-19.43-0.4790*
30	*1.0-19.83-0.6282*	*1.0-19.24-0.6282*	*1.0-19.80-0.4691*
40	0.5-19.98-0.6124	*2.0-19.09-0.6670*	0.5-19.92-0.4536
50	*1.0-19.52-0.6133*	*1.0-19.02-0.6250*	*1.0-19.77-0.4489*
function			
n	IG	RELIEF	SVMONE
10	*1.5-20.24-0.8055*	0.5-20.93-0.5444	1.0-20.37-0.3255
20	*1.0-20.29-0.7129*	*1.0-20.93-0.6304*	*1.0-20.45-0.3694*
30	*1.0-20.27-0.6824*	*1.0-20.93-0.6566*	*1.0-20.47-0.3789*
40	*2.0-20.06-0.6824*	0.5-20.93-0.6032	0.5-20.68-0.3916
50	*2.0-19.98-0.6649*	0.5-20.93-0.5732	0.5-20.68-0.3996
structure			
n	IG	RELIEF	SVMONE
10	*2.0-21.02-0.8855*	0.5-22.66-0.5072	0.5-21.84-0.5325
20	*2.0-19.78-0.8141*	0.0-22.26-0.6296	1.0-20.77-0.4977
30	*2.0-19.39-0.8481*	0.0-21.95-0.6486	1.0-20.40-0.4853
40	*2.0-19.05-0.7718*	0.5-20.54-0.6288	0.5-19.80-0.4889
50	*1.5-19.08-0.7331*	0.0-20.71-0.6062	*1.5-19.50-0.4866*
subcell			
n	IG	RELIEF	SVMONE
10	*1.5-15.97-0.8980*	0.0-16.72-0.6794	*1.5-15.80-0.7663*
20	*1.5-15.84-0.8646*	0.0-16.47-0.7110	*1.5-15.43-0.6749*
30	*1.5-15.48-0.8039*	0.0-16.49-0.5878	*1.5-15.19-0.6349*
40	*1.5-14.86-0.8117*	0.0-16.48-0.5438	*1.5-15.03-0.6044*
50	*1.5-14.64-0.8460*	0.0-16.44-0.5339	*1.5-14.93-0.5773*

TABLE 9.5 – Performance sur les données textuelles. Chaque triplet $x - y - z$ indique la note de la méthode x, son erreur de classification y, et la valeur de la mesure de stabilité $\overline{S_S}$ z. Les algorithmes de sélection d'attributs avec les meilleures notes sont indiqués en *italique*.

formances de classification des algorithmes sont identiques, mais les valeurs de stabilité sont différentes. Par exemple pour le jeu de donnée *stroke* et $n = 30-50$, les quatres algorithmes ont les mêmes notes de classification, mais RELIEF est le plus stable de tous. Sa mesure de stabilité dans ce cas est même plus de deux fois plus grande que celle de SVMRFE. Pour *leukemia* aussi, c'est encore la mesure de stabilité qui fait la différence entre les méthodes : toutes les méthodes ont la même note de classification, mais SVMRFE est parfois plus de deux fois moins stable que IG et RELIEF. Ces observations montre bien l'utilité d'utiliser la stabilité pour caractériser les méthodes de classification, mais retenez néanmoins que certaines des meilleures performances de classification sont obtenus avec des méthodes très instables comme SVMRFE.

Il y a aussi de nombreux cas où les performances de classification sont plus élevées pour certains algorithmes que pour d'autres. Il faut encore remarquer à ce sujet que les performances de classification les plus élevées ne correspondent pas forcément à une stabilité élevée des méthodes de sélection d'attributs. En particulier, SVMRFE obtient souvent la note de performance maximum alors que sa stabilité est la plus faible de toutes. Par exemple c'est le cas sur la plupart des jeux de données de protéomique : *prostate*, *stroke* avec $n = 10-20$, *ovarian* avec $n = 10-40$; et sur *nervous* 40-50. Une raison qui pourrait expliquer les bonnes performances et l'instabilité de SVMRFE, mais qui reste à vérifier, c'est la présence d'attributs redondants dans les données. En effet, si il y a des attributs redondants, il y a potentiellement plusieurs sous-ensembles d'attributs qui mènent à des performances élevées, et les instabilités apparaissent si parmi ces sous-ensembles SVMRFE en choisi un différent à chaque fois. Une autre explication pourrait être notre mauvaise interprétation des modèles SVM-linéaire : en effet si notre interprétation de ces modèles est incorrecte, la sélection des attributs qui résulte de cette interprétation est faussé, et on observe une instabilité des modèles. Remarquez à ce sujet que la manière dont les attributs sont normalisés dans SVMRFE peut jouer un rôle important dans l'interprétation des modèles.

Enfin mentionnons les cas où la stabilité et les performances de classification sont élevées toutes les deux ensembles. C'est régulièrement le cas de IG dans les données textuelles (e.g. *function* avec $n = 40-50$, *structure* avec $n = 10-40$), et parfois de RELIEF (e.g. *colon* avec $n = 40-50$ *disease* avec $n = 40$). Dans ces cas là, ce sont clairement les méthodes à utiliser. Mentionnons également que SVMRFE bat régulièrement SVMONE en terme de performance de classification sur les données de protéomique, et aussi sur *nervous* $n = 40-50$. Cela montre le gain de performance apporté par SVMRFE par rapport à SVMONE, mais cela se fait au détriment de la stabilité car SVMRFE est plus instable que SVMONE.

Bilan des expériences de stabilité

Les expériences de stabilité décrites dans cette section ont montré l'utilité de mesurer la stabilité des algorithmes de sélection d'attributs. Elle fournie en particulier un critère objectif pour privilégier une méthode par rapport à une autre en l'absence de différences sur les performances de classification. Dans

la section qui suit, nous allons voir les directions qui ont été suivies dans la littérature en relation avec ce travail.

9.2 Etat de l'art

Nous abordons dans cette section différents travaux qui sont en relation avec celui sur la stabilité des algorithmes de sélection d'attributs que nous avons développé dans la section précédente. Nous mentionnons en particulier plusieurs travaux qui ont découlé de ce travail.

L'approche de [25] vis à vis de la stabilité des algorithmes de sélection d'attributs est intéressante. Pour étudier des données de puces à ADN en vu de l'extraction de bio-marqueurs potentiels, il propose un système pour choisir parmi un univers d'algorithme de sélection d'attributs et d'algorithmes de classification un couple qui montre à la fois de bonnes performances de classification, et une bonne stabilité. Le système évalue le couple d'algorithme ce qui garanti que la méthode de sélection d'attribut et le classificateur sont compatibles, c'est à dire que le classificateur arrive à identifier des motifs dans les attributs sélectionnées par l'algorithme de classification. La stabilité est prise en compte non seulement du point de vu de la classification, mais aussi du point de vu de la sélection des attributs. Pour chacun des trois aspects : performance de classification, stabilité de la classification, stabilité de la sélection d'attribut ; une mesure est définie et une somme pondérée des trois valeurs obtenues permet de représenter la qualité globale du couple. Dans cette somme, la moitié du poids est attribuée à la stabilité de la sélection d'attributs ce qui montre l'importance de cet aspect aux yeux de l'auteur. Concernant l'estimation des différentes mesures, elle passe par un découpage aléatoire des données en un ensemble apprentissage et un ensemble de validation pour y entraîner le couple d'algorithme étudié. Le processus est répétée 400 fois, afin que la stabilité des modèles puisse être estimée. La variance de l'erreur de classification est utilisée pour estimer la stabilité des prédictions, et pour la stabilité de la sélection des attributs, la mesure est basée sur la fréquence d'apparition de chaque attribut dans les 400 répétions. Au final, les couples d'algorithmes qui ressortent du système ne sont pas les plus performants car ils montrent une stabilité faible qui les pénalise, ni les plus stables car ils montrent une performance faible qui les pénalise, mais ceux sont ceux qui montrent un compromis adéquate entre les deux. L'auteur soutient que ces critères génèrent des modèles qui correspondent mieux à ce qu'attend l'utilisateur.

[66] va un peu plus loin dans la formalisation du concept de stabilité des algorithmes de sélection d'attribut. En s'appuyant sur nos travaux [59], il étudie la probabilité que des algorithmes sélectionnent les mêmes attributs et en déduit un index de stabilité. Cette index mesure la stabilité d'un ensemble de séquences d'attributs générées par un algorithme de sélection d'attribut glouton (*forward feature selection algorithm*). Il est construit pour rester constant si il apparaît que les séquences proviennent de distributions indépendantes, ce qui lui donne un avantage par rapport aux mesures de stabilité proposées jusqu'alors (voir

section 9.1.1). Dans ses expériences, l'auteur étudie le comportement de l'index et il montre comment il peut aider à sélectionner un sous-ensemble d'attributs final à partir de plusieurs résultats de sélection qui sont obtenus en générant des ensembles d'apprentissage par tirage aléatoire. Les expériences exploratoires qui sont menées à ce sujet laissent à penser que lorsque la stabilité est élevée (selon l'index), un sous-ensemble d'attributs obtenu en identifiant les variables qui apparaissent les plus fréquemment dans tous les résultats obtient des performances de classification supérieure à un sous-ensemble d'attributs choisi parmi les résultats. Ce résultat suggère qu'il est possible d'améliorer les performances d'un algorithme de sélection d'attribut en fusionnant les résultats obtenus sur des ensembles d'apprentissage différents, et la stabilité y joue un rôle important. [110] suit un cheminement comparable à [66], et propose lui aussi une mesure de consistance dans l'optique de corriger un effet indésirable des mesures que nous proposons dans [59]. L'intérêt de ces nouvelles mesures c'est qu'elle permettent de comparer plus facilement des méthodes de sélection d'attributs qui génèrent des sous-ensembles d'attributs de taille différentes (voir section 9.1.1 et 9.1.2).

La question soulevée par [66] au sujet de la fusion des résultats de plusieurs sélections d'attributs est abordée spécifiquement dans les excellents articles de [104] et [57]. Dans le premier, l'auteur montre par des expériences rigoureuses qu'une approche simple de fusion permet d'améliorer la stabilité de la sélection d'attributs ainsi que les performances de classification qui en découlent. Les conclusions sont fiables car elles reposent sur un grand nombre d'expériences qui mettent en scène six jeux de données (trois relatifs à la spectrométrie de masse, et trois relatifs aux puces à ADN), quatre algorithmes de sélection d'attributs (SVMRFE, Random Forrest based Feature Selection, RELIEF, Symmetrical Uncertainty) et trois algorithmes de classification (SVM, Random Forrest, 5-Nearest Neighbor). Les algorithmes de sélection d'attributs choisis renvois tous un ordonnancement des attributs. Chacun d'eux est considéré dans sa version simple et dans une version "ensembliste" où il est appliqué à 40 reprises sur des données "bootstrap" de l'ensemble d'entraînement. Les 40 ordonnancements obtenus sont agrégés en un ordonnancement final donnée par la moyenne des rangs de chaque attribut (l'auteur signal qu'une méthode d'agrégation mettant plus de poids sur les attributs avec les meilleurs rangs a donnée des résultats moins satisfaisant que cette approche). Les expériences réalisées montrent que la version ensembliste de l'algorithme de sélection d'attributs est dans la large majorité des cas plus stable que la version simple. Concernant les performances de classification, la version ensembliste mène à des erreurs de prédictions proches voir légèrement supérieures à la version simple (une exception, Random Forrest based Feature Selection). Enfin, la combinaison SVM + "SVMRFE version ensembliste" se distingue des autres à la fois pour sa stabilité et sa performance supérieure. Ce dernier résultat met en relief la pertinence de nos choix pour l'étude que nous menons dans cette deuxième partie qui est justement consacré à l'étude de ce type d'algorithme.

L'article de [57], apporte en revanche une dimension théorique au problème de la fusion de plusieurs résultats d'algorithmes de sélection d'attributs et se focalise plus spécifiquement sur la fusion de plusieurs ordonnancements d'at-

tributs (*Ensemble Feature Ranking*). Dans ce travail, les ordres d'apparition de chaque pair d'attributs (i, j) dans un ordonnancement O_t sont vu comme des variables aléatoires indépendantes avec l'hypothèse que la probabilité d'apparition d'un attribut i avant un attribut j est légèrement différentes de 0.5 (c'est à dire pas entièrement aléatoire). Etant donné T ordonnancements $O_1, ..., O_T$, l'article étudie une stratégie de fusion qui génère un ordonnancement $O*$ en plaçant l'attribut i avant l'attribut j si i apparaît avant j dans la majorité des ordonnancements. L'article montre que cette stratégie de fusion par vote converge assez rapidement vers une solution. De ce résultat théorique, un algorithme de sélection d'attributs est proposé sur la base d'un ensemble d'ordonnancements obtenus avec une méthode à base de courbes ROC. Les expériences menées avec cette approche dans [57] sont également très intéressantes car elles ne reposent pas sur l'évaluation de la performance d'un algorithme de classification (comme c'est généralement le cas), mais sur l'évaluation directe des ensembles d'attributs sélectionnées. Pour pouvoir réaliser cela, des jeux de données synthétiques sont utilisés pour lesquelles on a une connaissance a priori des attributs qui doivent être sélectionnés. En comparant les attributs sélectionnés par la méthode aux attributs attendus, on estime la qualité de l'approche. Le générateur de donnée synthétique prend en compte plusieurs aspects important dans sélection d'attributs dont la redondance entre les attributs, le bruit sur les valeurs, et la linéarité du problème.

Toujours au sujet de la fusion des résultats de plusieurs sélections d'attributs à base de SVM et sur la stabilité, mentionnons également le travail de [9]. Celui-ci propose un algorithme de sélection d'attributs VS-SSVM (Variable Selection via Sparse Support Vector Machines) basé sur l'interprétation des modèles SVM-linéaire. De nombreux aspects rendent VS-SSVM intéressant d'un point de vu conceptuel et pour les applications biologiques. Tout d'abord, il est basé sur l'interprétation de modèle *Sparse*-SVM qui ont généralement davantage de coefficients linéaires à zéro que les SVM-linéaire standards (voir remarque de la section 8.3 sur la norme L1). Ensuite, il s'attaque au problème de stabilité des coefficients dans les modèles SVM-linéaires, et propose un moyen pour réduire la variabilité des attributs sélectionnés à l'aide d'une approche de type "bagging". Cette approche consiste à réaliser l'apprentissage d'une vingtaine de modèles SSVM-linéaire sur des données obtenues par *bootstrap* de l'ensemble de départ, puis à fusionner les résultats en gardant les variables qui ont obtenues au moins une fois un coefficient non nul dans les modèles. Pour éviter d'avoir à choisir le nombre d'attributs à sélectionner, l'algorithme suit la proposition de [112] en introduisant des variables aléatoires dans les données de départ, et les coefficients linéaires qu'elles obtiennent dans les modèles SVM servent à définir un seuil de coupure pour toutes les attributs. Une visualisation de la variabilité des coefficients dans les 20 modèles SVM est offerte et permet à l'utilisateur de mieux appréhender la variabilité de la méthode d'apprentissage. Egalement dans l'optique de faciliter l'utilisation des modèles, les paramètres qui contrôle l'apprentissage (comme par exemple le paramètre C du SVM) sont ajustés automatiquement par la méthode. Dans le même esprit citons également le travail de [56].

Enfin, pour en finir avec la littérature sur la stabilité, mentionnons le récent travail de [72] qui avance encore une étape plus loin dans l'analyse de la stabilité. Ce travail repose sur deux observations importantes pour les données de grande dimension : tout d'abord, l'existence de groupes d'attributs dans les données qui ont un comportement corrélé qui est robuste aux variations des données. Par exemple dans les données d'expression de puces à ADN, certains gènes sont co-régulés est sont activés ou inactivés en même temps. La seconde observation, c'est que si les groupes de variables sont traités comme une seule entité et que l'on réalise un apprentissage au niveau des groupes, les variations aléatoires de certains attributs d'un groupe peuvent être compensées par la valeur des autres attributs du groupe pour en améliorer la robustesse[3]. [72] propose un algorithme de sélection basé sur l'identification de groupe d'attributs corrélées. Effectivement, la formation de groupes d'attributs est un moyen de réduire la dimension des données, d'ailleurs [72] montre sur des données synthétiques que l'identification des groupes d'attributs permet de réduire le nombre d'instance nécessaire pour appliquer SVMRFE avec succès. Pour former les groupes d'attributs d'une manière la plus stable possible, [72] réalise plusieurs clustering des attributs et analyse les résultats afin d'identifier ceux qui apparaissent couramment ensembles. Les groupes ainsi identifiés sont représentés par un seul attribut, celui situé le plus au centre du groupe, et la pertinence de chaque groupe vis à vis de la classe est estimée par une mesure statistique simple (F-mesure) afin d'établir un ordonnancement des groupes. Pour valider son algorithme, [72] introduit une mesure de stabilité avancée qui généralise les mesures de stabilité introduite par les autres auteurs. En effet, cette nouvelle mesure permet d'estimer la similitude entre deux ensembles de groupes d'attributs et non plus seulement entre deux ensemble d'attributs. Les expériences faites sur des données synthétiques et sur six jeux de données de puces à ADN montrent que l'algorithme proposé est plus stable que SVMRFE, pour des performances de classification similaires.

9.3 Bilan de la stabilité

La stabilité de la sélection d'attributs initié dans [59] est devenue un outil important pour caractériser les algorithmes de sélection d'attributs. Ce travail ouvre de nouvelles perspectives de recherche prometteuses pour l'amélioration des algorithmes de sélection d'attributs. Certaines d'entre elles ont été mentionnées dans la section 9.2, et nous pensons que d'autres viendrons s'y ajouter.

Ce que l'on retient également de la section 9.1.3, c'est la stabilité relativement moyenne des algorithmes de sélection d'attributs à base de SVM. Ceux-ci fournissent pourtant des performances de classification attractives, et il reste donc une marge d'amélioration possible sur leur stabilité. Nous avons émis plusieurs hypothèses pour expliquer ces instabilités : la présence de redondance entre les attributs, une mauvaise interprétation des modèles, une mauvaise normalisation des données. Dans le chapitre qui suit, nous proposons une extension

3. Remarquez comme cette notion de groupe est étroitement liée à la notion de redondance entre les attributs que nous avons introduit dans la section 8.1

de SVMRFE qui est motivée par ces observations et par l'étude des interaction entre les attributs. Nous verrons que notre approche se montre effectivement adaptée au traitement de l'information redondante, qu'elle est robuste face à la normalisation des données, et que la stabilité des méthodes de sélection d'attributs à base de SVM qui en découle s'en trouve légèrement améliorée (section 10.4.3).

Chapitre 10

Sélection d'attributs avec noyau logRatio

Ce chapitre présente notre méthode pour la sélection des attributs. Celle-ci s'articule autour du noyau logRatio que nous introduisons dans la section 10.1. Une fois le noyau définie, nous montrons comment il s'intègre à l'algorithme SVM et nous permet de réaliser une sélection d'attributs similaire à SVMRFE mais qui à l'avantage d'être insensible à la normalisation des données (section 10.2). Ensuite, les expériences menées avec le noyau logRatio se divisent en deux parties. La première porte sur des exemples crées artificiellement à partir du jeu de données Iris en y ajoutant de la redondance artificiellement. Ces exemples sont utilisés pour étudier en détail le comportement de l'algorithme de sélection d'attributs SVMRFE-logRatio (section 10.3). La seconde porte sur des jeux de données réelles de grande dimension concernant le domaine de la biologie. Ceux-ci nous permettront d'évaluer d'une part la faculté du noyau logRatio à capturer l'information pertinente contenue dans des jeux de données réelles, et d'autre part la qualité de notre interprétation des classificateurs SVM-logRatio (section 10.4).

10.1 Le Noyau logRatio

Nous définissons l'espace des caractéristiques associé au noyau logRatio au moyen de la fonction de projection ϕ. Celle-ci projette une instance $\mathbf{x}$ de n variables sur le point $\phi(x)$ de n^2 caractéristiques pour lequel chacune des composante est le résultat du logarithme du ratio entre deux attributs de $\mathbf{x}$ dans l'espace initial :

$$\phi(\mathbf{x}) = \left(\log \frac{x_i}{x_j} \right)_{(i,j)=(1,1)}^{(n,n)} \tag{10.1}$$

Les raisons pour lesquelles nous nous intéressons à cette représentation des données est que les biologistes en font un usage intensif dans leurs expériences qui

portent sur l'analyse comparative de données d'expressions, et notamment dans le domaine des puces à ADN [98] même si son utilisation à été récemment remise en question [119]. Les propriétés appréciables de cette mesure est que le signe du résultat dépend de la comparaison entre le numérateur et le dénominateur (positif si le numérateur est supérieur au dénominateur, et négatif sinon), et que la valeur évolue en fonction de l'ordre de grandeur de cet écart. Toujours d'un point de vue biologique, cette représentation des données est également attirante car elle combine deux à deux les attributs (comme le fait le noyau polynomial de degré 2, mais à la place du produit, le logarithme de leur ratio est considéré), ce qui met l'accent sur les interactions attribut-attribut qui intéressent tant les biologistes. Effectivement, très souvent, la présence d'un gène/protéine à un effet d'activation ou d'inhibition sur l'interaction entre deux autres gènes/protéines qui peut être responsable du changement d'état que l'on cherche à diagnostiquer chez un spécimen. Il est également clair que la représentation logRatio est insensible à la normalisation des instances car $\forall \alpha > 0 : \phi(\alpha \mathbf{x}) = \phi(\mathbf{x})$, et ces propriétés rappellent un peu la représentation employé par TSP (voir section 8.5.1). Enfin nous insistons sur les facilités d'interprétation et de visualisation offertes par cette représentation des données. En effet, comme $\log \frac{x_i}{x_j} = \log x_i - \log x_j$, il est possible de visualiser les n^2 composantes d'un vecteur projeté $\phi(\mathbf{x})$ en traçant seulement les n composantes du vecteur $\log \mathbf{x}$ et en considérant l'ensemble des différences entre les composantes (voir figure 10.1). Par contre, comme nous en discuterons plus en détail dans la section 10.1.2, il n'est possible de traiter que des valeurs strictement positives avec ce noyau.

Mentionnons aussi que le logarithme du ratio est une mesure très appréciée des milieux financier et de l'analyste quantitatif. Effectivement, dans ce domaine le logarithme du ratio appliqué aux prix d'un instrument entre deux date est une estimation fidèle du retour sur investissement réalisé sur cet instrument. De plus, les prix étant toujours des valeurs strictement positives, cette représentation ne souffre d'aucune limitation. La raison de l'attractivité de cette représentation tient au fait qu'en première approximation on a $log(1+x) \simeq x$, pour x proche de 0 et que le logarithme introduit une certaine symétrie sur les gain et les pertes. Prenons l'exemple d'un instrument qui a un prix initial de 100chf qui monte à 110chf au terme de la première année et qui chute à 99chf au terme de la seconde. Il réalise ainsi un gain de 10% (=(110-100)/100) la première année, et une perte de 10%((110-99)/110) la seconde année pour une perte totale de 1% ((100-99)/100) sur les 2 ans. Cette perte totale de 1% n'apparaît pas clairement si l'on additionne le gain de +10% et la perte de -10% des deux années. En revanche, si maintenant on fait le calcul des retours r avec le logarithme du ratio des prix entre deux des dates ($r = \log(p_1/p_0)$), on obtient $\log(110/100) = 0.0953 \simeq 10\%$ la première année et $\log(99/110) = -0.1053 \simeq -10\%$ la deuxième année, mais surtout on a $\log(99/100) = \log(110/100) + \log(99/110) \simeq -1\%$, c'est à dire qu'en faisant la somme des retours annuelles on obtient une estimation du retour sur les deux ans.

Il est relativement facile de montrer que la fonction noyau associée à la projection logRatio ci-dessus est donnée par la formule 10.3, c'est à dire qu'elle

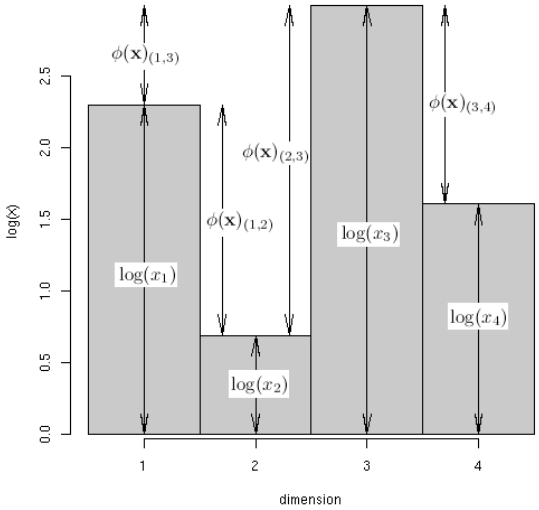


FIGURE 10.1 – Exemple de visualisation des 16 composantes du vecteur $\phi(\mathbf{x})$ dans l'espace de caractéristiques associé au noyau logRatio, à l'aide du logarithme des 4 composantes des attributs originaux ($\log(x)$).

se résume à un calcul de la covariance entre le logarithme des instances (voir annexe C pour les détails du calcul) :

$$\mathcal{K}(\mathbf{x},\mathbf{y}) = \sum_{i,j} \log\frac{x_i}{x_j}\log\frac{y_i}{y_j} \tag{10.2}$$

$$= 2n(n-1)\,\mathrm{cov}(\log(\mathbf{x}),\log(\mathbf{y})) \tag{10.3}$$

Rappelons que la covariance est simplement le produit scalaire entre deux vecteurs desquels on a soustrait de chaque élément la valeur moyenne du vecteur. Cela signifie qu'un apprentissage avec le noyau logRatio revient à réaliser le processus suivant : 1) transformer les valeurs du jeu de données en leur logarithme ; 2) soustraire de chaque instance sa moyenne arithmétique ; 3) réaliser un apprentissage avec le noyau linéaire. C'est donc en apparence assez similaire à un apprentissage linéaire où il n'y a pas de combinaison deux à deux des attributs, mais nous allons voir qu'en réalité la représentation logRatio change notre interprétation des résultats et aboutit à une intéressante stratégie de sélection des attributs quand le noyau est intégré à l'algorithme SVM.

10.1.1 Propriétés du noyau logRatio

En ce qui concerne les propriétés arithmétiques du noyau, les quatre suivantes sont assez faciles à démontrer sans être étonnantes, du fait de l'analogie entre le noyau logRatio et le noyau linéaire. Dans celles-ci, $\mathbf{x}^\alpha$ dénote l'élévation

de chaque composante du vecteur $\mathbf{x}$ à la puissance α, $\mathbf{y} \otimes \mathbf{z}$ correspond à la multiplication composante par composante, et $\frac{\mathbf{y}}{\mathbf{z}}$ à la division composante par composante.

$$\mathcal{K}(\alpha\mathbf{x}, \mathbf{y}) = \mathcal{K}(\mathbf{x}, \mathbf{y}) \tag{10.4}$$

$$\mathcal{K}(\mathbf{x}, \mathbf{y}) + \mathcal{K}(\mathbf{x}, \mathbf{z}) = \mathcal{K}(\mathbf{x}, \mathbf{y} \otimes \mathbf{z}) \tag{10.5}$$

$$\mathcal{K}(\mathbf{x}, \mathbf{y}) - \mathcal{K}(\mathbf{x}, \mathbf{z}) = \mathcal{K}(\mathbf{x}, \frac{\mathbf{y}}{\mathbf{z}}) = -\mathcal{K}(\mathbf{x}, \frac{\mathbf{z}}{\mathbf{y}}) \tag{10.6}$$

$$\alpha\mathcal{K}(\mathbf{x}, \mathbf{y}) = \mathcal{K}(\mathbf{x}, \mathbf{y}^\alpha) = \mathcal{K}(\mathbf{x}^\alpha, \mathbf{y}) \tag{10.7}$$

Signalons aussi que toute normalisation des attributs de l'espace initial se traduit en une translation de l'espace des caractéristiques. En effet, si chaque attribut i de l'espace initial doit subir un facteur de normalisation β_i, alors $\phi(\mathbf{x}) = \left(\log \frac{\beta_i x_i}{\beta_j x_j}\right) = \left(\log \frac{x_i}{x_j} + \log \frac{\beta_i}{\beta_j}\right)$, c'est à dire que chaque composante ij de l'espace de caractéristiques subit une translation $\log \frac{\beta_i}{\beta_j}$. De cette propriété du noyau, l'on déduit que la distance euclidienne dans l'espace de caractéristiques de logRatio entre deux instances $\phi(\mathbf{x})$ et $\phi(\mathbf{y})$ ne change pas quelle que soit la normalisation des attributs dans l'espace initial. Ceci constitue une propriété intéressante pour celui qui souhaite par exemple implémenter un algorithme de classification par voisins proches qui n'est sensible ni à la normalisation des instances, ni à la normalisation des attributs. Nous allons aussi voir dans la section qui suit (section 10.2) que, de par cette propriété, l'algorithme SVM à base de noyau logRatio est aussi insensible à la normalisation des instances et des attributs.

A ce niveau, il est intéressant de faire un parallèle avec la méthode de sélection d'attributs proposé par [99] et que nous avons évoqué dans la section 8.4.1. Effectivement, pour le noyau logRatio, l'introduction des facteurs multiplicatifs ρ se traduit par une simple translation de l'espace de caractéristiques ce qui ne change pas les bornes $\mathcal{L}$ de l'erreur de généralisation que cherche à minimiser [99]. En d'autre termes, pour le noyau logRatio le gradient de la borne supérieure $\mathcal{L}$ est nulle ($\frac{\partial \mathcal{L}}{\partial \rho_k} = 0$) ce qui empêche d'appliquer l'approche d'élimination d'ordre 1 pour ce noyau, mais l'approche de minimisation d'ordre 0 reste valable. Cela montre également une certaine robustesse de notre noyau face aux méthodes SVM.

10.1.2 Problème des valeurs négatives et nulles :

Le logarithme d'un nombre n'étant pas défini pour les valeurs négatives, le noyau logRatio n'est pas en mesure de projeter une instance lorsque le ratio entre deux de ses dimensions est négatif. C'est pourquoi nous devons nous limiter aux jeux de données dont les instances possèdent toutes des valeurs de même signe. Heureusement, de nombreux jeux de données contiennent des valeurs exclusivement positives ce qui atténue les limitations imposées par cette contrainte. Par exemple, de manière assez général, les périphériques de capture de signaux produisent des données qui contiennent seulement des valeurs positives (spectre de masse, images, videos, ...). Par contre, logRatio n'est pas non

plus capable de traiter les valeurs nulles qui sont contenues dans ces données et cela constitue une limitation importante de la méthode.

Plusieurs solutions peuvent être envisagés pour contourner le problème des valeurs nulles. La première consisterai à translater les données de manière à ce que la plus petite valeur qu'elle contiennent soit positive, ce qui peut être obtenue en ajoutant une valeur $\epsilon > 0$ à toutes les valeurs du jeu de donnée – noter qu'il faut de préférence choisir un ϵ petit pour que les rapports entre les valeurs ne soient pas trop altéré. Cette approche à cependant l'inconvénient de modifier l'ensemble des valeurs du jeu de donnée, c'est pourquoi dans la suite nous avons préféré une alternative moins destructive qui remplace seulement les valeurs inférieures à ϵ par ϵ. Dans ce cas la il faut cependant faire attention de choisir un ϵ petit par rapport aux autres valeurs du jeu de donnée pour éviter un seuillage trop brutal.

10.2 SVM-logRatio & SVMRFE-logRatio

De la même manière que les propriétés du noyau linéaire permettent de simplifier l'équation de la fonction de décision d'un SVM, les propriétés du noyau logRatio permettent d'aboutir à une simplification très proche. En effet, en utilisant les propriétés (10.5) et (10.7) dans l'équation 8.3, la fonction de décision de SVM-logRatio devient :

$$f(\mathbf{x}) = \text{sgn}(\mathcal{K}(\prod_i \mathbf{x}_i{}^{y_i\alpha_i}, \mathbf{x}) + b) \tag{10.8}$$

et, en posant $\mathbf{w} = \prod_i \mathbf{x}_i{}^{y_i\alpha_i}$ nous obtenons

$$f(\mathbf{x}) = \text{sgn}(\mathcal{K}(\mathbf{w}, \mathbf{x}) + b) \tag{10.9}$$

ce qui est identique à l'expression obtenue dans le cas du SVM linéaire (équation 8.4), excepté que le produit scalaire est remplacé par la fonction noyau logRatio. L'expression résultante est plus simple car le vecteur $\mathbf{w}$ concentre à lui seul toute l'information du modèle, et il suffit d'un seul calcul de la fonction noyau entre l'instance de test $\mathbf{x}$, et le vecteur $\mathbf{w}$ construite durant l'apprentissage pour faire la prédiction. Noter aussi la similitude entre le vecteur $\mathbf{w}$ de SVM-logRatio qui est la moyenne géométrique pondéré des vecteurs supports, et le vecteur $\mathbf{w}$ de SVM-linéaire qui est la moyenne arithmétique pondéré des vecteurs supports. Encore une fois, ces observations n'ont rien d'étonnant du fait de l'analogie entre le noyau logRatio et le noyau linéaire.

Cependant, la représentation logRatio intervient différemment du noyau linéaire au niveau de l'interprétation des modèles SVM. Dans le cas linéaire le vecteur $\phi(\mathbf{w}) = \mathbf{w}$ s'interprète comme un vecteur normal à l'hyperplan séparateur qui contient n coefficients linéaires associés à chacun des n attributs de l'espace initial, alors que dans le cas logRatio, le vecteur $\phi(\mathbf{w})$ s'interprète aussi comme un vecteur normal à l'hyperplan séparateur mais il contient n^2 coefficients linéaires associés à chaque caractéristique de l'espace logRatio. Les

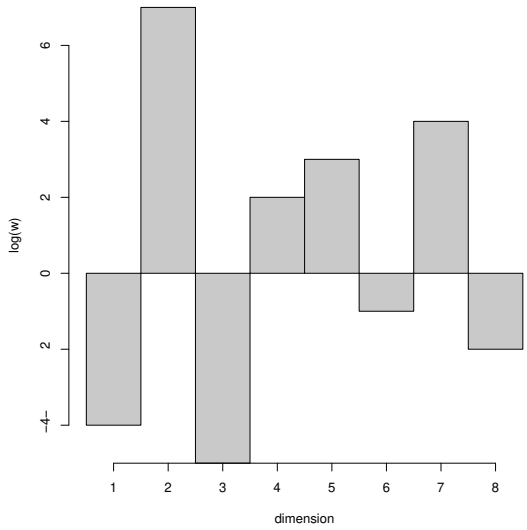


FIGURE 10.2 – Exemple de visualisation d'un modèle SVM-logRatio.

n^2 coefficients de $\phi(\mathbf{w})$ dans le cas logRatio peuvent être matérialisés sur une figure identique à celle de la figure 10.2 en traçant les valeurs du vecteur $\log(\mathbf{w})$. Sur une telle visualisation, il est facile d'identifier le coefficient de $\phi(\mathbf{w})$ le plus important en valeur absolue car c'est celui qui combine les deux valeurs extrêmes de la visualisation (le maximum avec le minimum). Nous pourrions même envisager d'ordonner les n^2 caractéristiques de l'espace logRatio en fonction de leur importance dans la prise de décision du modèle SVM grâce aux n^2 différences du vecteur $\log(\mathbf{w})$, mais ceci n'est pas vraiment intéressant car trop coûteux en temps de calcul et car notre but est plutôt d'ordonner les attributs de l'espace initial, or chaque caractéristique est produite en combinant deux attributs de l'espace initial.

Si on veut identifier l'attribut de l'espace initial qui participe le moins à la prise de décision de SVM-logRatio, en vue de son élimination comme dans SVMRFE, il faut chercher l'attribut qui produit les plus petits coefficients linéaires lorsqu'il est combiné à tous les autres par la transformation logRatio. Pour l'identifier, il s'agit de trouver l'attribut k qui minimise $\sum_{i=1}^{n}(|\log(w_i) - \log(w_k)|)$, dont la solution est donnée par l'attribut k qui a la valeur $\log(w_k)$ médiane. La stratégie d'ordonnancement qui découle de ce résultat consiste donc à trier les attributs en fonction de leur distance (rang) par rapport à l'attribut médian (de rang $n/2$), les plus pertinents étant les attributs les plus éloignés que sont le minimum (rang 1) et le maximum (rang n). En insérant cette stratégie d'ordonnancement des attributs dans une boucle RFE, nous définissons l'algorithme de sélection d'attribut SVMRFE-logRatio dont le pseudo code est donnée par l'algorithme 10. Notez que nous aurions pu choisir une autre fonc-

Algorithm 10 SVMRFE-logRatio

1: Entrée : X un jeu de données
2: Sortie : r[] l'ordonnancement des attributs du pire vers le meilleur
3: **while** X contient des attributs **do**
4: $\mathbf{w} \leftarrow$ entraîner un SVM-logRatio sur X
5: Trier le vecteur $\log \mathbf{w}$
6: $F \leftarrow$ { l'attribut qui à la valeur médiane dans $\log(\mathbf{w})$ plus éventuellement les attributs qui ont les rangs les plus proches de lui }
7: Eliminer de X les attributs de l'ensemble F
8: $r \leftarrow r \cup F$
9: **end while**

tion de score pour ordonner les attributs selon leur pertinence dans les modèles logRatio. Par exemple nous aurions pu choisir d'éliminer l'attribut k qui minimise $\sum_{i=1}^{n}(\log(w_i)-\log(w_k))^2$, ce qui revient à éliminer l'attribut dont la valeur est la plus proche de la moyenne (plutôt que de la médiane).

La seconde chose à observer dans les modèles logRatio-SVM comme celui représenté figure 10.2 sont les relations et les contraintes entre les coefficients qui ont un lien avec la redondance des attributs. Prenons l'exemple de trois attributs a_1, a_2, a_3 dont les valeurs sur la visualisation sont $\log w_1 = -4$, $\log w_2 = 7$, $\log w_3 = -5$. Le coefficient linéaire le plus faible est associée à la caractéristique a_1a_3 qui combine les attributs a_1 et a_3 et pour lequel l'écart n'est que de 1. Cela signifie que la valeur de la caractéristique a_1a_3 influence peu la décision du modèle SVM-logRatio, en revanche les valeurs des combinaisons a_2a_3 (écart de 12) et a_2a_1 (écart de 11) influencent beaucoup plus le modèle. En résumé, a_1 et a_3 sont des attributs importants pour le modèle lorsqu'ils sont combinés avec a_2, mais pas lorsqu'ils sont combinés ensemble. Ce motif apparaît par exemple lorsque les trois attributs a_1,a_2,a_3 sont pertinents pour la classification des instances et que $a1$, a_3 contiennent des informations redondantes : en effet, dans ce cas la combinaison a_1a_3 est moins utile que les autres pour la classification et elle se voit en conséquence assigné un coefficient faible ; d'un autre coté, lorsque a_1 et a_3 sont combinés avec a_2, ils gagnent en pertinence et c'est pourquoi ils sont gratifiés d'un coefficient plus élevé. Plus généralement, nous pouvons dire que si deux attributs a_i et a_j apportent une information redondante à l'information de classe, alors ils sont susceptibles d'avoir des valeurs $\log w_i$ et $\log w_j$ proches, mais l'inverse n'est pas forcément vrai, c'est à dire que si les valeurs de deux attributs sont proches, alors ils ne sont pas forcément redondants. On voit bien dans cet exemple les contraintes qui s'appliquent durant l'apprentissage du modèle SVM-logRatio, et également l'utilité de combiner deux à deux les attributs pour en déduire des informations sur leur redondance (même si il faut prendre certaines précautions dans la lecture de ces résultats). En revanche, une interprétation similaire des coefficients appris par un modèles SVM-linéaire semble plus difficile à interpréter car les coefficients w_i associés à chaque attributs sont libres d'évoluer indépendamment les uns des autres sans qu'il n'y ait explicitement de contraintes entre eux comme dans SVM-logRatio. Certes, si

deux attributs apportent des informations identiques, SVM-linéaire a également tendance à assigner des coefficients similaires, mais notre interprétation classique de ce modèle nous suggère qu'il s'agit d'une conséquence qui ne relève pas de contraintes explicite entre les attributs [1].

Pourtant, cette observation pourrait appuyer une autre stratégie pour ordonner les attributs à partir du modèle : à la place d'utiliser la valeur médiane pour déterminer l'attribut le moins influent du modèle SVM-logRatio, nous pourrions envisager d'éliminer l'un des deux attributs qui produit le plus petit coefficient linéaire dans l'espace des caractéristiques. Celui-ci correspond aux deux attributs a_i, a_j qui ont les valeurs $\log w_i$ et $\log w_j$ les plus proches, or ceux-ci ont des chances d'être redondants. En d'autre termes, cette stratégie d'élimination viserait l'objectif d'éliminer la redondance. Ce critère d'élimination des attributs a été testé, mais il a affiché des résultats très peu satisfaisants qui s'explique par le fait qu'il ne se soucie pas de la pertinence de l'attribut éliminé. Nous ne l'avons donc pas utilisé dans les expériences qui suivent.

Enfin, plus généralement, il est intéressant d'analyser les contraintes sur le vecteur $\log \mathbf{w}$ pendant l'apprentissage des modèles SVM-logRatio. Noter par exemple que la modification d'une des valeurs dans $\log \mathbf{w}$ à pour effet de la rapprocher ou de l'éloigner des $n-1$ autres. Ainsi, durant l'apprentissage, l'augmentation d'une des valeurs $\log w_i$ a pour effet de l'éloigner des valeurs $\log w_j$ situées en dessous d'elle ($\log w_j \leq \log w_i$) ce qui accroît l'importance des coefficients linéaires associés aux caractéristiques $a_i a_j$, et inversement, cela la rapproche aussi des $\log w_k$ situées au dessus ($\log w_k > \log w_i$) ce qui diminue l'importance des caractéristiques $a_i a_k$ pour le modèle. Finalement, la valeur $\log w_i$ est ajustée de manière à ce que l'ensemble des coefficients $(\log w_i - \log w_j)$ que l'on peut former avec les valeurs $\log w_j$ supérieures forment un équilibre avec l'ensemble des coefficients $(\log w_k - \log w_i)$ que l'on peut former avec les valeurs $\log w_k$ inférieures. Au terme de l'apprentissage, l'ordre des valeurs du vecteur $\log \mathbf{w}$ reflète cet équilibre, et chaque valeur $\log w_i$ pour un attribut a_i prend en considération l'ensemble des coefficients linéaires que l'on peut former avec l'attribut i dans la représentation logRatio, ce qui peut aussi être perçu comme une une forme de régularisation des modèles.

10.2.1 Insensibilité à la normalisation

Un des intérêts des algorithmes SVM-logRatio et SVMRFE-logRatio par rapport à SVM-linéaire et SVMRFE-linéaire est qu'ils sont insensibles à la fois à tous facteur de normalisation des instances et à tous facteur de normalisation des attributs de l'espace initial, ce qui rend inutile un tel pré-traitement des données. Ils sont clairement insensibles à la normalisation des instances car le noyau logRatio l'est (propriété 10.4), en revanche l'insensibilité vis à vis de la normalisation des attributs est moins triviale et nécessite quelques explications.

1. En réalité, nous verrons dans la section 11.1.4 qu'une interprétation particulière des modèles SVM-linéaire permet de faire apparaître des contraintes entre les coefficients qui permettent de tirer des conclusions similaires à celles tiré pour logRatio

Nous avons vu (section 10.1.1) que la normalisation des attributs de l'espace initial est équivalente à une translation de l'espace des caractéristiques et que les distances entre les instances dans cet espace sont invariables quelle que soit la normalisation des attributs qui est effectuée. En conséquence, l'hyperplan séparateur de marge maximal appris par SVM-logRatio subit seulement une translation, mais il a la même direction quelle que soit la normalisation des attributs que l'on applique aux instances, c'est-à-dire que les coefficients linéaires associés à chaque dimension de l'espace des caractéristiques sont identiques et que seul l'intercepte du plan change. La normalisation des attributs à donc seulement un effet sur l'intercepte, b, des modèles SVM-logRatio et aucun sur $\phi(\mathbf{w})$, ce qui ne change pas notre interprétation des résultats.

Pour vérifier que SVM-logRatio ne nécessite aucune normalisation, nous avons mené des expériences ou d'une part SVM-logRatio est appliqué sur les données d'origine ; et d'autre part, ou nous avons au préalable "dénormalisé" les données en multipliant chaque instance et chaque attribut par des facteurs aléatoires positifs. Les modèles obtenus par SVM-logRatio à partir des deux variantes sont exactement identiques (mis à part l'intercepte de l'hyperplan). En revanche, il est connu [43] que le SVM-linéaire exige une normalisation préalable. Nous avons donc normalisé ces mêmes deux variantes (suivant une approche classique de normalisation des instances par la L1-norme et de standardisation des attributs) avant l'apprentissage. Les modèles obtenus avec SVM-linéaire étaient différents.

Maintenant que nous avons prouvé que SVMRFE-logRatio est insensible à la normalisation des attributs, il est intéressant de faire le rapprochement de cette méthode de sélection d'attributs et l'approche proposée par [124] que nous avons évoqué dans la section 8.4.3. Supposez en effet que l'on utilise un SVM-logRatio à la place d'un SVM-linéaire à l'étape 1 de l'algorithme 9 dans la section 8.4.3. Comme la multiplication des variables d'entrée à l'étape 2 de l'algorithme n'a que pour effet de translater l'espace des caractéristiques de SVM-logRatio, il serait inutile de répéter ces étapes comme si le noyau logRatio prenait déjà en compte la norme L0. Ce parallèle est certes un peu exagéré, mais il est intéressant de remarquer les implications induite par une approche insensible à la normalisation.

10.3 Expériences sur données semi-synthétiques

Dans cette section, nous étudions le comportement des algorithmes de sélection d'attributs SVMRFE-logRatio et SVMRFE-linéaire sur des exemples simples dérivés du jeu de données "Iris" [83]. Nous cherchons ici à étudier en détail le fonctionnement des deux algorithmes de sélection d'attributs, et à comprendre leurs différences. Pour cela, nous nous attardons sur des exemples simples pour lesquels les résultats sont faciles à interpréter. Deux types d'expériences sont menées : d'une part nous étudions les modèles dans différents cas de normalisation des données Iris (section 10.3.1) ; d'autre part, nous introduisons de la redondance dans Iris afin d'observer la réaction des modèles (section 10.3.2).

Avant de commencer à décrire les résultats expérimentaux, signalons que nous avons réalisé toutes nos expériences (de cette section et des sections qui suivent) avec notre propre implémentation de SVMRFE qui diffère légèrement de celle de [43]. La seule différence en réalité est que nous utilisons un modèle ν-SVM [22] plutôt qu'un modèle C-SVM (voir sections 8.2, 8.3) dans notre implémentation de SVMRFE. L'avantage de cette démarche est que le choix du paramètre ν est plus cohérent d'une itération RFE à l'autre, alors qu'en cas d'utilisation du paramètre C, la valeur devrait être remise en cause à chaque élimination d'attributs dans les itérations RFE (voir discussion dans section 8.2.1). Les expériences sont réalisées au moyen du logiciel R [2] et du package *e1071* qui contient une implémentation de ν-SVM basée sur la librairie *libsvm-2.83* [13]. Le paramètre ν de l'algorithme ν-SVM est toujours fixé à la valeur $\nu = 0.3$, c'est à dire que l'on tolère une erreur de marge sur 30% des exemples d'apprentissage.

Le jeu de donnée Iris contient 150 instances de fleurs d'iris réparties en trois espèces : setosa, virginica et versicolor. Afin de simplifier l'analyse des résultats, nous préférons nous concentrer sur un problème à deux classes équilibrées, c'est à dire avec autant d'instance dans chaque classe. En conséquence, nous limitons notre analyse aux 50 fleurs de type virginica et aux 50 fleurs de type versicolor qui sont les plus difficiles à distinguer. Chaque fleur est caractérisée par quatre valeurs qui correspondent à la longueur et la largeur de leurs pétales et de leurs sépales exprimés en centimètres. La figure 10.3 offre une visualisation de ces données et a l'avantage de montrer les attributs par paire ce qui fournit des informations à la fois sur leur pertinence et sur leur redondance. Par exemple, les attributs "Petal.Length(cm)" et "Sepal.Length(cm)" semblent redondants car les valeurs du premier paraissent linéairement corrélées à celles du second. En même temps, ces attributs sont très pertinents pour la classification car ils séparent distinctement les deux espèces de fleurs. Un bon algorithme de sélection d'attributs doit être capable de décider si il faut conserver seulement l'un des deux car l'autre apporte une information trop redondant, ou si il faut conserver les deux pour éviter d'éliminer trop d'information pertinente. Nous allons justement étudier nos algorithmes sur ces données en menant les deux types d'expériences mentionnées plus haut.

10.3.1 Expériences sur données Iris

Cette série d'expériences analyse les différences de comportement entre les modèles SVM-linéaire et SVM-logRatio obtenus sur Iris. Ces modèles sont utilisés par les algorithmes SVMRFE-linéaire et SVMRFE-logRatio pour décider des attributs à éliminer. Chaque modèle est entraîné à la fois sur les données originales et sur les données normalisées, donc quatre modèles sont produits au total (voir figure 10.4). Les données sont normalisées de manière à ce que les attributs aient une déviation standard de 1. La table 10.1 donne les déviations standards de chacun des attributs et montre que les variations des largeurs sont

2. `http://www.r-project.org`

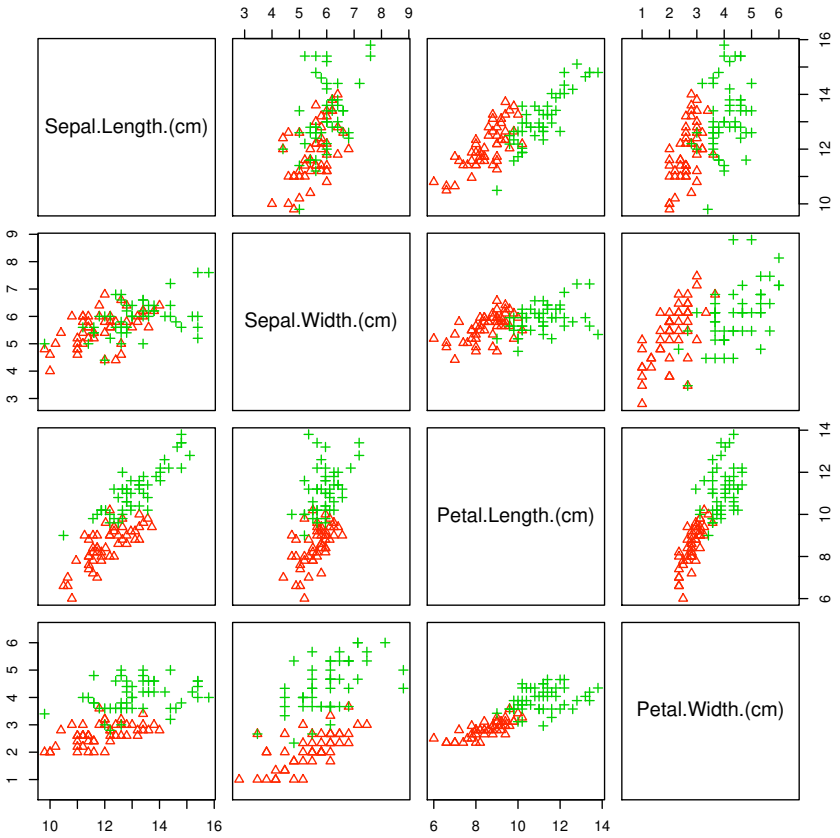


FIGURE 10.3 – Le jeu de données Iris limité aux espèces virginica et versicolor

deux fois inférieures à celles des longueurs, ce qui signifie que la normalisation a pour effet de doubler les valeurs des attributs de largeurs par rapport aux attributs de longueurs. Ce comportement correspond bien à celui espéré pour corriger le fait que une largeur est par définition inférieure à une longueur. Cependant, comme nous l'avons mentionné dans la section 8.5, il ne s'agit pas forcement de la normalisation idéale. En ce qui concerne la normalisation des instances, celles-ci ont des valeurs cohérentes entre elles et n'ont pas besoin d'être normalisées.

Les quatre modèles obtenus sont graphiquement représentés figure 10.4. Comme on s'y attendait, le modèle SVM-linéaire change entre les données normalisées et les données originales, alors que le modèle SVM-logRatio reste identique car la représentation des données qu'il emploi n'est pas sensible à la nor-

Attribute	Standard Deviation	Coefficient of Variance	
	virginica & versicolor	virginica	versicolor
Sepal.Length.(cm)	0.662	8.6%	9.6%
Sepal.Width.(cm)	0.332	11.3%	10.8%
Petal.Length.(cm)	0.825	11.0%	9.9%
Petal.Width.(cm)	0.424	14.9%	13.5%

TABLE 10.1 – Deviation standard des attributs de Iris sur l'ensemble des 100 instances, ainsi que le coefficient de variance pour chaque classe.

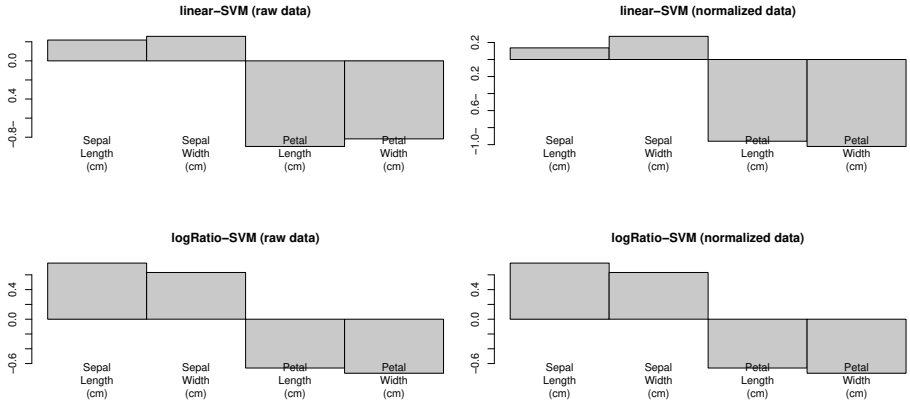


FIGURE 10.4 – Modèles SVM-linéaire et SVM-logRatio obtenus sur iris sans normalisation (à gauche) et iris avec standardisation (à droite).

malisation. Il est donc important de normaliser correctement les données lorsque SVMRFE-linéaire est utilisé, c'est pourquoi dans la suite nous considérons seulement des modèles obtenus avec des données normalisées car les coefficients linéaires du modèle sont censés mieux représenter le pouvoir discriminant des attributs. Essayons maintenant d'interpréter les modèles SVM-linéaire (normalisés) en vu d'ordonner les attributs selon leur influence sur les prédictions, comme le font en interne SVMRFE-linéaire et SVMRFE-logRatio. Malgré que les modèles SVM-logRatio et SVM-linéaire soient visuellement assez proches sur la figure 10.4, leur interprétation est différente. En effet, pour SVM-linéaire, les attributs les plus discriminants sont ceux avec les coefficients les plus forts en valeur absolue, c'est à dire Petal.Length(cm) et Petal.Width(cm) alors que pour SVM-logRatio les attributs les plus discriminants sont ceux pour lesquels la différence entre les coefficients est maximum, c'est à dire d'un coté une information sur le pétale (Petal.Width(cm) ou Petal.Length(cm)) et de l'autre une information sur le sépale (Sepal.Width(cm) ou Sepal.Length(cm)). En conséquence, les modèle SVM-linéaire et SVM-logRatio ne mettent pas l'accent sur les mêmes

iris (normalized)		
rank	linear	logRatio
1	Petal.Width.(cm)	Sepal.Length.(cm)
2	Petal.Length.(cm)	Petal.Length.(cm)
3	Sepal.Width.(cm)	Petal.Width.(cm)
4	Sepal.Length.(cm)	Sepal.Width.(cm)

TABLE 10.2 – Ordonnancement d'attributs produits par SVMRFE-linéaire et SVMRFE-logRatio sur Iris (standardisé)

informations : le premier base essentiellement sa décision sur les dimensions du pétale, alors que le second base la sienne sur des dimensions de nature séparés, une du pétale et une du sépale. D'un point de vu expert, on peut supposer que le couple (Sepal.Length(cm), Petal.Length(cm)) extrait par SVM-logRatio est le couple idéale pour le jeu de donnée "Iris" car il combine des informations d'origines différentes (une du Petal et une du Sepal), et privilégie les longueurs aux largeurs, or celles-ci devrait avoir une erreur de mesure plus faible car leur dimension est plus grande –pour vérifier quantitativement cela, on peut se reporter aux coefficients de variance des attributs de iris (table 10.1) qui sont effectivement inférieurs sur les longueurs.

SVM-logRatio à également un comportement intéressant vis à vis de la redondance. Nous avons effectivement vu que les attributs redondants devaient avoir des valeurs proches dans le modèle SVM-logRatio (section 10.2), d'ailleurs Petal.Length(cm) et Petal.Width(cm) semblent linéairement corrélés sur la figure 10.3 et ils obtiennent des valeurs proches dans le modèle SVM-logRatio. Mais, les attributs Sepal.Length(cm) et Petal.Length(cm) semblent également redondant sur la figure 10.3, et pourtant leurs valeurs sont éloignés dans le modèle. SVM-logRatio a préféré assigner un poids fort à ce couple car il estime que l'information que l'un apporte à l'autre est plus pertinente que redondante. Cette exemple montre bien les conflits entre la pertinence et la redondance des attributs dans les modèles logRatio-SVM, et ce type d'analyse des modèles logRatio peut fournir des informations précieuses à celui qui cherche à comprendre les interactions attribut-attribut impliquées dans le mécanisme de prédictions des classes, comme la recherche de bio-marqueurs à partir de données d'expression. Effectivement, une méthode classique de détéction de redondance par corrélation aurait indiqué à l'utilisateur que Sepal.Length(cm) et Petal.Length(cm) sont redondant, mais en combinant cette information avec l'interprétation des modèles logRatio-SVM on s'apercoit que les deux informations sont également pertinente pour la prédiction de la classe. Dans un cadre biologique, cela peut permettre de mettre en évidence des attributs/protéines particulièrement intéressants à étudier.

Rappelons aussi que l'ordre des attributs déduit des modèles SVM-linéaire et SVM-logRatio ne correspond pas forcement à l'ordre final rendu par SVMRFE-linéaire et SVMRFE-logRatio car les modèles évoluent au fur et à mesure que les attributs sont éliminés. Pour observer cet effet dans nos expériences,

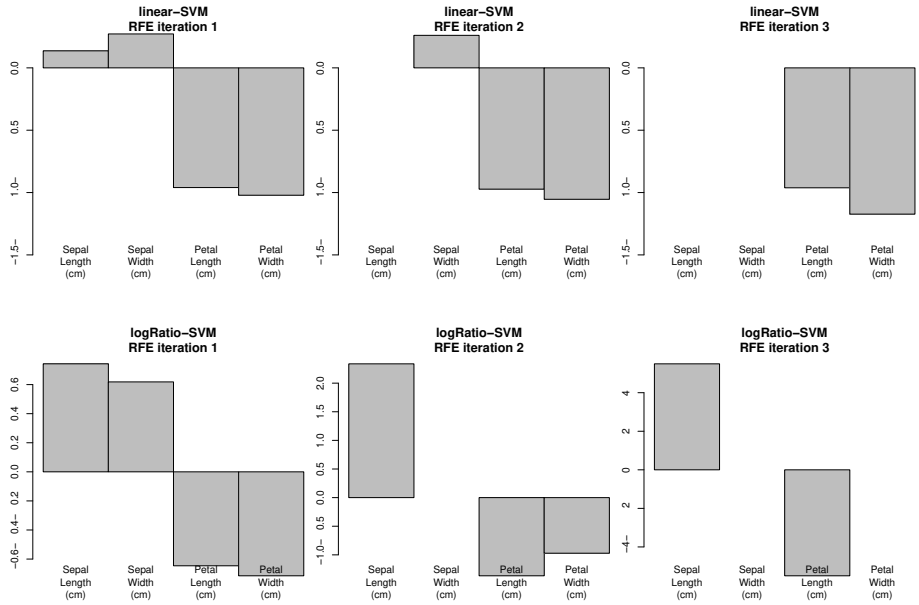


FIGURE 10.5 – Modèles successivement produits par linear-SVMRFE et logRatio-SVMRFE sur le jeu de donnée Iris standardisé

la figure 10.5 montre l'évolution des modèles SVM-linéaire et SVM-logRatio dans SVMRFE-linéaire et SVMRFE-logRatio itération après itération (sur les données normalisées), et la table 10.2 affiche les ordonnancements finaux. A partir de la figure, peu de renseignements utiles peuvent être tirés de l'évolution des modèles SVM-linéaires dont les coefficients $\mathbf{w}$ peuvent évoluer indépendamment les uns des autres. Par contre, les valeurs de $\log \mathbf{w}$ dans SVM-logRatio sont liées entre elles et l'évolution des modèles fournit des informations intéressantes. On voit par exemple, sur la figure 10.5, que la suppression de l'attribut Sepal.Width(cm) dans la première itération de SVMRFE-logRatio a pour effet de diminuer l'importance de l'attribut Petal.Width(cm) dans le modèle de l'itération 2, ce qui suggère que les deux attributs se complètent (ce que que l'on peut vérifier sur la figure 10.3). La table montre quand à elle que les ordonnancement produits après les trois itérations RFE correspondent approximativement à ceux déduits des modèles de l'itération 1. Il ne faut cependant pas en déduire que ces itérations sont inutiles car nous allons maintenant voir qu'elles se révèlent très puissantes pour supprimer les attributs vraiment redondants.

	$\text{iris}_{(\delta_{\text{mm}}=0,\delta_{\text{cm}}=0)}$		$\text{iris}_{(\delta_{\text{mm}}=0,\delta_{\text{cm}}=0.5)}$		$\text{iris}_{(\delta_{\text{mm}}=5,\delta_{\text{cm}}=0.5)}$	
Rang	linear	logRatio	linear	logRatio	linear	logRatio
1	P.W.(mm)	S.L.(mm)	P.W.(mm)	S.L.(mm)	P.L.(cm)	P.L.(cm)
2	P.L.(mm)	P.L.(mm)	P.L.(mm)	P.L.(mm)	P.W.(cm)	S.L.(cm)
3	P.W.(cm)	P.W.(cm)	P.L.(cm)	P.W.(mm)	P.L.(mm)	P.L.(mm)
4	P.L.(cm)	S.W.(cm)	S.W.(mm)	S.W.(mm)	P.W.(mm)	S.W.(cm)
5	S.W.(cm)	S.L.(cm)	P.W.(cm)	P.L.(cm)	S.L.(mm)	P.W.(cm)
6	S.W.(mm)	P.L.(cm)	S.L.(mm)	S.L.(cm)	S.L.(cm)	S.L.(mm)
7	S.L.(mm)	P.W.(mm)	S.L.(cm)	S.W.(cm)	S.W.(cm)	P.W.(mm)
8	S.L.(cm)	S.W.(mm)	S.W.(cm)	P.W.(cm)	S.W.(mm)	S.W.(mm)

TABLE 10.3 – Comparaison des ordonnancement produits par SVMRFE-linéaire et SVMRFE-logRatio sur Iris avec redondance introduite artificiellement. (P.L :Petal Length ;P.W : Petal Width ;S.L :Sepal Length ;S.W Sepal Width)

10.3.2 Expériences avec redondance artificielle

Pour introduire de la redondance dans Iris, quatre attributs supplémentaires sont ajoutés qui reflètent les dimensions des pétales et des sépales en millimètres, c'est à dire qu'ils correspondent aux quatre attributs originaux multipliés par un facteur 10. Du bruit est rajouté en additionnant à chaque valeur exprimée en millimètre un nombre aléatoire tiré selon la loi normale $\mathcal{N}(0, \delta_{\text{mm}})$, et à chaque valeur exprimée en centimètre un nombre tiré aléatoirement selon la loi $\mathcal{N}(0, \delta_{\text{cm}})$. Au final, ce processus simule deux acquisitions indépendantes de la même valeur exprimées dans des unités différentes. Cette répétition de l'information n'est donc pas forcement inutile car en combinant les attributs exprimés en centimètre avec ceux exprimés en millimètre il est possible de réduire le bruit et d'améliorer la précision de chaque mesure. Le comportement espéré pour un algorithme de sélection d'attribut dans le cadre de ces jeux de données dépend des paramètres (δ) qui contrôlent le bruit. Trois cas sont étudiés dans nos expérimentations : $\text{iris}_{(\delta_{\text{mm}}=0,\delta_{\text{cm}}=0)}$, $\text{iris}_{(\delta_{\text{mm}}=0,\delta_{\text{cm}}=0.5)}$, et $\text{iris}_{(\delta_{\text{mm}}=5,\delta_{\text{cm}}=0.5)}$. Les algorithmes linear-SVMRFE et logRatio-SVMRFE sont exécutés sur chacun des trois jeu de données en éliminant un attribut à chaque itération. Dans le cas de SVMRFE-linéaire, les attributs sont standardisées au préalable alors que dans le cas de SVMRFE-logRatio aucun pré-traitement n'est effectué. Comme précédemment, la table 10.3 donne les différents ordonnancements d'attributs obtenus, et les figures 10.6, 10.7, 10.8 montrent les modèles SVM qui sont produit au fur et à mesure que les attributs sont éliminés. Les sous-sections qui suivent commentent ces résultats.

La figure 10.6 dévoile le comportement de SVMRFE-linéaire et SVMRFE-logRatio sur $\text{iris}_{(\delta_{\text{mm}}=0,\delta_{\text{cm}}=0)}$ qui ne contient aucune information bruitée. Comme les attributs en mm et en cm sont exactement identiques mais exprimées dans des unités différentes, le comportement attendu des algorithmes de sélection d'attributs sur ces données est qu'ils éliminent au moins une des information

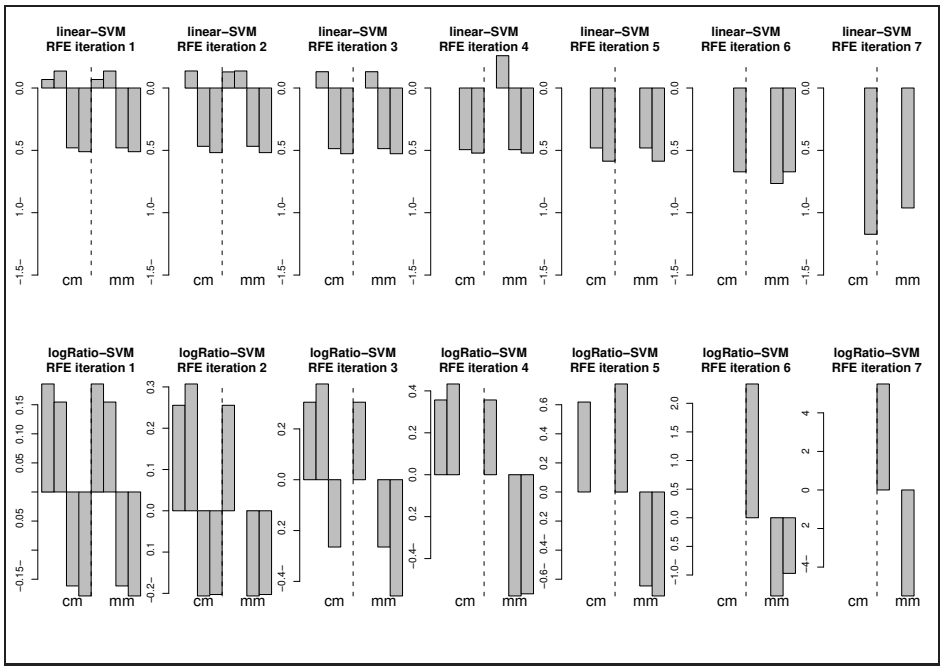


FIGURE 10.6 – Modèles SVM-linéaire et SVM-logRatio produits par SVMRFE-linéaire et SVMRFE-logRatio sur $\text{iris}_{(\delta_{\text{mm}}=0,\delta_{\text{cm}}=0)}$. Les attributs sont ordonnés de gauche à droite selon la séquence : Sepal.Length(cm), Sepal.Width(cm), Petal.Length(cm), Petal.Width(cm), Sepal.Length(mm), Sepal.Width(mm), Petal.Length(mm), Petal.Width(mm). Attention l'axe des ordonnées des modèles logRatio-SVM n'est pas uniforme

redondante indifféremment de son unité. La table 10.3 montre que SVMRFE-logRatio remplie parfaitement cette tâche en exposant quatre dimensions d'origine différente en haut du classement, alors que SVMRFE-linéaire sélectionne deux fois les mêmes informations. L'explication de ce phénomène se trouve dans les modèles de la figure 10.6, où on s'aperçoit qu'immédiatement après la suppression d'un attribut le modèle logRatio assigne un poids très fort à l'attribut redondant correspondant. Par exemple à l'itération 1, les modèles sont symétriques (dans le sens ou les poids assignés aux attributs mm sont identiques à ceux des attributs cm pour les deux modèles) et les attributs avec la valeur médiane sont Sepal.Width(cm) et Sepal.Width(mm). Sepal.Width(mm) est supprimé, ce qui à pour effet d'affecter une valeur extrême à l'attribut Sepal.Width(cm) à l'itération 2 pour lui donner un poids important (loin de la valeur médiane). Le même phénomène se produit dans les autres itérations. Quand au modèle SVM-linéaire, la suppression d'un attribut à également pour effet d'affecter un poids plus important à l'attribut redondant correspondant (voir par exemple le passage de l'itération 1 à 2 ou de l'itération 3 à 4 sur la figure 10.6), mais le nouveau poids n'est pas assez fort pour passer devant celui des autres attributs. Cette différence provient vraisemblablement du fait que SVM-logRatio prend en considération les informations (i.e. les attributs) par paire, ce qui lui permet de mieux saisir la redondance entre eux, alors que le noyau linéaire distingue les informations. Une autre manière de voir cela est de remarquer que lorsque l'on supprime un attribut, un double effet se produit dans SVMRFE-logRatio : d'une part les coefficients changent, et d'autre part la position de la valeur médiane par rapport à laquelle sont calculés les scores à tendance à s'écarter de sa position antérieure. A l'inverse, dans SVMRFE-linéaire, seul la valeur absolue des coefficients change, mais pas la valeur de référence qui reste à 0. Ce double effet pourrait expliquer que la suppression d'un attribut est un impact plus important sur les modèles SVM-logRatio que sur les modèle SVM-linéaire comme nous l'avons observé.

L'autre chose à voir sur cette figure est l'importance de la boucle RFE (i.e. de l'élimination successive des attributs) pour aboutir au résultat idéal de SVMRFE-logRatio. En effet, remarquez comme les modèles de la figure 10.6 sont symétriques à l'itération 1, dans le sens où les attributs exprimés dans l'unité centimètre obtiennent les mêmes valeurs que ceux exprimés dans l'unité millimètre. En conséquence, si nous nous étions seulement basé sur ces modèles pour produire l'ordonnancement d'attributs sans effectuer de boucle RFE, les attributs redondants auraient reçus des rangs similaires. C'est donc la suppression et la mise à jour des modèles logRatio-SVM qui ont permis à logRatio-SVMRFE de placer un attribut redondant en queue de classement et de propulser son "frère" en tête.

La figure 10.7 dévoile les résultats pour $\text{iris}_{(\delta_{\text{mm}}=0,\delta_{\text{cm}}=0.5)}$ qui n'a que les attributs cm de bruités. Le comportement espéré des algorithmes de sélection d'attributs sur ce jeu de données est donc qu'ils éliminent les attributs cm, et conserve les attributs mm qui contiennent les informations originales (non bruitées). Noter que même si les données originales contenaient déjà du bruit nous attendons également ce comportement, car les attributs exprimées en cen-

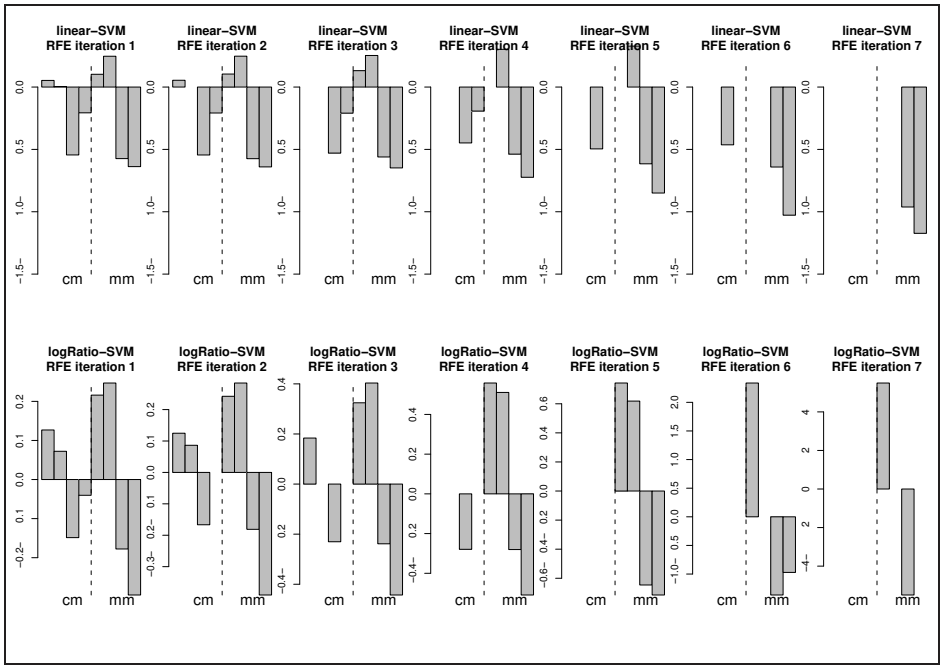


FIGURE 10.7 – Modèles SVM-linéaire et SVM-logRatio produits par SVMRFE-linéaire et SVMRFE-logRatio sur $\text{iris}_{(\delta_{\text{mm}}=0,\delta_{\text{cm}}=0.5)}$. Les attributs sont ordonnés de gauche à droite selon la séquence : Sepal.Length(cm), Sepal.Width(cm), Petal.Length(cm), Petal.Width(cm), Sepal.Length(mm), Sepal.Width(mm), Petal.Length(mm), Petal.Width(mm).

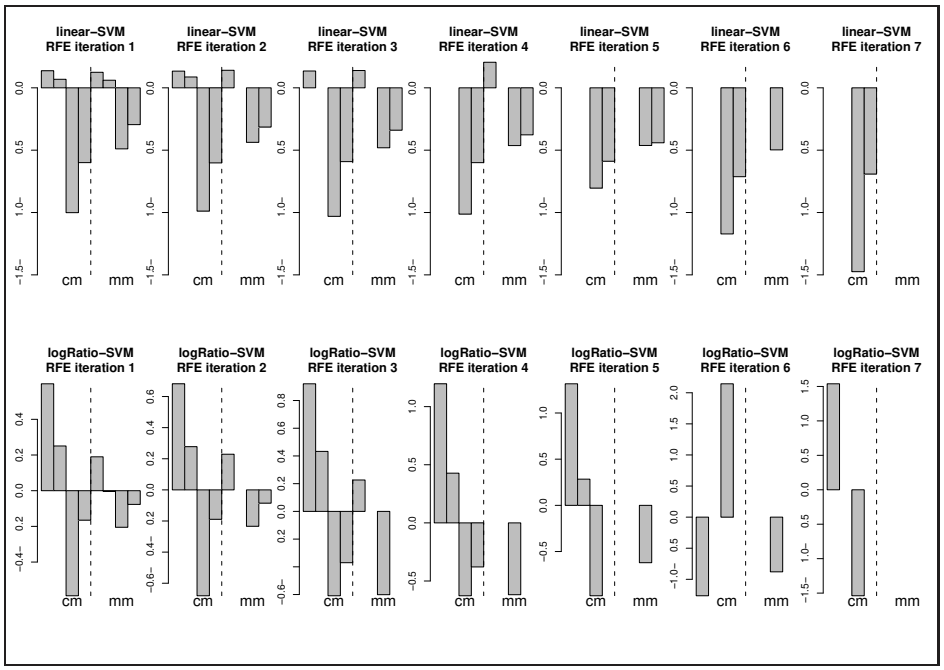


FIGURE 10.8 – Modèles SVM-linéaire et SVM-logRatio produits par SVMRFE-linéaire et SVMRFE-logRatio sur $\text{iris}_{(\delta_{\text{mm}}=5,\delta_{\text{cm}}=0.5)}$. Les attributs sont ordonnés de gauche à droite selon la séquence : Sepal.Length(cm), Sepal.Width(cm), Petal.Length(cm), Petal.Width(cm), Sepal.Length(mm), Sepal.Width(mm), Petal.Length(mm), Petal.Width(mm).

timètre n'apporte quand même aucune information supplémentaire aux attributs exprimées en millimètre puisqu'ils sont des répliques encore plus bruité de ces attributs. La table 10.3 montre qu'encore une fois, SVMRFE-logRatio a le comportement parfait, c'est à dire qu'il range les quatre attributs mm en haut de son ranking, alors que SVMRFE-linéaire y a placé un attribut avec l'unité cm. Cette défaillance de SVMRFE-linéaire peut venir d'une erreur de normalisation de l'attribut. En effet, l'ajout de bruit sur les valeurs exprimées en cm fait que leurs déviations standards sont supérieures à celles des attributs mm, et conduit à une normalisation différente des deux informations. Les poids du modèles sont influencés en conséquence, et l'interprétation des modèles est biaisée, ce qui conduit à une erreur de ranking. SVMRFE-logRatio ne souffre pas de ce problème car les modèles construits sont identiques quelle que soit la normalisation effectuée.

Finalement, les résultats pour $\text{iris}_{(\delta_{\text{mm}}=5,\delta_{\text{cm}}=0.5)}$ sont fourni figure 10.8. Dans ces données, les attributs cm et mm sont bruités de manière équivalente car une déviation standard de 5mm équivaut à une de 0.5cm. En conséquence, les deux informations exprimées dans des unités différentes se complètent car en les com-

binant il est possible d'améliorer la précision de la mesure. Aucun comportement spécifique des algorithmes de sélection d'attributs n'est donc attendu à priori car tous les attributs sont potentiellement informatifs. Par exemple, il est correct que SVMRFE-logRatio cherche à combiner P.L(cm) et P.L(mm) (table 10.3) pour améliorer la précision de la mesure sur la longueur du pétale. Par contre, il faut noter que sur ce jeu de données, on s'attend à ce que les attributs qui apportent la même information soient standardisés par le même facteur, car leur bruit est identique. Or, ce bruit perturbe l'estimation de la déviation standard ce qui résulte en une normalisation différente. Par exemple S.L(cm) a une déviation standard de 1.3, alors que celle de S.L(mm) est de 9.8. Cette dernière devrait être de 13 pour être équivalente à la première. Comme précédemment, ces différences de normalisation peuvent perturber l'interprétation des modèles SVMRFE-linéaire, mais pas celle des modèles SVMRFE-logRatio.

10.3.3 Bilan des expériences sur Iris

Le bilan de ces expérimentations dérivées d'Iris montrent clairement les avantages qui sont offerts par SVMRFE-logRatio et les faiblesses de SVMRFE-linéaire pour la sélection des attributs redondants. En tout cas, les résultats suggèrent que l'interprétation que nous faisons des modèles SVM-logRatio se rapproche d'avantage de la réalité attendue par les utilisateurs que celle des modèles SVM-linéaire. Ceci viendrait d'une part de la représentation des données dans logRatio qui le rend robuste à la normalisation des données et qui permet de mieux détecter la redondance d'information grâce à la combinaison deux à deux des attributs. Cependant, deux questions restent en suspens. La première concerne la pertinence des attributs sélectionnées car nous avons principalement concentré nos observations sur l'élimination de la redondance, or il reste à vérifier si les sous-ensembles d'attributs sélectionnés permettent de prédire efficacement la classe des instances [3]. La seconde interrogation porte sur les facultés du noyau logRatio à traiter des données de plus grande dimension que Iris. Ces questions sont justement l'objet de la section suivante, dans laquelle est étudiée l'erreur de classification engendrée par les sous-ensembles d'attributs sélectionnés avec le noyau logRatio sur des jeux de données couvrant des applications réelles de grande dimension.

10.4 Expériences sur données réelles

Nos expérimentations sur les jeux réelles portent sur quinze jeux de données pour la plupart d'origine biologique, qui concernent les domaines de la spectrométrie de masse, des puces à ADN, et des documents textuelles. Tous définissent un problème de classification binaire et ont en commun une forte dimension des

3. Signalons que pour Iris, nous avons réalisé plusieurs expériences qui n'ont montré aucune différence statistiquement significative entre SVMRFE-logRatio et SVMRFE-linéaire (résultats non fournis)

	#Positifs	#Négatifs	#Attributs	Data Type
Alt	1425	2732	2112	Textmining
Structure	2621	927	2368	Textmining
Disease	2606	631	2376	Textmining
Function	3089	818	2708	Textmining
Dexter	300	300	20000	Textmining
Breast	46	51	24481	Genomic, Microarray
StrokeRaw	101	107	28664	Proteomic, Unpreprocessed MS
OvarianRaw	162	91	15154	Proteomic, Unpreprocessed MS
ProstateRaw	69	253	15154	Proteomic, Unpreprocessed MS
StrokeRawGt2000	101	107	23795	Proteomic, Unpreprocessed MS
OvarianRawGt2000	162	91	10361	Proteomic, Unpreprocessed MS
ProstateRawGt2000	69	253	10361	Proteomic, Unpreprocessed MS
Stroke	101	107	172	Proteomic, Preprocessed MS
Ovarian	162	91	385	Proteomic, Preprocessed MS
Prostate	69	253	390	Proteomic, Preprocessed MS

TABLE 10.4 – Caractéristiques des différents jeux de données. Les colonnes #Positifs et #Négatifs donnent le nombre d'instances de la classe positive et le nombre d'instance de la classe négative.

données avec plus de 100 attributs par instance. La table 10.4 résume les caractéristiques de chacun des jeux de données, et plus de détails sont fournis sur dans la sous-section 10.4.1.

Les expériences réalisées avec ces données visent deux objectifs. Dans un premier temps, elles visent à étudier les facultés de logRatio à capturer l'information pertinente contenue dans les données de grande dimension, ce qui est fait en mesurant les performances de prédiction de logRatio-SVM sur les jeux de données (sous-section 10.4.2). Dans un second temps, elles visent à étudier la qualité des attributs sélectionnés par SVMRFE-logRatio pour en déduire la qualité de notre interprétation des modèles (sous-section 10.4.3).

10.4.1 Les données

Les caractéristiques des jeux de données utilisés dans nos expériences sont résumées table 10.4. La plupart d'entre eux ont servis dans nos travaux sur la stabilité (section 9.1.3 et [59]), et ceux sur l'analyse de la qualité des prétraitements en spectrométrie de masse [90]. Cependant, nous rappelons dans cette section l'origine de ces données et décrivons les pré-traitements spécifiques que nous leur avons appliqué pour pouvoir étudier le noyau logRatio.

StrokeRaw, OvarianRaw, et ProstateRaw sont des jeux de données contenants des informations de spectrométrie de masse que nous avons déjà utilisé dans la première partie de la thèse. Chaque instance est le spectre de masse complet d'un patient dans lequel l'intensité des pics reflète l'expression d'une partie des protéines contenus dans son corps. Le but est de déceler dans ces données un petit sous-ensemble de bio-marqueurs qui puissent différencier les patients

sains (les négatifs) des patients souffrants d'une maladie (les positifs), ce qui correspond exactement à l'objectif visé par la sélection d'attributs. OvarianRaw [86] (version 08-07-02) et ProstateRaw [87] sont des données publiquement téléchargeables depuis internet, et qui concernent le cancer des ovaires et de la prostate. StrokeRaw [59] est une jeu de donnée privé de l'hôpital de Genève au sujet des accidents vasculaires cérébraux. Le seul pré-traitement appliqué à ces données par rapport aux données originales est la suppression de la ligne de base selon la méthode décrite dans [90].

Stroke, Ovarian et Prostate sont des jeux de données dérivés de StrokeRaw, OvarianRaw et ProstateRaw après détection et alignement des pics selon le pré-traitement décrit dans [90] (et dans partie de ce document relatif au pré-traitement des spectres de masse). Le principal paramètre de ce processus contrôle le rapport signal/bruit minimum qu'un pic doit avoir pour être détectable. Nous avons fixé cette valeur à 2.5 pour générer les trois datasets comme le suggère les résultats de l'article, et nous n'avons conservé que les pics avec un m/z supérieur à 2000 Dalton.

StrokeRawGt2000, OvarianRawGt2000, et ProstateRawGt2000 sont exactement identique à StrokeRaw, OvarianRaw, et ProstateRaw, avec les informations correspondant aux masses inférieures à 2000 Daltons supprimées pour que l'information se rapproche de celle contenu dans Stroke, Ovarian et Prostate.

Le jeu de données Breast contient quand à lui des données de gènes exprimés dans le corps de patientes saines (les négatifs) et de patientes atteintes d'un cancer du sein (les positifs). Il est librement téléchargeable depuis le site de Kent Ridge Bio-medical Data Set Repository (`http://sdmc.lit.org.sg/GEDatasets/Datasets.html`). Les données contiennent des valeurs négatives car elles ont subit une transformation logarithmique ($\log_{10}(x)$), pour rétablir les valeurs positives originales, nous leur appliquons la transformation exponentielle inverse (10^x).

Le jeu de données Dexter provient du challenge NIPS03 sur la sélection d'attributs (`http://www.clopinet.com/isabelle/Projects/NIPS03`). Les instances de celui-ci représentent des documents textuels rangées dans deux catégories. Les attributs représentent les mots des textes, ils sont normalisés pour varier dans l'intervalle [0..1000] avant d'être arrondies à l'entier le plus proche. Cependant, seulement 9947 sur les 20000 sont des attributs réelles, les 10053 autres sont générées aléatoirement. L'identité des "vrais" attributs est laissée à la discrétion des auteurs du challenge qui sont les seuls capable de déterminer le pourcentage de vrais/faux attributs contenus dans un sous-ensemble spécifique. Originellement, les données sont divisées en trois partis : un ensemble d'entraînement avec 300 instances étiquetées, un ensemble de validation avec les mêmes caractéristiques, et un ensemble de test sans étiquette mais dont les prédictions sont à soumettre pour participer au challenge. Le jeu de donnée que nous utilisons est construit par concaténation de l'ensemble d'entraînement et de l'ensemble de validation.

Enfin, les quatre derniers jeux de données se rapportent, comme Dexter, au textmining. Dans ceux-ci, le but est de déterminer si des phrases se rapportent à un sujet donné. Disease concerne des phrases qui lient des protéines à une

maladie ; Function contient des phrases qui lient des protéines à une fonction ; Structure se rapporte aux protéines et à leur structure ; et Alt aux protéines produites par épissage alternatif [76]. Chaque attribut est associé à un mot du langage, et chaque instance décrit une phrase de texte par une représentation tf.idf (*term frequency . inverse document frequency*) qui est une mesure couramment employé en text mining pour évaluer l'importance d'un mot dans un ensemble de documents. Ce choix de représentation génère des jeux de données avec des valeurs positives et de nombreuses valeurs nulles car les phrases sont en générale courte (en moyenne une douzaine de termes) alors que le vocabulaire du langage très riche.

Les valeurs numériques des jeux de données qui sont décrits ci-dessus sont toutes positives, mais certaines sont aussi nulles, ce qui rend impossible leur traitement par le noyau logRatio. Soulignons en particulier que les données textuelles sont très "clairsemées" : les instances de Alt, Function, Structure et Disease n'ont en moyenne que douze attributs non nulles. Pour pouvoir y appliquer le noyau logRatio, nous avons donc éliminé les valeurs nulles de tous les jeux de données en les remplaçant par une valeur positive, ϵ, petite par rapport aux autres valeurs que contiennent ces jeux de données. Plus exactement, toutes les valeurs inférieures au seuil ϵ sont remplacées par ϵ (voir section10.1.2). Pour tous les jeux de données, la valeur $\epsilon = 0.01$ est adoptée (après que nous ayons vérifié que c'est une valeur petite par rapport à toutes les autres valeurs).

10.4.2 Classification avec SVM-logRatio

Dans cette section, nous allons estimer les performances de classification de SVM-logRatio sur quinze jeux de données et les comparer à celles obtenues avec un noyau linéaire (SVM-lin) et un noyau polynomial de degré 2 (SVM-poly). Notre objectif ici est de déterminer si il y a une représentation des données induite par un des noyau qui est plus appropriée que les autres pour la classification des données et d'en comprendre les raisons. Notre étude est facilité par le fait que les noyaux partagent de nombreuses similitudes ce qui nous permet d'identifier précisément les raisons qui expliquent les différences entre eux. En plus de cela, nous examinons également l'impact que peu avoir une transformation logarithmique des données sur les performances de SVM-lin et SVM-poly pour rapprocher encore leurs similitudes avec logRatio (voir section 10.1).

Les résultats obtenues sont fournies table 10.5. Ils sont excellents puisque la table 10.5 indiquent que les performances de logRatio-SVM ne sont jamais en-dessous des autres, et qu'elles sont même toujours égales ou significativement supérieures aux autres. logRatio est particulièrement efficace sur les données textuelles où il affiche à chaque fois les plus petites erreurs. Sur Alt, il commet même significativement moins d'erreurs que tous les autres classificateurs. Un résultat surprenant lorsque l'on sait qu'au départ les données textuelles contiennent de nombreuses valeurs nulles que logRatio ne traite pas naturellement.

Les raisons de ces bons résultats sont multiples. Il ressort tout d'abord que la transformation logarithmique des informations joue un rôle dans l'amélioration des résultats car la comparaison des colonnes (logLin vs lin) et (logPoly vs

Noyau	logRatio	lin	logLin	poly	logPoly
Alt	10.17%	**13.80%**	**12.53%**	**28.57%**	**16.55%**
Disease	17.11%	**19.74%**	17.76%	**19.15%**	18.13%
Structure	18.09%	19.44%	19.41%	**22.35%**	19.89%
Function	19.17%	19.98%	19.91%	20.78%	20.01%
Dexter	6.16%	**12.00%**	6.33%	**44.16%**	6.30%
Breast	28.86%	29.89%	32.98%	40.20%	41.23%
StrokeRaw	18.26%	15.38%	16.34%	**36.05%**	**39.42%**
OvarianRaw	0%	1.18%	0.39%	**7.50%**	0.79%
ProstateRaw	7.45%	6.52%	7.76%	10.24%	10.24%
StrokeRawGt2000	18.26%	15.38%	17.30%	**45.67%**	**43.26%**
OvarianRawGt2000	1.58%	3.16%	2.37%	**7.11%**	**5.53%**
ProstateRawGt2000	13.35%	11.80%	12.73%	14.59%	16.14%
Stroke	22.11%	21.15%	21.63%	31.73%	**38.94%**
Ovarian	1.18%	1.97%	1.58%	**8.69%**	**7.90%**
Prostate	13.97%	14.59%	13.66%	15.52%	15.21%

TABLE 10.5 – Erreur de classification de l'algorithme SVM estimé par 10-fold-cross-validation pour différentes fonctions noyaux. Les erreurs significativement supérieures à l'erreur logRatio sont montrés en gras et il n'y a aucune erreur significativement inférieure (test de significativité de McNemar avec seuil à 0.05 [27]).

poly) montre plusieurs baisses d'erreurs significatives. La combinaison deux à deux des attributs dans logRatio semble également importante car les performances qu'affiche logRatio sont meilleures que celles de SVM-lin et SVM-logLin qui ne les combines pas. Par contre, les attributs ne doivent pas être combiné n'importe comment, car SVM-poly et SVM-logPoly qui font le produit des attributs obtiennent des performances catastrophiques. Il serait donc plus judicieux de combiner les attributs deux à deux en considérant leur ratio car ceci annule les éventuelles erreurs de normalisation, plutôt que considérer leur produit qui a tendance à les amplifier.

Les bons résultats de logRatio sur les données textuelles proviennent aussi certainement de problèmes relatifs à la normalisation des données "clairsemées", c'est à dire avec beaucoup de valeurs manquantes ou nulles. La standardisation des attributs n'est certainement pas adaptée aux nombreuses valeurs ϵ contenues dans ces données et qui ont tendance à fausser l'évaluation de la déviation standard nécessaire à la standardisation. Dans ce cas, la robustesse de logRatio face à la normalisation est sans doute un atout face aux autres approches. Signalons aussi que le nombre d'instances dans les données textuelles est beaucoup plus élevé que dans les données d'expression ce qui permet de distinguer des algorithmes de classification qui ont des performances assez proches. Sur les données d'expression, sans-doute aurions-nous pu observer le même phénomène si les instances avaient été plus nombreuses et moins uniformes comme c'est souvent le cas en pratique : acquisition à des temps différents, en des lieux différents, et

sur des instruments différents.

Toutes les conditions sont donc réunies pour espérer un comportement performant du noyau logRatio pour la sélection d'attributs : 1) il produit les meilleures performances de classification sur les jeux de données réelles ; 2) il a parfaitement su éliminer la redondance dans Iris (section 10.3) 3) l'interprétation des modèles SVM-logRatio devrait être plus correcte que celle des modèles linéaires pour extraire la pertinence d'un attribut, vu qu'elle est indépendante de la normalisation des données et étant donnée les résultats observées sur Iris. Pour vérifier cette hypothèse, la section suivante est consacrée aux algorithmes de sélection d'attributs à base des noyaux logRatio et linéaire, qui devraient engendrer des résultats d'autant meilleurs que l'interprétation des modèles est correcte. Les modèles SVM à base de noyau polynomial fournissant les moins bonnes performances, et la sélection d'attributs étant plus difficile à réaliser avec ce noyau, nous l'avons ignoré dans la suite (noter cependant que nous pourrions la réaliser en suivant les propositions de [43]).

10.4.3 Sélection d'attributs avec SVMRFE-logRatio

Dans cette section, nous tâchons d'évaluer la qualité des sous-ensembles d'attributs sélectionnés par l'algorithme SVMRFE-logRatio et de la comparer aux deux algorithmes de références SVMRFE-linéaire et SVMRFE-logLin (ce dernier correspond à SVMRFE-linéaire après transformation des données en leur logarithme). Cette comparaison est faite à trois niveaux différents. Dans un premier temps, nous analysons les performances de classification qu'il est possible d'obtenir avec les sous-ensembles d'attributs sélectionnés par chaque méthode. Comme précédemment, ces performances de classifications sont estimées par validation croisée (*10 folds cross validation*) et comparée à l'aide d'un test statistique de McNemar's. Dans un second temps, nous nous intéressons à la stabilité des algorithmes de sélection d'attributs. Enfin, nous nous attardons plus spécifiquement sur les données de spectrométrie de masse avec lesquelles nous tacherons d'évaluer l'impact du pré-traitement sur la stabilité des sous ensembles d'attributs sélectionnés par les différentes méthodes. Les trois paragraphes qui suivent traitent de ces sujets.

Performance de classification

Pour comparer les performances de classification des algorithmes de sélection d'attributs, 10% des attributs sont éliminés à chaque itération RFE, et les ordonnancements produits par SVMRFE-logRatio, SVMRFE-linéaire et SVMRFE-logLin sont utilisés pour sélectionner les n attributs les plus pertinents pour chacun. Les algorithmes de classifications SVM-logRatio, SVM-linéaire et SVM-logLin sont ensuite entraînés en limitant l'apprentissage aux n attributs les mieux classés par chaque méthode, et testé sur un ensemble de validation distinct. Le tout est itéré dix fois de manière à opérer une estimation d'erreur par validation croisé. Les erreurs de classification obtenues constituent une estimation de la qualité des sous-ensembles d'attributs sélectionnés car plus

elles sont petites et plus l'information que l'on peut extraire des sous-ensembles sélectionnés pour prédire la classe est importante.

Pour clarifier les résultats, nous limitons le nombre de combinaisons possibles entre algorithmes de sélection d'attributs et algorithmes de classification aux trois cas SVMRFE-logRatio + SVM-logRatio, SVMRFE-lin + SVM-lin et SVMRFE-logLin + SVM-logLin. Chaque algorithme de sélection est ainsi associé à l'algorithme de classification dont il interprète le modèle, et pour lequel il est censé sélectionner les attributs qui lui correspondent. Aussi, toujours par souci de clarté, nous dissocions les jeux de données relatifs à la spectrométrie de masse des autres dans la présentation des résultats, de plus les résultats des trois jeux de données StrokeRaw, OvarianRaw et ProstateRaw qui sont proches de ceux obtenus avec StrokeRawGt2000, OvarianRawGt2000 et ProstateRawGt2000 ne sont fournis qu'en annexe D (figure D.1). Pour le reste, l'ensemble des résultats peuvent être consulté sur les figures 10.9 et 10.10. Chacune des deux figures contient six cadres de deux tracés : celui du haut montre les trois courbes de l'évolution de l'erreur de classification en fonction de la taille (n) des sous-ensembles d'attributs sélectionnés ; celui du bas montre les deux courbes correspondants au test de significativité de McNemar [27] contre les performances du noyau logRatio, c'est à dire que lorsque cette courbe est en-dessous de la ligne 0.05, nous pouvons considérer que les erreurs sont significativement différentes. Signalons enfin que pour les jeux de données contenants beaucoup d'instances (Alt, Disease, Structure et Function), les performances du noyau logRatio pour un nombre d'attributs inférieure à 200 n'ont pu être obtenu pour des raisons de temps d'apprentissage bizarrement excessifs des modèles SVM. L'algorithme implémenté dans *libsvm* pour trouver l'hyperplan séparateur de marge maximal semble diverger (ou converge t'il trop lentement ?) lorsque moins de 200 attributs sont sélectionnés, alors que au-dessus les temps d'exécution sont raisonnables.

Les performances de classifications observées sur les figures 10.9 et 10.10 sont logiquement dans la continuité de celles de la table 10.5 (les valeurs de la table se retrouvant à l'extrême droite des courbes correspondantes). On note en particulier que les performances de logRatio sont intéressantes sur les jeux de données textuelles. Les performances de Alt montre même qu'après sélection de seulement 30% des attributs (i.e. environ 600 sur les 2112 de départ), logRatio est toujours significativement plus performant que les deux autres méthodes. Par contre ses performances restent stable au fur et à mesure que le nombre d'attributs diminue quand celles des autres ont tendance à s'améliorer jusqu'à rejoindre le niveau de performance de logRatio. Sur Disease et Structure, le comportement de logRatio est également très bon car il est stable dans sa bonne performance, et il suffirait de baisser légèrement le seuil de confiance du test de significativité pour considérer son erreur significativement inférieure aux deux autres.

A coté de ces résultats assez satisfaisants les performances de classification de logRatio-SVMRFE sont dans l'ensemble très proches de celles des noyaux linéaires. A l'exception du jeu de donnée StrokeRawGt2000 ou logRatio montre une défaillance, la plupart des différences entre les algorithmes ne sont pas si-

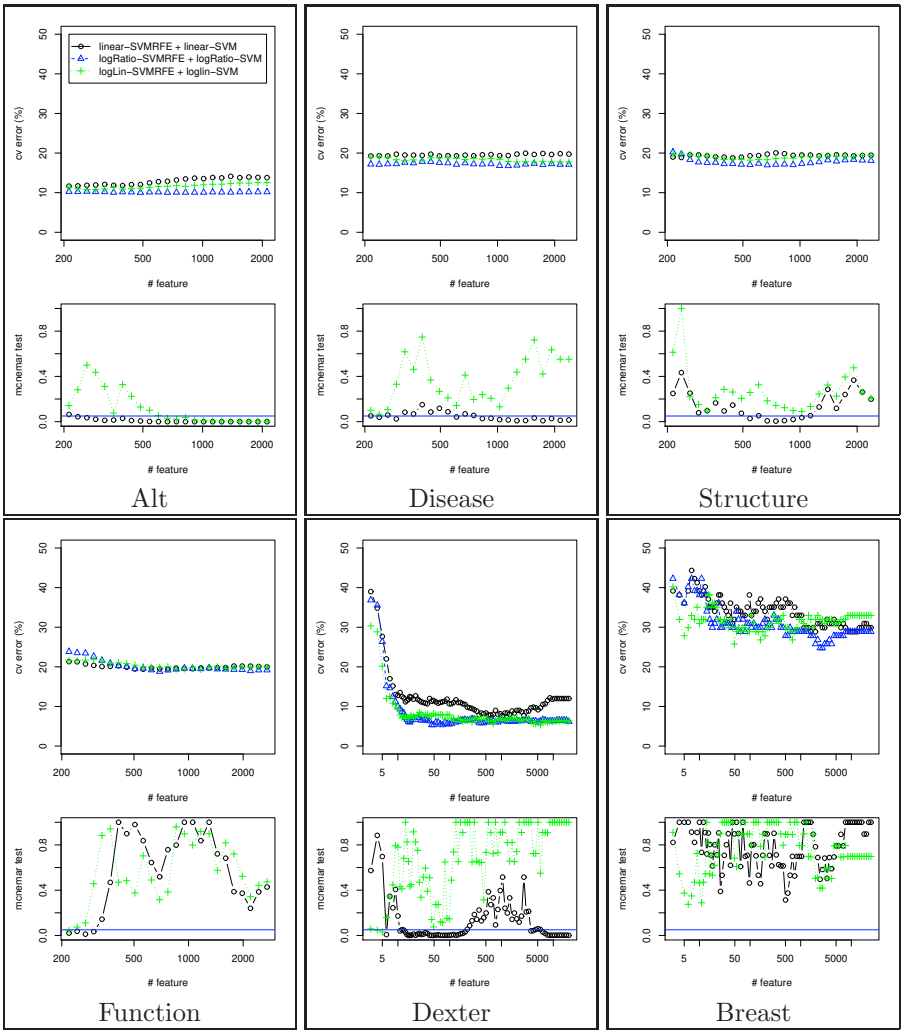


FIGURE 10.9 – Performances de classification sur les données autre que MS. Dans chacun des six cadres, les trois courbes du haut montrent l'évolution de l'erreur de classification en fonction de la taille (n) des sous-ensembles d'attributs sélectionnés par chaque méthode de sélection d'attribut ; les deux courbes du bas correspondent au test de significativité de McNemar [27] contre les performances du noyau logRatio – lorsque ces courbes sont en-dessous de la ligne 0.05, nous pouvons considérer les erreurs comme significativement différentes.

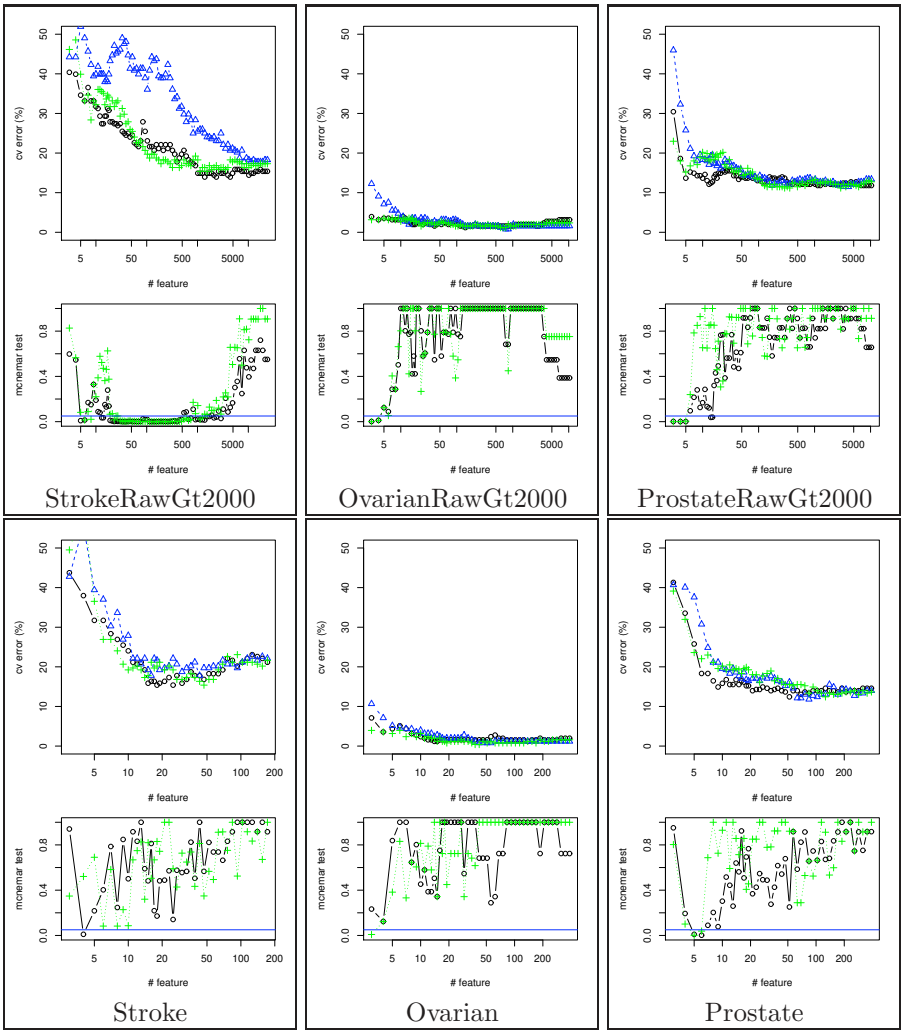


FIGURE 10.10 – Performances de classification sur les données MS. Dans chacun des six cadres, les trois courbes du haut montrent l'évolution de l'erreur de classification en fonction de la taille (n) des sous-ensembles d'attributs sélectionnés par chaque méthode de sélection d'attribut ; les deux courbes du bas correspondent au test de significativité de McNemar [27] contre les performances du noyau logRatio – lorsque ces courbes sont en-dessous de la ligne 0.05, nous pouvons considérer les erreurs comme significativement différentes.

gnificative. Il semble cependant que logRatio ait une tendance à être moins performant que les noyaux linéaires pour la sélection de très petits sous-ensembles qui comportent moins de vingt attributs (voir les courbes des jeux de données Dexter, OvarianRawGt2000, Ovarian et ProstateRawGt2000 et Prostate qui remonte sur l'extrême gauche des graphes). Cette fois, par contre, cela vient plus d'une détérioration des performances de logRatio que d'une amélioration de celles des noyaux linéaires, et remarquez aussi que au dessus de ce seuil les performances de logRatio-SVMRFE sont assez constantes. Trois raisons peuvent expliquer ce phénomène. La première serait que notre façon d'ordonner les attributs en fonction des modèles logRatio est inapproprié, et a pour conséquence l'élimination de variables pertinentes qui ne devraient pas l'être. Cette explication est cependant peu vraisemblable car elle n'explique pas les bons résultats qui sont obtenus sur des sous-ensembles de taille supérieure à vingt attributs. La deuxième explication, plus plausible, est que dans ces jeux de données, les attributs sont individuellement informatifs. En conséquence, parmi toutes les combinaisons possibles des vingts attributs sélectionnés, seulement dix paires mutuellement disjointes contiennent l'information et toutes les autres sont inutiles. logRatio doit donc se contenter de ces dix informations quand le noyau linéaire peut exploiter les vingts informations dans leur individualité. Enfin, la troisième possibilité est que les vingts attributs sélectionnés par logRatio-SVMRFE interagissent très fortement les uns avec les autres (vu que logRatio-SVMRFE est conçu pour se concentrer sur les interactions attributs-attributs) et que, l'ensemble des interactions sont utiles pour prédire efficacement la classe dans les modèles SVM-logRatio. En conséquence, si un seul des attributs est éliminé, tout le système est perturbé. Il faut enfin être prudent avant de déduire de ces résultats que logRatio-SVMRFE est moins performant que linear-SVMRFE sur les jeux de données de dimension inférieure à vingt attributs, car rappelons qu'il a montré un comportement exemplaire sur Iris qui ne contient que huit attributs.

Stabilité de la sélection d'attributs

Les expériences sur la performance de classification que nous venons de décrire montrent au final peu de différences entre les algorithmes de sélection d'attributs (si ce n'est sur Alt et StrokeRawGt2000). Nous essayons donc maintenant de les distinguer selon un autre critère qui est leur stabilité. La stabilité est une indication importante de la qualité des algorithmes de sélection d'attribut, car nous attendons d'eux qu'ils extraient des sous-ensembles d'attributs identiques lorsque les données sont légèrement perturbées/bruitées (voir chapitre 9). Il est cependant facile de définir un algorithme de sélection d'attribut très stable dont les qualités sont discutables. Par exemple l'algorithme qui choisi les n premiers attributs d'un jeu de donnée est très stable, car il extrait à chaque fois les mêmes attributs, cependant, comme ils ne sont pas forcement pertinent pour le problème de classification posé, il est d'un intérêt limité. Cette exemple montre bien que la stabilité n'a qu'un rôle secondaire, et que l'on attend avant tout des algorithmes de sélection d'attributs qu'ils extrait des sous-ensembles d'attributs

contenants le maximum d'information pertinente. Comme nous avons vu que les algorithmes testés se valent plus ou moins et terme de classification, nous passons maintenant à la comparaison de leur stabilité.

Pour quantifier la stabilité de nos algorithmes de sélection d'attributs, nous reprenons les dix ordonnancements générés par les différents algorithmes SVMRFE lors de la validation croisée des expérimentations précédentes (au sujet de la performance de classification des algorithmes de sélection d'attributs). Si l'algorithme étudié est stable, les attributs qu'il extrait dans les dix ensembles d'entraînement doivent être identiques. Nous décidons donc de comparer deux à deux les dix ordonnancements produits par un algorithme, et de calculer la proportion moyenne d'attributs qu'ils ont en commun parmi les n situés en tête de classements (voir section 9.1.1). Ainsi, pour prendre un exemple, une stabilité de 50% quand $n = 60$ signifie que lorsque l'on compare les 60 premiers attributs des dix ordonnancements deux à deux, il y a en moyenne 30 attributs en commun. La figure 10.11 livre l'ensemble des résultats obtenus en affichant l'évolution de la stabilité des différents algorithmes de sélection d'attributs en fonction du nombre d'attributs sélectionné (n). Comme nous en avons déja discuté dans la section 9.1.1, cette mesure fourni un nombre qui résume bien l'image global de la stabilité de la sélection d'attributs, mais ce nombre ne reflète pas la stabilité dans son ensemble comme par exemple un coefficient de corrélation de Spearman. En revanche, la mesure que nous adoptons est facile à interpréter, et comme nous traçons l'évolution de cette mesure en fonction de n, on obtient au final une information qui se rapproche de celle que l'on obtient avec un coefficient de corrélation de Spearman. Nous utilisons le résultat de la section 9.1.2 pour tracer la courbe de stabilité de référence qui rappelons le est une droite, mais qui apparaîtra comme une courbe sur nos figure en raison de l'échelle logarithmique utilisée pour l'axe des abscisses.

Les résultats que montre la figure 10.11 semblent légèrement jouer en faveur de logRatio-SVMRFE pour la sélection de plus de vingt attributs. En effet, les courbes de logRatio-SVMRFE sont souvent un peu au dessus de celles de ses concurrents quand il reste plus d'une vingtaine d'attributs dans les jeux de données. Ces observations peuvent être faites sur Alt, Dexter, Breast et aussi, dans une moindre mesure, sur les données de spectrométrie de masse OvarianRawGt2000 et StrokeRawGt2000. Quand aux autres jeux de données, les différences de stabilité sont quasi inexistantes. En considérant ces résultats en complément de ceux sur la performance de classification, logRatio-SVMRFE apparaît comme le plus efficace des trois car il obtient des performances de classification significativement supérieures ou égales à celles de ses concurrents avec une meilleure stabilité sur Alt, Dexter, Breast et OvarianRawGt2000. Sa stabilité supérieure nous indique que logRatio-SVMRFE est capable de capturer des attributs plus représentatifs des données c'est à dire dont le motif est plus fréquemment reconnaissable tout en étant important pour la classification. De plus, sur les autres jeux de données, ses performances ne sont pas en dessous de celles des autres ni en terme de classification, ni en terme de stabilité. StrokeRawGt2000 fait cependant exception car logRatio-SVMRFE obtient des performances significativement en dessous de celles des autres. Pourtant la sta-

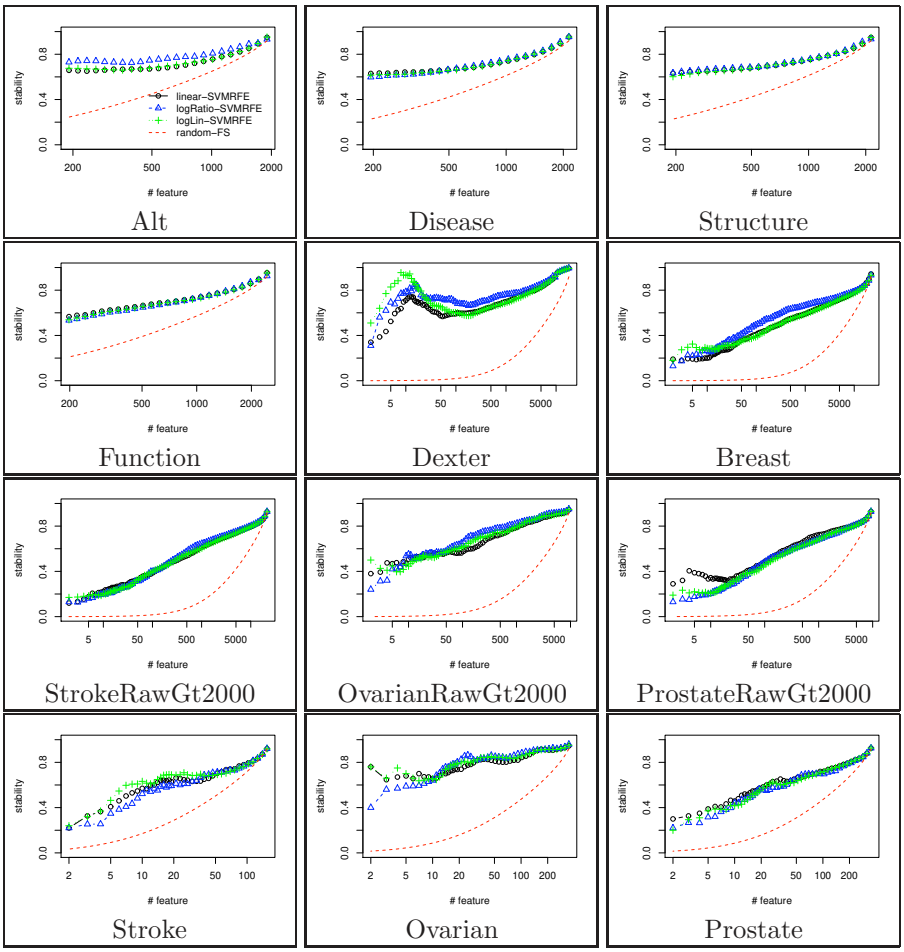


FIGURE 10.11 – Évolution de la stabilité des algorithmes de sélection d'attributs en fonction de la taille (n) des sous-ensembles d'attributs sélectionnés. 10% des attributs sont éliminés à chaque itération RFE. La courbe de référence en pointillé rouge montre la stabilité d'une sélection d'attribut qui choisie les attributs aléatoirement.

bilité qu'il affiche sur ces données est comparable (voir légèrement supérieure) à la leur, ce qui est curieux car on pouvait s'attendre à ce que logRatio-SVMRFE soit plus instable que les autres vu que les modèles logRatio-SVM sur lesquelles il repose ont du mal à capturer les attributs pertinents de StrokeRawGt2000.

En ce qui concerne la sélection de moins de vingt attributs, la stabilité de logRatio-SVMRFE se dégrade à l'image des résultats de classification déjà observées. Cette baisse de stabilité s'explique justement par la chute des performances de classification, car comme les modèles SVM-logRatio ont du mal à identifier les attributs pertinents pour la classification quand il y a peu d'attributs, les ordonnancements qui sont produits en interprétant ces modèles sont moins stables. De plus, l'instabilité est accentuée par la répétition des apprentissages dans les itérations des SVMRFE : les modèles et les attributs éliminés à chaque itération dépendants des modèles et des attributs éliminés dans les itérations précédentes, les instabilités sont répercutées d'une itération à l'autre et s'accumulent petit à petit. La répétition des apprentissages aurait donc un impact négatif sur la stabilité des algorithmes de sélection d'attributs (comme cela a déjà été remarqué dans [59]). Nous pouvons éviter ces répétions en effectuant la sélection d'attributs seulement en fonction du modèle SVM appris avec l'ensemble des attributs. Pour vérifier si cette simplification améliore les performances de la sélection d'attributs nous avons conduit la même série d'expériences que celles qui ont permis d'étudier la classification et la stabilité de logRatio-SVMRFE en utilisant les ordonnancements déduit de l'apprentissage des modèles SVM sur l'ensemble des attributs. Les résultats (fournis dans l'annexe E) montre effectivement une net amélioration de la stabilité mais elle se fait au détriment des performances de classification (les erreurs de classification des figures E.1 et E.2 augmentent beaucoup plus rapidement quand le nombre d'attribut diminue que celles des figures 10.9 et 10.10), ce qui rend la simplification moins intéressante que l'original. Nous observons aussi sur ces résultats que la stabilité du noyau logRatio est plus compétitive pour la sélection de moins de vingt attributs (les courbes de stabilité sont situées au dessus des autres dans plusieurs jeux de données), ce qui tend encore à montrer que les modèles SVM-logRatio (qui ont servis à produire l'ordonnancement) sont plus stables que les autres quand ils sont entraînés sur la totalité des données.

Stabilité face au pré-traitement MS

Les analyses précédentes ont mis en valeur les qualités de logRatio-SVMRFE dans le contexte général de la sélection d'attribut en *datamining*. Nous nous concentrons maintenant plus spécifiquement sur le problème de la recherche des biomarqueurs en spectrométrie de masse, car dans ce domaine, des exigences supplémentaires peuvent être formulées. En particulier, face à l'ampleur de l'effort nécessaire au pré-traitement des données MS, nous sommes en droit de nous intéresser aux méthodes d'apprentissage qui ne serait pas sensible au pré-traitement MS [75]. A priori, logRatio part avec un petit avantage car il est conçu pour être insensible à la normalisation des spectres, mais le pré-traitement inclut aussi une importante phase de détection des pics pour laquelle logRatio

ne garantit rien. Or nous nous attendons à ce que les biomarqueurs identifiés par les algorithmes de sélection d'attributs dans les données brutes correspondent à des pics des spectres de masse, comme si l'algorithme de sélection d'attribut opérait en interne un pré-traitement des spectres de masse. Plus contraignant encore, l'idéal serait un algorithme de sélection d'attributs capable d'extraire les mêmes biomarqueurs dans les données brutes que dans les données pré-traitées. Pour vérifier si nos algorithmes de sélection sont sensibles au pré-traitement MS, nous comparons donc maintenant les attributs sélectionnés sur les données brutes de spectrométrie de masse (StrokeRawGt2000, OvarianRawGt2000, ProstateRawGt2000), avec ceux sélectionnés sur les données pré-traitées (Stroke, Ovarian, Prostate).

Pour cela, reprenons les expériences qui ont servis à évaluer les performances de classification des algorithmes SVMRFE et leur stabilité, et intéressons nous aux $n = 20$ attributs sélectionnés dans les dix validations croisées. Chaque attribut peut donc apparaître de 0 à 10 fois dans ces sous-ensembles, et il est raisonnable de supposer que les plus pertinents sont ceux qui apparaissent le plus grand nombre de fois. Les figures 10.12,10.13,10.14 permettent de visualiser ces informations et de confronter les attributs sélectionner sur les données brutes avec ceux sélectionnés sur les données pré-traitées : a l'emplacement qui correspond au m/z de chaque attribut, nous mettons en ordonnée positive le nombre de fois où celui-ci fait parti des $n = 20$ attributs sélectionnés sur les données brutes, et en ordonnée négative le nombre de fois ou il est apparu dans les données pré-traitées. Les tables contenants les valeurs qui ont servi à produire cette visualisation peuvent être consulté dans l'annexe F et sont susceptible d'intéressé le lecteur qui s'intéresse à la position des pics choisis par les méthodes. La première remarque que l'on peut faire sur les figures 10.12,10.13,10.14 est qu'il y a beaucoup plus de valeurs proches de -10 que de valeurs proche de +10, ce qui signifie que les modèles appris sur les données pré-traités sont plus stables que les modèles appris sur les données brutes. Cela n'a cependant rien d'étonnant car les données pré-traités contiennent beaucoup moins d'attributs que les données brutes, il est donc plus probable de sélectionner des sous-ensembles d'attributs stables. Si l'on compare maintenant chacun des trois cadres disposés horizontalement nous constatons que les différentes méthodes présentent des similitudes, mais aussi des différences notables. Les attributs que l'on retrouve dans les différentes méthodes sont les attributs qui sont incontournables car ils apportent un pouvoir de discrimination important. Par contre, les attributs qui sont souvent sélectionnés par une méthode, et jamais par les d'autres peuvent correspondre à plusieurs chose : il peut s'agir d'attributs que la méthode doit sélectionner pour corriger son biais naturel ; il peut aussi s'agir d'attributs qu'une des méthodes n'a pas identifié comme étant redondant alors que les autres oui ; enfin cela peut être un attribut qui n'est pertinent que si il est combiné d'une façon particulière avec un autre attribut et que seul la méthode en question est capable d'identifier. Mais dans tout les cas, il peu être intéressant de se concentrer un peu plus sur ce type d'analyse. Enfin, si nous comparons, à l'intérieur de chaque cadre, les attributs sélectionnés sur les données brutes aux attributs sélectionnés sur les données pré-traité, nous n'avons pas vraiment la

symétrie horizontal espéré qui nous garantirait que les modèles sont identiques, et qui nous éviterait d'effectuer le pré-traitement des données MS (hormis peut-être sur ovarian et ovarianRawGt2000).

Pour observer plus globalement l'influence du pré-traitement MS sur les algorithmes de sélection d'attributs sans nous limiter à la sélection de $n = 20$ attributs, les courbes de la figure 10.15 montrent le nombre de fois (en moyenne sur les 10 ordonnancements de la validation croisée) où les rapports masse/charges des n attributs sélectionnés sur les données pré-traitées coïncident avec les n attributs sélectionnés sur les données brutes (RawGt2000), avec une tolérance d'erreur de 1%. Les courbes sont dans l'ensemble assez basses puisque au mieux, l'on peut espérer que les modèles ne trouve dans les données brutes que 50% [4] des attributs sélectionnés dans les données pré-traités. On s'aperçoit que logRatio-SVMRFE est un peu en dessous des autres sur stroke et ovarian, ce qui suggère que les modèles qu'il construit changent d'avantage entre les données brutes et les données pré-traités que ceux des autres. Ceci pourrait s'explique par le fait que les modèles à base du noyau logRatio cherchent à combiner les attributs deux à deux, et que pour les données que nous possédons, il est préférable de considérer les attributs dans leur individualité. logRatio-SVMRFE cherche donc des attributs de référence avec lesquels combiner les autres pour refléter le mieux possible leur valeur de départ. C'est à dire que logRatio-SVMRFE à besoin d'attributs qui ont des valeurs le plus constantes possible sur tout le jeu de donnée pour les combiner avec les autres, or il s'avère que pour les données de spectrométrie de masse, ceux sont les régions qui n'ont pas de pics qui sont les plus constantes car elles ont toujours des valeurs proche de 0. Comme ces régions ne sont présentes que dans les données brutes, logRatio-SVMRFE ne les retrouve pas dans les données pré-traitées ce qui abaisse ses courbes de coïncidence (figure10.15). Rappelons que cela ne remet pas en cause les qualité de logRatio-SVMRFE pour analyser les données de spectrométrie de masse, mais plutôt, cela nous enseigne qu'il est préférable d'opérer un bon pré-traitement des données MS (en suivant les recommandation données dans dans la première partie de ce document) si l'on souhaite employer cet algorithme.

10.5 Bilan du noyau logRatio

Dans ce chapitre nous avons introduit le noyau logRatio qui présente des caractéristiques intéressantes pour traiter des données issues du domaine de la biologie. logRatio partage des similitudes avec le noyau linéaire, mais il met davantage l'accent sur les interactions attribut-attributs et sur les problèmes de normalisation. De plus, l'interprétation des modèles SVM appris avec logRatio est simple, elle permet de tirer des conclusions sur la redondance entre les attributs, et elle met en évidence les contraintes qui existent entre les coefficients

4. Ce pourcentage est obtenu par lecture des graphes aux abscisses $n = 20$, et où l'on voit qu'il y a au mieux 10 correspondances. Cela signifie qu'en moyenne parmi les 20 attributs sélectionnées dans les données pré-traités et dans les données brute par chaque méthode il y a 10 attributs en commun. Cela correspond à un pourcentage de $10/20 = 50\%$.

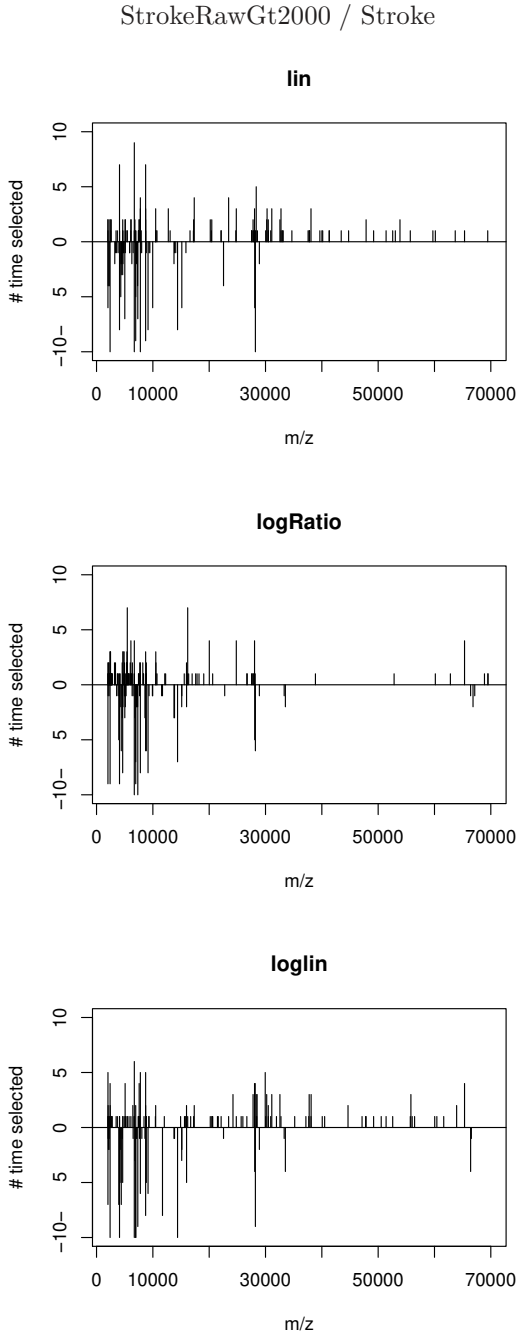


FIGURE 10.12 – Comparaison du nombre de fois ou un pic est sélectionné par SVMRFE sur les données brutes par rapport au nombre de fois ou il est sélectionné sur les données pré-traités. Il y a un tracé par noyau utilisé pour la sélection (linéaire, logRatio, loglin), et on limite la comparaison au $n = 20$ attributs les plus discriminants pour chaque méthode. L'axe des abscisse indique le m/z du pic concerné, la partie positive de l'axe des ordonnées indique le nombre de fois ou le pic est sélectionné dans les données brutes, la partie négative de l'axe des ordonnées indique le nombre de fois ou le pic est sélectionné dans les données pré-traités.

OvarianRawGt2000 / Ovarian

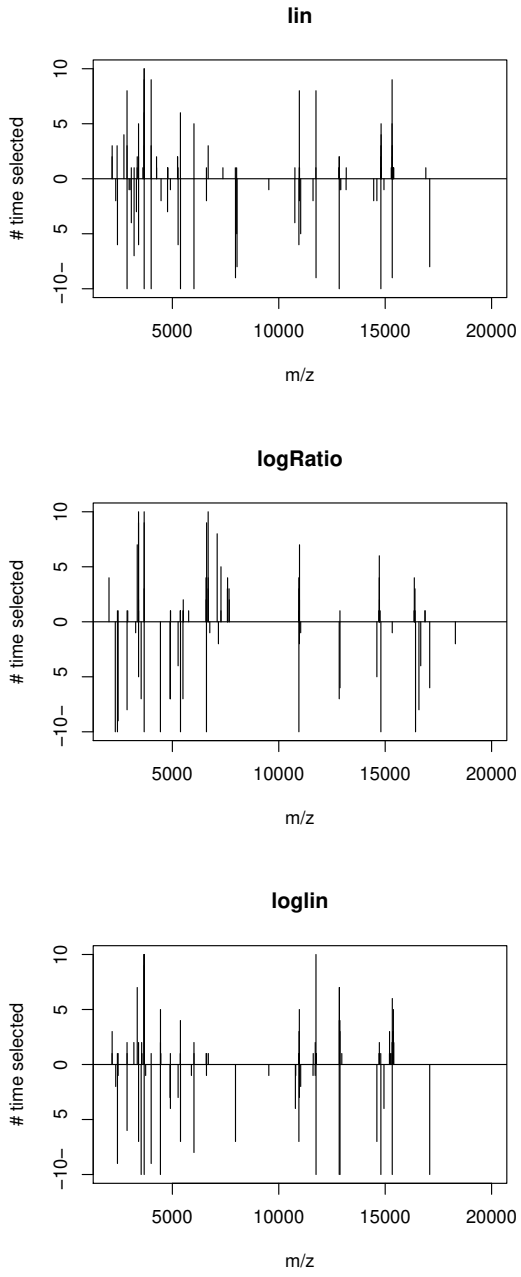


FIGURE 10.13 – Comparaison du nombre de fois ou un pic est sélectionné par SVMRFE sur les données brutes par rapport au nombre de fois ou il est sélectionné sur les données pré-traités. Il y a un tracé par noyau utilisé pour la sélection (linéaire, logRatio, loglin), et on limite la comparaison au $n = 20$ attributs les plus discriminants pour chaque méthode. L'axe des abscisse indique le m/z du pic concerné, la partie positive de l'axe des ordonnées indique le nombre de fois ou le pic est sélectionné dans les données brutes, la partie négative de l'axe des ordonnées indique le nombre de fois ou le pic est sélectionné dans les données pré-traités.

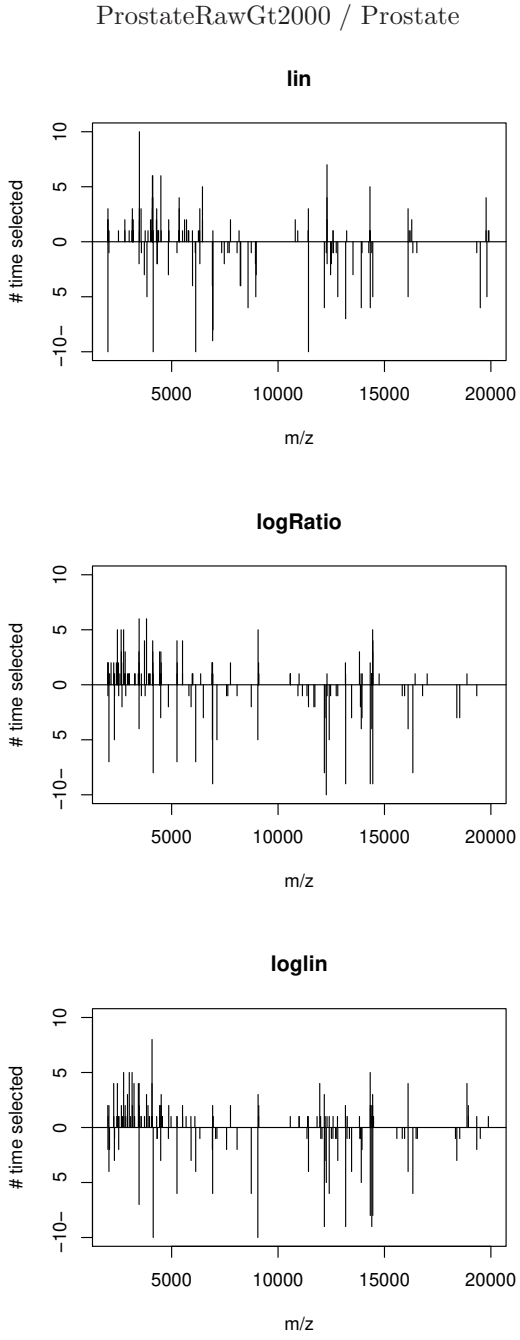


FIGURE 10.14 – Comparaison du nombre de fois ou un pic est sélectionné par SVMRFE sur les données brutes par rapport au nombre de fois ou il est sélectionné sur les données pré-traités. Il y a un tracé par noyau utilisé pour la sélection (linéaire, logRatio, loglin), et on limite la comparaison au $n = 20$ attributs les plus discriminants pour chaque méthode. L'axe des abscisse indique le m/z du pic concerné, la partie positive de l'axe des ordonnées indique le nombre de fois ou le pic est sélectionné dans les données brutes, la partie négative de l'axe des ordonnées indique le nombre de fois ou le pic est sélectionné dans les données pré-traités.

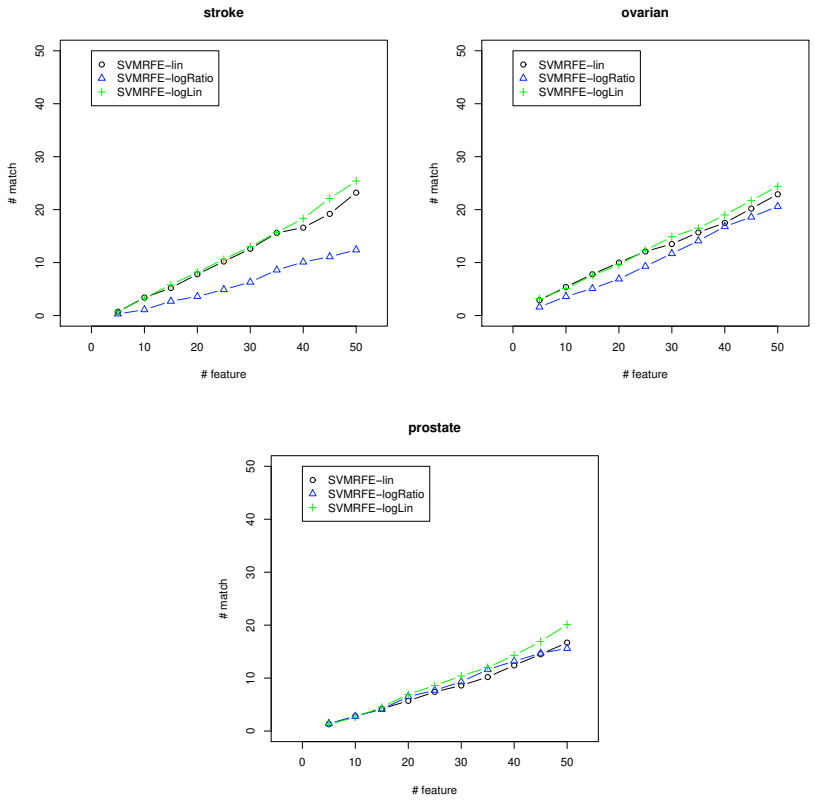


FIGURE 10.15 – Nombre moyen de fois où les m/z des n attributs sélectionnés sur les données pré-traitées coïncident avec les n attributs sélectionnés sur les données brutes (RawGt2000), avec une tolérance d'erreur de 1%.

du modèle SVM. Nous avons également pu définir grace à notre interprétation des modèles SVM-logRatio un algorithme de sélection d'attributs SVMRFE-logRatio.

Les expérimentations que nous avons menées avec le noyau logRatio montrent des résultats intéressants concernant les performances et l'interprétation des modèles qui en découle. Tout d'abord l'algorithme de sélection d'attributs logRatio-SVMRFE que nous avons défini se révèle exemplaire pour supprimer la redondance introduite artificiellement dans le jeu de donnée Iris, alors que linear-SVMRFE montre davantage de difficultés. Ensuite, sur les jeux de données réelles de grande dimension, les performances de classification du noyau logRatio sont aussi systématiquement égales ou supérieures à celles des autres noyaux testés (linéaire, et polynomial). Enfin, pour la sélection de plus de vingt attributs, logRatio-SVMRFE est légèrement plus stable que SVMRFE et choisit des attributs aussi pertinent que lui.

En dessous du seuil de vingt attributs, logRatio-SVMRFE voit ses performances de classification se dégrader jusqu'à passer en dessous de celles de ses concurrents. Ce résultat est cependant contredit par les expériences sur Iris car logRatio il a montré un excellent comportement alors que ces données ne contiennent que huit attributs. Néanmoins, cela montre une tendance de l'algorithme à privilégier des informations en relation avec les interactions entre les attributs au détriments des performances de classification lorsque l'on est en présence d'un petit nombre d'attributs. Signalons aussi que la réduction des données à vingt attributs est généralement suffisant pour de nombreuses applications, et notamment pour l'extraction des biomarqueurs. Vingt attributs est effectivement un nombre raisonnable car l'utilisateur final, expert du domaine, peut les explorer manuellement et y apporter les corrections qu'il juge nécessaire.

Concernant les données de spectrométrie de masse, logRatio-SVMRFE se montre également plus sensible que SVMRFE au pré-traitement des données. Cela ne remet pas en cause ses qualités dans ce domaine, mais nous enseigne qu'il est mieux d'opérer un bon pré-traitement des données MS (en suivant les recommandation données dans la première partie de ce document) si l'on souhaite employer cet algorithme. De plus, logRatio-SVMRFE est a priori une solution attractive dans le domaine de la spectrométrie de masse si l'on envisage d'analyser des spectres en provenance de conditions expérimentales différentes. En effet, cela nécessite de prêter une attention particulière au pré-traitement des données, et en particulier à la normalisation des spectres de masse, pour laquelle logRatio-SVMRFE montre une certaine robustesse de part le choix de la représentation des données qu'il utilise.

En conséquences, notre travail a montré qu'il y a plusieurs avantages à utiliser le noyau logRatio qui a été conçu spécifiquement dans l'optique de résoudre les problèmes auxquelles nous avons à faire face pour l'extraction des biomarqueurs dans les données d'expression. Il est cependant certainement possible d'améliorer notre travail à plusieurs niveau. Dans la conclusion (chapitre 11),

nous présenterons plusieurs pistes de recherche intéressante qui peuvent être explorer dans cette perspective.

Chapitre 11

Bilan de l'apprentissage

Dans cette partie, nous mettons en évidence les problèmes d'apprentissage que pose la sélection des attributs en vu de l'extraction des bio-marqueurs dans les données d'expression. On souligne en particulier l'importance de la normalisation des données, de la redondance d'information dans les données, et de la stabilité que doivent montrer les solutions apportés. Ces problèmes touchent en réalité de nombreux autres domaines qui génèrent une grande quantité d'information (par exemple textmining, finance, imagerie, astronomie, ...). La plupart des méthodes de sélection d'attributs, qui permettent de faire face à la grande dimension de ces données, ne prennent cependant pas en considération ces problèmes. Nous montrons en particulier que l'une des méthodes les plus efficaces pour la sélection d'attributs (SVMRFE [43]) peine à éliminer la redondance d'information (section 10.3), et est trés dépendante de la normalisation des données. Ce constat nous amène à proposer une extension de SVMRFE qui se concentre sur la résolution de ces maux. Notre solution est basée sur le noyau logRatio qui offre plusieurs avantages en relation avec le domaine de la biologie : il focalise l'apprentissage sur les interactions entre attributs ; il permet de mieux prendre en compte la redondance entre les attributs ; la représentation des données qu'il utilise est robuste à la normalisation ; les modèles SVM-logRatio sont représentable visuellement et riche en information. Il est en particulier possible d'identifier les attributs les plus pertinents des modèles SVM-logRatio et d'en déduire une méthode de sélection d'attributs SVMRFE-logRatio.

La limitation principale de notre approche est qu'elle est restreinte à l'analyse des valeurs strictement positives. Si des valeurs nulles sont présentes dans les données, nous avons cependant un moyen de les éliminer sans trop influencer le résultat, et au final, nous pouvons tout de même appliquer notre méthode pour traiter de nombreux jeux de donné. Nous allons également voir dans les perspectives qui suivent un dérivé du noyau logRatio (le noyau diff) qui n'a pas cette limitation.

11.1 Perspectives

Nous présentons dans cette sections plusieurs perspectives de recherche pour poursuivre notre travail sur le noyau logRatio.

11.1.1 Extension de logRatio-SVMRFE

Pour essayer d'améliorer les performances de logRatio-SVMRFE, nous pouvons commencer par considérer l'emploi du noyau logRatio avec les autres méthodes à noyau de la littérature. En particulier, il serait intéressant d'expérimenter la méthode de [99], qui cherche à minimiser la borne supérieure de l'erreur *leave-one-out* des SVM avec le noyau logRatio. En prenant cette voie, on peut espérer obtenir une amélioration des erreurs de classification car la sélection d'attribut est explicitement guidée par la minimisation de l'erreur de généralisation.

L'autre travail qui pourrait être étendu au noyau logRatio est celui de [132] qui modifie SVMRFE en proposant une alternative à l'interprétation des SVM-linéaire. Dans son interprétation des modèles, le score attribuée à un attribut dépend de sa capacité à éloigner les centroïdes des deux classes, de telle manière que l'objectif visée par la sélection d'attribut prend en considération cette distance. Nous pensons qu'il est certainement possible d'étendre cette idée pour l'interprétation des modèles SVM-logRatio.

11.1.2 Combinaison des méthodes de sélection d'attributs

Nous avons vu dans le chapitre 10.3 que logRatio-SVMRFE à des difficultés à sélectionner des sous-ensembles de moins de vingt attributs, mais il se comporte très bien pour des sous-ensembles d'attributs plus grand (performance similaire et stabilité supérieure aux autres approches testées). Pour les applications qui nécessitent de sélectionner moins de vingt attributs, il peut donc être intéressant d'envisager de combiner les méthodes, c'est à dire de commencer par sélectionner une vingtaine d'attributs avec logRatio-SVMRFE pour profiter de sa stabilité et de ses performances de classification, et poursuivre la réduction avec SVMRFE par exemple.

Cependant, il n'est pas certain que cette approche fonctionne car il est possible que les attributs sélectionnés par SVMRFE-logRatio lui soient très spécifiques et que, quand SVMRFE prend le relais, il ne parvienne pas à identifier les motifs que SVMRFE-logRatio a détecté dans les données. Ce problème est en réalité un problème plus générale qui se pose lorsque l'on cherche à combiner les résultats de différentes méthodes de sélection d'attributs pour essayer d'améliorer les performances de la sélection. L'idée de combiner les résultats de différents algorithmes de sélection d'attributs est pourtant tentante car elle fonctionne bien pour les algorithmes de classification : les méthodes de type bagging, stacking, boosting, arrivent à améliorer les performances en combinant les prédictions de plusieurs algorithmes de classification. Néanmoins, concernant les algorithmes de sélections d'attributs la combinaison des résultats est moins trivial à réaliser car les attributs sélectionnés ne sont pas indépendants comme

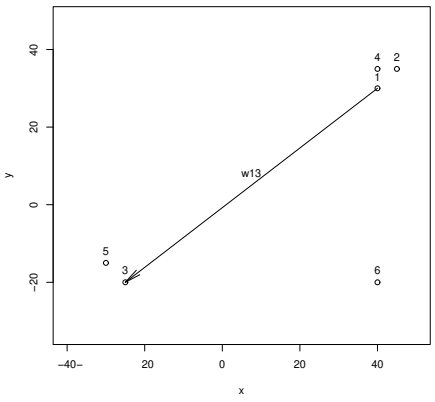


FIGURE 11.1 – Exemple de carte d'interaction entre attributs pour un modèle SVM : une visualisation bidimensionnelle que l'on aimerais construire à partir d'un modèle SVM. Les points du plan représente les n attributs de l'espace initial qui sont combinés deux à deux pour produire les n^2 caractéristiques de l'espace de caractéristiques. Et la distance entre deux points est égale à la valeur du coefficient linéaire du SVM pour la caractéristique associée aux deux attributs en question.

le sont les prédictions des modèles de classification sur des instances. En effet, les sous-ensembles d'attributs sélectionnés forment un tout, et se pose la question de la combinaison de plusieurs sous-ensembles. Pour la réaliser au mieux, il faut étudier les relations entre les différents attributs des sous-ensembles à combiner afin d'identifier des clusters d'attributs qui agissent en groupe, et leur appliquer un traitement cohérent. C'est une piste de recherche très intéressante pour le développement des méthodes de sélection d'attributs, dans laquelle nous pourrions exploiter notre travail sur la stabilité des algorithmes de sélection d'attributs. Il y a encore peu de travaux qui ont vu le jour dans cette direction [57, 66, 104, 72].

11.1.3 Effet de regroupement des attributs

Un autre point que nous avons négligé dans notre travail et qui serait intéressant d'étudier plus en profondeur, est l'effet de regroupement des attributs redondant dans les modèles SVM-logRatio. Nous avons effectivement vu que les attributs qui apportent une information redondante vis à vis de la classe ont tendance à avoir des poids proches dans les modèles SVM-logRatio, et inversement à être éloigné si la combinaison des deux est pertinente pour prédire la classe. En examinant les écarts de coefficients sur la visualisation 1D des modèles logRatio-SVM, il est possible d'extraire, dans une certaine mesure, des informations à la fois sur la pertinence des attributs et sur leur redondance. Malheureusement la visualisation des modèles logRatio-SVM est limité à une seule dimension, et il faudrait étudier la possibilité de les étendre à deux dimensions ou plus.

Supposons en effet que l'on ait un noyau comparable à logRatio, qui projette les instances dans un espace de caractéristiques contenant n^2 attributs obtenus en combinant deux à deux les n attributs de l'espace initial. Supposons également que l'on soit en mesure d'apprendre un modèle SVM avec ce noyau

et de le visualiser en deux dimension comme sur la figure 11.1. Les n points de la figure représentent chacun un attribut de l'espace initial, et la distance euclidienne entre deux points est égale à la valeur du coefficient linéaire calculé par le SVM et qui pondère la caractéristique qui combine les deux attributs de l'espace initial en question. Nous obtenons ainsi une représentation similaire à celle de logRatio mais beaucoup plus expressive car elle a deux degrés de liberté. Si nous construisons une telle représentation des modèles SVM, les attributs correspondants à une combinaison d'attribut pertinente seraient éloignés les uns des autres, et les attributs redondants seraient situés proches les uns des autres (comme dans logRatio mais avec deux degrés de liberté).

Imaginons un instant que ces modèles soit utilisés en bioinformatique sur des données d'expression de gène/protéine, comme les jeux de données Breast, Stroke, Ovarian, Prostate que nous avons utilisés dans ce chapitre. La visualisation des données de la figure 11.1 offrirait des informations simples et précieuses au biologiste qui analyse ces données car il pourrait y identifier, en un coup d'oeil, les interactions entre les gènes/protéines qui sont importantes pour le modèle SVM et en même temps y déceler les redondances entre eux. Cette figure est en quelque sorte une carte d'interaction des attributs dans les modèles SVM.

Les utilisateurs expriment en réalité ici un besoin grandissant pour mieux comprendre et interpréter les modèles de classification qui sont produits en apprentissage automatique. C'est d'ailleurs l'une des raison qui rend les arbres de décision si populaire. Avec nos modèles SVM-logRatio, nous sommes en mesure d'offrir des informations un peu plus riche (sur la redondance) que celles offertes par les modèles SVM-linéaire. Malheureusement, nous sommes limité à une seul dimension, et notre problème pour passer à deux dimensions (voir plus) est de trouver le noyau qui permet d'interpréter les modèles SVM de manière adéquate pour produire la représentation bidimensionnelle.

11.1.4 Noyau *diff*

Enfin, il y a un noyau très simple, étroitement liée avec logRatio et au noyau linéaire, qu'il est intéressant de mentionner pour des expérimentations futures (même si nous pouvons déjà tirer des conclusions sur ce noyau à partir de nos résultats sur de logRatio). Il s'agit du noyau *diff* que nous appelons ainsi pour sa connotation avec le terme "différence". Ce noyau permet de projeter les instances de l'espace initial dans un espace de n^2 caractéristiques contenant simplement l'ensemble des différences entre les valeurs des attributs de l'espace initial (plutôt que le logarithme de leur ratio). La fonction de projection ϕ associée au noyau *diff* est donc :

$$\phi(\mathbf{x}) = (x_i - x_j)_{(i,j)=(1,1)}^{(n,n)} \tag{11.1}$$

La projection *diff* présente des similitudes très fortes avec la projection de logRatio car nous pouvons voir qu'il est équivalent d'utiliser le noyau logRatio ou de transformer les valeurs en leur logarithme puis d'y appliquer le noyau

diff. Aussi, la fonction noyau associé à *diff* est égale, à une constante prêt, à la covariance des instances ce qui le rend très proche du noyau linéaire :

$$K_{\text{diff}}(\mathbf{x}, \mathbf{z}) = \sum_{i,j} (x_i - x_j)(z_i - z_j) \tag{11.2}$$

$$= 2n(n-1)\,\text{cov}(\mathbf{x}, \mathbf{y}) \tag{11.3}$$

Les similitudes de *diff* avec logRatio, nous permettent de suivre un cheminement comparable à celui que nous avons suivi dans le chapitre 10 pour nous apercevoir : 1) qu'il est possible de visualiser l'espace des caractéristiques du noyau *diff* de la même manière que celui de logRatio simplement en affichant les coefficients ; 2) qu'il est possible d'opérer la sélection des attributs de la même manière qu'avec logRatio (en fonction de la distance à la médiane). En plus de cela, le noyau *diff* travail directement avec les données de l'espace initial comme le noyau linéaire et il n'est pas limité au seul valeurs positives comme logRatio. En revanche, *diff* est sensible à la normalisation des données.

Une chose intéressante qu'apporte le noyau *diff*, c'est qu'il offre une alternative à l'interprétation classique des modèle SVM-linéaire dans le cas ou tous les attributs s'expriment dans une unité identique. Effectivement, comme on peut le constater, le noyau *diff* est équivalent un calcul de covariance entre deux instances. La covariance est un simple produit scalaire entre des instances qui ont subi une translation égale à la moyenne de leurs valeurs (c'est à cause du calcul de la moyenne des valeurs de l'instance qu'il est nécessaire que tous les attributs aient la même unité). En conséquence, interpréter les coefficients d'un modèle SVM-*diff* revient à interpréter les coefficients d'un modèle SVM-linéaires. Cette fois, par contre, les coefficients ne sont pas interprétés indépendamment les uns des autres comme c'est le cas dans l'interprétation classique des SVM-linéaire, mais par pair comme dans logRatio. Cela laisse apparaître les contraintes qui peuvent exister entre les coefficients d'un SVM-linéaire (se reporter à la discussion de la section 10.2 au sujet de logRatio), et pourrais permettre de mieux comprendre le fonctionnement des modèles SVM-linéaire.

Troisième partie

Conclusion

Chapitre 12

Conclusion

Nous proposons dans cette thèse un pipeline complet pour l'extraction de bio-marqueurs potentiels à partir des données de spectrométrie de masse MALDI-TOF et SELDI-TOF. Notre approche met l'accent sur la résolution de plusieurs points problématiques rencontrés dans ce domaine : la reproductibilité des technologies ; le pré-traitement et la structuration de l'information ; la normalisation des données ; la forte dimension des données ; la redondance d'information ; la stabilité des résultats ; le paramétrage des algorithmes ; l'interprétation des modèles ; l'interaction entre les variables.

On distingue deux phases dans ce pipeline de de traitement des données. La première phase est un pré-traitement des données dans laquelle des opérations non supervisées sont réalisées afin de structurer l'information contenue dans les spectres de masse. La deuxième phase est un apprentissage supervisé qui permet d'extraire de la matrice d'expression des bio-marqueurs potentielles. Les deux phases se rejoignent néanmoins sur plusieurs concepts. Tout d'abord, les deux phases visent à réduire la quantité d'information à analyser tout en minimisant la perte d'information. Pour le pré-traitement cela se traduit par une compression de l'information contenue dans les spectres de masse aux seuls informations contenues dans les pics. Pour l'apprentissage, il s'agit de sélectionner l'information sans éliminer les éléments qui permettent de distinguer les différents groupes de spectres de masse identifiés. Remarquez que la détection des pics peut être vu comme un algorithme de sélection d'attribut qui sélectionne la position des pics dans les données brutes des spectres de masse ce qui explique certains points parallèles entre les deux. Le deuxième point commun, c'est l'importance de la stabilité dans les deux phases du processus pour garantir la reproductibilité des expériences et garantir une certaine qualité des résultats. Nous avons exploité ce critère à trois reprises, d'abord pour analyser la qualité des protocoles expérimentaux (chapitre 6), ensuite pour comparer les algorithmes de pré-traitement (chapitre 7), enfin comme critère de qualité des algorithmes de sélection d'attribut (chapitre 9). Les deux phases ont également en commun le thème de la normalisation des données. La phase de pré-traitement nécessite une normalisation des spectres de masse pour générer une matrice d'expression

contenant des valeurs comparables entre elles. Dans l'apprentissage, la normalisation des données est en réalité un problème récurent.

Dans la suite, nous reprenons nos principales contributions dans ces deux phases de pré-traitement et d'apprentissage, puis mentionnons quelques perspectives de travail.

12.1 Pré-traitement

Au niveau du pré-traitement, la réduction de la quantité d'information est essentiellement guidée par les connaissances du domaine qui nous informent que l'information pertinente est concentrée dans les pics des spectres. Un soin particulier est donc pris dans le chapitre 4 pour définir un algorithme de détection de pics qui soit simple, rapide (complexité linéaire), universelle (on peut l'étendre à d'autres types de données que les spectres de masse), et facile à paramétrer (pas de paramètre fixé a priori, et seulement un paramètre pour contrôler le rapport signal/bruit à posteriori, une fois le résultat de l'algorithme connu). Nous montrons, section 7.3, que cet algorithme a un comportement plus approprié au traitement des données de spectrométrie de masse que les approches proposées jusqu'alors. Effectivement, dans la totalité des conditions expérimentales testés sa stabilité est plus élevée que celle obtenue avec un algorithme de type maximum local (PROcess) couramment employé. Nous pensons que cet algorithme de détection des pics est une contribution importante de notre travail de part sa qualité. De plus, sa complexité linéaire en fait un algorithme intéressant pour l'avenir pour traiter les volumes de données de plus en plus important en spectrométrie de masse. Signalons néanmoins que son utilisation est limité aux applications qui ne requièrent pas la distinction des pics qui se chevauchent dans les spectres. Aussi, il n'intègre pas le calcul de la charge des pics et le "déisotpoing" qui doivent être réalisé a postériori si on souhaite exploiter cet algorithme sur des spectres dont la résolution laisse apparaître la distribution isotopique des pics.

La deuxième contribution intéressante de notre travail est l'algorithme d'alignement des pics (chapitre 5) qui exploite une méthode de clustering hiérarchique. Lui aussi à une complexité faible, et il est bien adaptée à notre problème d'alignement car nos spectres de masse ne souffrent pas d'un défaut important de calibration. Néanmoins, la généralisation de cette algorithmes à d'autres application est assez limitée. En particulier, il est difficile de le généraliser pour l'alignement d'expérience LCMS, qui pose un problème d'alignement intéressant.

Enfin, l'ensemble de notre travail sur le pré-traitement est mis en valeur par notre travail sur la reproductibilité des protocoles MALDI-TOF (chapitre 6) et par l'effort que nous avons fait pour montrer la qualité de notre approche (chapitre 7). Ces deux travaux permettent de fournir un ensemble recommandations pour mener au mieux une expériences de spectrométrie de masse. Tout d'abord, au niveau des protocoles de préparation des échantillons, nous avons vu des différences de reproductibilité très importante entre les différents protocoles, et il a été possible de déterminer le protocoles le plus adapté. Ensuite, au niveau

du pré-traitement, nous avons pu déterminer le paramétrage et la représentation des données la plus adaptée à la discrimination des données (section 7.2).

12.2 Apprentissage automatique

La réduction de la quantité d'information durant le pré-traitement est considérable, mais elle reste limitée au nombre de pics dans les spectres masse, et plus généralement a des aspects non supervisées. Or pour aller plus loin et extraire des biomarqueurs potentiels, il faut également exploiter l'information sur la classe des patients associés aux spectres de masse (sain/malade). C'est le rôle de l'apprentissage automatique et de sélection d'attributs que nous abordons dans la seconde partie du document. Dans notre approche de l'apprentissage, nous avons mis l'accent sur des concepts importants de la biologie : interprétation des modèles, aspects liées à l'interaction entre les attributs (redondance d'information), normalisation des données, et la stabilité des résultats.

La stabilité s'est en particulier montré très importante pour évaluer la qualité des algorithmes de sélection d'attributs. Le travail présenté dans le chapitre 9 à ce sujet est une contribution importante dans ce domaine qui ouvre la voie à de nombreuse perspectives recherches. On a pu en particulier constater avec ce travail une instabilité assez importante des méthodes à base de machines à vecteurs supports et certaines raisons qui peuvent expliquer ces instabilités recoupent les différents aspects énuméré précédemment (normalisation des données, mauvaise interprétation des modèles, redondances entre les attributs). L'étude du noyau logRatio qui en a suivi essaye d'aborder ces problématiques.

Effectivement, malgré leur instabilité, les algorithmes à base de machines à vecteur supports sont apparus comme un point de départ idéal pour résoudre ces problèmes, notamment face à la grande dimension des données. Nous avons cherché à résoudre les lacunes des approches antérieurs mentionnées ci-dessus à l'aide du noyau logRatio (chapitre 10). Pour cela, notre méthode d'apprentissage concentre son attention sur les interactions attributs-attributs ce qui lui permet de traiter plus efficacement la redondance d'information. De plus, elle simplifie le pré-traitement des données car elle est insensible à la normalisation des données ce qui évite d'avoir à réaliser cette opération parfois hasardeuse. Une autre originalité de notre méthode réside dans l'interprétation des modèles de classification produits, ceux-ci sont faciles à déchiffrer et permettent d'assigner un score à chaque attribut en fonction de son importance sur les prédictions du modèle. Ce score sert par la suite à sélectionner les attributs qui sont potentiellement des biomarqueurs, et en poussant loin leur interprétation, ils fournissent même des informations sur les interactions attribut-attribut. Hélas la représentation unidimensionnelle employée limite le pouvoir expressif de ces modèles.

Les expériences menées avec SVMRFE-logRatio pour la sélection d'attributs montrent un comportement très intéressant de la méthode par rapport à celle sur laquelle il est fondé (SVMRFE-linéaire). En particulier, sur des données où de la redondance est introduite artificiellement, SVMRFE-logRatio a un comportement exemplaire puisqu'il extirpe les attributs attendus, alors que SVMRFE-

linéaire faillit à cette tâche (section 10.3). Sur des jeux de données réelles, le noyau logRatio s'est également révélé au moins aussi puissant que les autres noyaux testés pour identifier des motifs de classification. La sélection d'attributs sur ces mêmes données est également très intéressante lorsqu'il s'agit de choisir plus d'une vingtaine d'attributs. La seconde contribution de ce travail est donc cette méthode à base de noyau logRatio et de SVM pour le diagnostique et la sélection des attributs. Ses propriétés, son comportement et le retour d'information qu'elle offre se sont effectivement montrés très intéressants et prometteurs.

12.3 Perspectives

Dans ce travail nous nous sommes essentiellement limité au domaine de la spectrométrie SELDI-TOF à basse résolution, or de part la simplicité et l'universalité des méthodes proposées, on peut songer à les étendre à d'autres domaines. C'est tout d'abord la spectrométrie de masse à haute résolution qui vient à l'esprit. Dans ce domaine les propriétés non destructives de notre algorithme de détection de pics, sa rapidité, et son extension naturelle aux données multidimensionnels peuvent effectivement être très attractive. D'ailleurs nous avons montré que nous pouvions appliquer l'algorithme pour le traitement de données LCMS, le but étant a terme de pouvoir opérer un pré-traitement des données en temps réelles. On peut aussi envisager d'appliquer certaines de nos méthodes de pré-traitement à des domaines connexes à la spectrométrie de masse comme par exemple le traitement d'images de puces à ADN ou de gel d'électrophorèse. Là encore l'algorithme de détection de pics pourrait être avantageux pour détecter les spots sur les images.

Par contre si on se replace dans le cadre général de l'apprentissage des signaux, les pré-traitements proposées n'ont qu'une faible portée. Ils semblent par exemple peu appropriés au traitement de données financières ou de sismologies dans lesquelles l'information est plus difficile à représenter et à structurer. Effectivement, dans ces données, ce n'est pas forcément les pics qui sont importants, et les motifs qui y apparaissent sont moins facile à identifier. Par exemple les pics que l'on peu identifier dans des données de cardiologie ne sont pas située à une position identique dans tous les signaux, comme c'est le cas dans les spectres de masse. Nous sommes donc encore loin d'une méthode universelle pour l'apprentissage à partir de signaux en général. Toutefois, certains aspects que nous avons étudier dans notre travail vont dans cette direction. Effectivement une des caractéristique commune à tous les signaux c'est la redondance d'information, et celle-ci est étroitement liée à l'information de leur voisinage dans les données. Effectivement, dans une image, par exemple, des pixels situées l'un à coté de l'autre ont de bonnes chances d'avoir des couleurs identiques. Or dans notre travail, nous avons concentré nos efforts d'apprentissage sur la redondance d'information, et nous avons employé un graphe de voisinage pour la détection des pics. Ces deux notions sont très universelles, et certainement est-il intéressant à l'avenir de se pencher sur des algorithmes d'apprentissage qui

puissent traiter des instances représentées sous la forme de graphes de voisinage plutôt que d'un vecteur de valeurs. Notez au passage que cela suppose que le pré-traitement est intrinsèque à l'algorithme d'apprentissage, or les efforts que nous avons faits avec logRatio pour intégrer la normalisation dans le processus d'apprentissage vont dans ce sens de l'intégration du pré-traitement dans le processus d'apprentissage.

A coté de ces perspectives alléchantes sur la généralisation des algorithmes d'apprentissage pour le traitement des signaux, rappelons également que d'autres perspectives de travail sur l'évolution du noyau logRatio ont déjà été évoqués dans le chapitre 11.

Bibliographie

[1] B.-L. Adam, Y. Qu, J. W. Davis, M. D. Ward, M. A. Clements, L. H. Cazares, O. J. Semmes, P. F. Schellhammer, Y. Yasui, Z. Feng, and G. L. Wright. Serum protein fingerprinting coupled with a pattern-matching algorithm distinguishes prostate cancer from benign prostate hyperplasia and healthy men. *Cancer Research*, 62 :3609–3614, 2002.

[2] R. Aebersold and M. M. Mass spectrometry-based proteomics. *Nature*, 422 :198–207, 2003.

[3] V. Andreev, T. Rejtar, H.-S. Chen, E. Moskovets, A. Ivanov, and B. Karger. A universal denoising and peak picking algorithm for lc-ms based an matched filtration in the chromatographic time domain. *Acta Haematologica*, 94(1) :10–15, 1995.

[4] K. Baggerly, J. S. Morris, J. Wang, D. Gold, X. Lian-Chun, and K. R. Coombes. A comprehensive approach to the analysis of matrix-assisted laser desorption/ionization-time of flight proteomics spectra from serum samples. *Proteomics*, 3 :1667–1672, 2003.

[5] K. A. Baggerly, J. S. Morris, and K. R. Coombes. Reproducibility of SELDI-TOF protein patterns in serum : comparing data sets from different experiments. *Bioinformatics*, 20 :777–785, 2004.

[6] V. Barclay, R. Bonner, and I. Hamilton. Application of wavelet transforms to experimental spectra : Smoothing, denoising, and data set compression. *Anal. Chem.*, 69 :78–90, 1997.

[7] M. Bellew, M. Coram, M. Fitzgibbon, M. Igra, T. Randolph, P. Wang, D. May, J. Eng, R. Fang, C. Lin, J. Chen, D. Goodlett, J. Whiteaker, A. Paulovich, and M. McIntosh. A suite of algorithms for the comprehensive analysis of complex protein mixtures using high-resolution lc-ms. *Bioinformatics*, 22(15) :1902–1909, Aug 2006.

[8] H. P. Benton, D. M. Wong, S. A. Trauger, and G. Siuzdak. Xcms2 : processing tandem mass spectrometry data for metabolite identification and structural characterization. *Anal Chem*, 80(16) :6382–6389, 2008 Aug 15.

[9] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3 :1229–1243, 2003.

[10] P.-A. Binz, D. Hochstrasser, and R. Appel. Mass spectrometry-based proteomics : Current status and potential use in clinical chemistry. *Clin Chem Lab Med*, 41(12), 2003.

[11] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[12] E. J. Breen, F. G. Hopwood, K. L. Williams, and M. R. Wilkins. Automatic poisson peak harvesting for high throughput protein identification. *Electrophoresis*, 21 :2243–2251, 2000.

[13] C.-C. Chang and C.-J. Lin. *LIBSVM : a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[14] O. Chapelle and V. Vapnik. Choosing multiple parameters for support vector machines. *AT&T Labs Technical Report*, 2000.

[15] E. Check. Running before we can walk ? *Nature*, 429 :496–497, 2004.

[16] P.-H. Chen, C.-J. Lin, and B. Scholkopf. A tutorial on nu-support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, 2002.

[17] W. Clarke, B. Silverman, Z. Zhang, D. Chan, A. Klein, and E. Molmenti. Characterization of renal allograft rejection by urinary proteomic analysis. *Annals of Surgery*, 237(5) :660–665, 2003.

[18] T. P. Conrads, V. A. Fusaro, S. Ross, D. Johann, V. Rajapakse, B. A. Hitt, S. M. Steinberg, E. C. Kohn, D. A. Fishman, G. Whitely, J. C. Barrett, L. A. Liotta, E. F. r. Petricoin, and T. D. Veenstra. High-resolution serum proteomic features for ovarian cancer detection. *Endocr Relat Cancer*, 11(2) :163–178, 2004 Jun.

[19] K. Coombes, J. Koomen, K. Baggerly, J. Morris, and R. Kobayashi. Quality control and peak finding for proteomics data collected from nipple aspirate fluid by surface-enhanced laser desorption and ionization. *Clin Chem.*, 49(10) :1615–1623, 2003.

[20] K. Coombes, J. Koomen, K. Baggerly, J. Morris, and K. Ryuji. Understanding the characteristics of mass spectrometry data through the use of simulation. *Cancer Informatics*, 1(1), 2005.

[21] K. Coombes, S. Tsavachidis, J. Morris, K. Baggerly, H. M.C., and K. H.M. Improved peak detection and quantification of mass spectrometry data acquired from surface-enhanced laser desorption and ionization by denoising spectra with the undecimated discrete wavelet transform. *Proteomics*, 2005.

[22] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2002.

[23] M. A. Davenport. The 2nu-svm : A cost-sensitive extension of the nu-svm. Technical report, Department of Electrical and Computer Engineering, Rice University, 2005.

[24] M. A. Davenport, R. G. Baraniuk, and C. D. Scott. Controlling false alarms with support vector machines. In *ICASSP*, pages 589–592, 2006.

[25] C. Davis, F. Gerick, V. Hintermair, C. Friedel, K. Fundel, R. Kuffner, and R. Zimmer. Reliable gene signatures for microarray classification : assessment of stability and performance. *Bioinformatics*, 22(19) :2356–2363, 2006.

[26] E. Diamandis. Mass spectrometry as a diagnostic and a cancer biomarker discovery tool. *Molecular and Cellular Proteomics*, 3 :397–378, 2004.

[27] T. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7) :1895–1923, 1998.

[28] Y. Ding and W. Dawn. Improving the performance of svm-rfe to select genes in microarray data. *BMC Bioinformatics*, 7, 2006.

[29] P. Du, W. A. Kibbe, and S. M. Lin. Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching. *Bioinformatics*, 22(17) :2059–2065, 2006.

[30] R. Duda, P. Hart, and D. Stork. *Pattern Classification and Scene Analysis*. John Willey and Sons, 2001.

[31] V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *ICML08*, 2008.

[32] E. Fung and C. Enderwick. Proteinchip clinical proteomics : Computational challenges and solutions. *Computational Proteomics Supplement*, 32 :S34–S41, 2002.
A small insight to the software of Ciphergen.

[33] C. Furlanello, M. Serafini, S. Merler, and G. Jurman. Entropy-based gene ranking without selection bias for the predictive classification of microarray data. *BMC Bioinformatics*, 6, 2003.

[34] D. Geman, C. d'Avigon, D. Q. Naiman, and R. L. Winslow. Classifying gene expression profiles from pairwise mrna comparison. *Statistical Applications in Genetics and Molecular Biology*, 3(1), 2004.

[35] R. Gentleman, V. Carey, W. Hubert, R. Irizarry, and S. Dudoit. *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Springer, 2005.

[36] M. Gentzel, T. Kocher, S. Ponnusamy, and M. Wilm. Preprocessing of tandem mass spectrometric data to support automatic protein identification. *Proteomics*, 3(8) :1597–1610, 2003 Aug.

[37] A. B. A. Graf and S. Borer. Normalization in support vector machines. In *in Proc. DAGM 2001 Pattern Recognition*, pages 277–282. SpringerVerlag, 2001.

[38] A. B. A. Graf, A. J. Smola, and S. Borer. Classification in a normalized feature space using support vector machines, 2003.

[39] R. Gras, M. Muller, E. Gasteiger, S. Gay, P. Binz, W. Bienvenut, C. Hoogland, J. Sanchez, A. Bairoch, D. Hochstrasser, and R. Appel. Improving protein identification from peptide mass fingerprinting through a parameterized multi-level scoring algorithm and an optimized peak detection. *Electrophoresis*, 20 :3535–50, Dec 1999.

[40] W. Grizzle, J. Semmes, J. Basler, E. Izbicka, Z. Feng, J. Kagan, B.-L. Adam, D. Troyer, S. Srivastava, M. Thornquist, Z. Zhang, and I. Thompson. The early detection research network surface-enhanced laser desorption and ionization prostate cancer detection study : a study in biomarker validation in genitourinary oncology. *Urologic Oncology*, 22 :337–343, 2004.

[41] Guibas and Stolfi. Primitives for manipulation of general subdivisions and the computation of voronoi diagrams. *ACM Transactions on Graphics*, 4(2) :74–123, 1985.

[42] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3, 2003.

[43] I. Guyon, J. Weston, and S. Barnhill. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46, 2002.

[44] J. Hartler, G. G. Thallinger, G. Stocker, A. Sturn, T. R. Burkard, E. Korner, R. Rader, A. Schmidt, K. Mechtler, and Z. Trajanoski. Maspectras : a platform for management and analysis of proteomics lc-ms/ms data. *BMC Bioinformatics*, 8 :197, 2007.

[45] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[46] R. C. Herrick and J. S. Morris. Wavelet-based functional mixed model analysis : Computation consideratons. *ASA Section on Statistical Computing*, 2006.

[47] M. Hilario and A. Kalousis. Approaches to dimensionality reduction in proteomic biomarker studies. *Briefings in Bioinformatics*, 9(2), 2008.

[48] M. Hilario, A. Kalousis, C. Pellegrini, and M. Müller. Processing and classification of protein mass spectra. *Mass Spectrometry Reviews*, 25 :409–449, 2006.

[49] M. Hilario, A. Kalousis, J. Prados, and P. A. Binz. Data mining for mass spectra-based cancer diagnosis and biomarker discovery. *Drug Discovery Today : Biosilico*, 2(5) :214–222, September 2004.

[50] H. Hong, Y. Dragan, J. Epstein, C. Teitel, B. Chen, Q. Xie, H. Fang, L. Shi, R. Perkins, and W. Tong. Quality control and quality assessment of data from surface-enhanced laser desorption/ionization (seldi) time-of flight (tof) mass spectrometry (ms). *BMC Bioinformatics*, 6, 2004.

[51] G. Hortin. Can mass spectrometric protein profiling meet desired standards of clinical laboratory practice ? *Clinical Chemistry*, 51(1), 2005.

[52] J. Hu, K. Coombes, J. Morris, and K. Baggerly. The importance of experimental design in proteomic mass spectrometry experiments : Some cautionary tales. *Briefings in functional genomics and proteomics*, 3(4) :322–331, 2005.

[53] N. Jeffries. Alogrithms for alignment of mass spectrometry proteomic data. *Bioinformatics*, 21(14) :3066–3073, May 2005.

[54] L. Jia and Z. Hongyuan. Simultaneous classification and feature clustering using discriminant vector quantization with applications to microarray data analysis. *Proceedings of the IEEE Computer Society Bioinformatics Conference (CSB02)*, 2002.

[55] T. Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

[56] K. Jong, E. Marchiori, M. Sebag, and A. Van der Vaart. Feature selection in proteomic pattern data with support vector machines. In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 41–48, 2004.

[57] K. Jong, J. Mary, A. Cornuéjols, E. Marchiori, and M. Sebag. Ensemble feature ranking. In *Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, Lecture Notes In Computer Science ; Vol. 3202, pages 267–278, 2004.

[58] A. Kalousis, J. Prados, and M. Hilario. Stability of feature selection algorithms. In *5th IEEE International Conference on Data Mining*, 2005.

[59] A. Kalousis, J. Prados, and M. Hilario. Stability of feature selection algorithms : a study on high-dimensional spaces. *Knowledge and Information Systems*, 2006.

[60] A. Kalousis, J. Prados, E. Rexhepaj, and M. Hilario. Feature extraction from mass spectra for classification. In *6th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2005.

[61] A. Kalousis, J. Prados, J. C. Sanchez, L. Allard, and M. Hilario. Distilling classification models from cross-validation runs : an application to mass spectrometry. In *International Conference on Tools with Artificial Intelligence (ICTAI-05)*, Boca Raton, FL, November 2004.

[62] M. Katajamaa and M. Oresic. Processing methods for differential analysis of lc/ms profile data. *BMC Bioinformatics*, 6 :179, 2005.

[63] M. Kempa, J. Sjodahl, and et al. Improved method for peak picking in matrix-assisted laser desorption/ionisation time-of-flight mass spectrometry. *Rapid Commun Mass Spectrom*, 18(11) :1208–1212, 2004.

[64] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 1-2, 1997.

[65] D. Koller and S. Mehran. Toward optimal feature selection. *13th International Conference on Machine Learning*, 1996.

[66] L. Kuncheva. A stability index for feature selection. In *Proceedings of the 25th IASTED International Multi-Conference : artificial intelligence and applications*, 2007.

[67] E. Lange, C. Gropl, K. Reinert, O. Kohlbacher, and A. Hildebrandt. High-accuracy peak picking of proteomics data using wavelet techniques. *Pac Symp Biocomput*, pages 243–254, 2006.

[68] E. Lange, C. Gropl, O. Schulz-Trieglaff, A. Leinenbach, C. Huber, and K. Reinert. A geometric approach for the alignment of liquid chromatography-mass spectrometry data. *Bioinformatics*, 23(13) :i273–81, 2007 Jul 1.

[69] X.-j. Li, E. C. Yi, C. J. Kemp, H. Zhang, and R. Aebersold. A software suite for the generation and comparison of peptide arrays from sets of data collected by liquid chromatography-mass spectrometry. *Mol Cell Proteomics*, 4(9) :1328–1340, 2005 Sep.

[70] X.-J. Li, H. Zhang, J. A. Ranish, and R. Aebersold. Automated statistical analysis of protein abundance ratios from data generated by stable-isotope dilution and tandem mass spectrometry. *Anal Chem*, 75(23) :6648–6657, 2003 Dec 1.

[71] S. M. Lin, P. Du, and W. A. Kibbe. Qa/qc of clinical seldi-tof data with wavelets. *CAMDA06*, 2006.

[72] S. Loscalzo, L. Yu, and C. Ding. Consensus group based stable feature selection. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD-2009*, 2009.

[73] A. Magnani and S. Boyd. Convex piecewise-linear fitting. *Submitted to Optimization and Engineering*, 2006.

[74] D. Malyarenko, W. Cooke, B. Adam, G. Malik, H. Chen, E. Tracy, M. Trosset, O. Semmes, and D. Manos. Enhancement of sensitivity and resolution of surface-enhanced laser desorption/ionization time-of-flight mass spectrometric records for serum peptides using time-series analysis techniques. *Clinical Chemistry*, 51(1) :65–74, 2005.

[75] E. Marchiori, C. Jimenez, M. West-Nielsen, and N. Heegaard. Robust svm-based biomarker selection with noisy mass spectrometric proteomic data. In *Applications of Evolutionary Computing*, pages 79–90, 2006.

[76] A. Mitchel, A. Divoli, J. Kim, M. Hilario, I. Selimas, and T. Attwood. Metis : multiple extraction techniques for informative sentences. *Bioinformatics*, 21, 2005.

[77] A. Mohammad-Djafari, J. Giovannelli, G. Demoment, and J. Idier. Regularization, maximum entropy and probabilistic methods in mass spectrometry data processing problems. *International Journal of Mass Spectrometry*, 215 :175–193, 2002.

[78] M. E. Monroe, N. Tolic, N. Jaitly, J. L. Shaw, J. N. Adkins, and R. D. Smith. Viper : an advanced software package to support high-throughput

lc-ms peptide identification. *Bioinformatics*, 23(15) :2021–2023, 2007 Aug 1.

[79] J. Morris, K. Coombes, J. Koomen, K. Baggerly, and R. Kobayashi. Feature extraction and quantification for mass spectrometry in biomedical applications using the mean spectrum. *Bioinformatics*, 21(9) :1764–1775, 2005.

[80] J. S. Morris, P. J. Brown, R. C. Herrick, K. A. Baggerly, and K. R. Coombes. Bayesian analysis of mass spectrometry proteomic data using wavelet-based functional mixed models. *Biometrics*, 64(2) :479–489, 2008 Jun.

[81] L. N. Mueller, O. Rinner, A. Schmidt, S. Letarte, B. Bodenmiller, M.-Y. Brusniak, O. Vitek, R. Aebersold, and M. Muller. Superhirn - a novel tool for high resolution lc-ms-based peptide/protein profiling. *Proteomics*, 7(19) :3470–3480, 2007 Oct.

[82] M. Muller. *Molecular Scanner Data Analysis*. PhD thesis, University of Geneva, 2003.

[83] D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases, 1998.

[84] K. Noy and D. Fasulo. Improved model-based, platform-independent feature extraction for mass spectrometry. *Bioinformatics*, 23(19) :2528–2535, 2007 Oct 1.

[85] R. Pelikan, B. W.L., D. Malehorn, J. Lyons-Weiler, and M. Hauskrecht. Intersession reproducibility of mass spectrometry profiles and its effect on accuracy of multivariate classification models. *Bioinformatics*, 2007.

[86] E. Petricoin, A. Ardekani, B. Hitt, P. Levine, V. Fusaro, S. Steinberg, G. Mills, C. Simone, D. Fishman, E. Kohn, and L. Liotta. Use of proteomic patterns in serum to identify ovarian cancer. *The Lancet*, 359 :572–577, Feb 2002.

[87] E. Petricoin, D. Ornstein, C. Paweletz, A. Ardekani, P. Hackett, B. Hitt, A. Velassco, C. Trucco, L. Wiegand, K. Wood, C. Simone, P. Levine, W. Linehan, M. Emmert-Buck, S. Steinberg, E. Kohn, and L. Liotta. Serum proteomic patterns for detection of prostate cancer. *Journal of the NCI*, 94(20), 2002.

[88] J. Pittman and C. Murthy. Fitting optimal piecewise linear functions using genetic algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7) :701–718, 2000.

[89] J. Prados, A. Kalousis, L. Allard, O. Carrette, J. C. Sanchez, and M. Hilario. Mining mass-spectra for diagnosis and biomarker discovery of cerebral accidents. *Proteomics*, 4 :2320–2332, 2004.

[90] J. Prados, A. Kalousis, and M. Hilario. On preprocessing of seldi-ms data and its evaluation. In *9th IEEE International Symposium on Computer-Based Medical Systems*, Salt Lake City, Utah, June 2006.

[91] J. Prados, A. Kalousis, and M. Hilario. Feature selection with the logratio kernel. *SIAM*, 2008.

[92] J. Prados, A. Kalousis, and M. Hilario. Logratio, un noyau pour la sélection d'attributs. *RFIA*, 2008.

[93] A. Prakash, P. Mallick, J. Whiteaker, H. Zhang, A. Paulovich, M. Flory, H. Lee, R. Aebersold, and B. Schwikowski. Signal maps for mass spectrometry-based comparative proteomics. *Mol Cell Proteomics*, 5(3) :423–432, 2006 Mar.

[94] P. Pratapa, E. F. Patz, and A. Hartemink. Finding diagnostic biomarkers in proteomic spectra. *Pacific Symposium on Biocomputing*, 11 :279–290, 2006.

[95] J. T. Prince and E. M. Marcotte. Chromatographic alignment of esi-lc-ms proteomics data sets by ordered bijective interpolated warping. *Anal Chem*, 78(17) :6140–6152, 2006 Sep 1.

[96] Y. Qu, B.-l. Adam, M. Thornquist, J. D. Potter, M. L. Thompson, Y. Yasui, J. Davis, P. F. Schellhammer, L. Cazares, M. Clements, G. L. Wright, and Z. Feng. Data reduction using a discrete wavelet transform in discriminant analysis of very high dimensional data. *Biometrics*, 59 :143–151, 2003.

[97] Y. Qu, B.-l. Adam, Y. Yasui, M. Ward, L. Cazares, P. Schellhammer, Z. Feng, J. Semmes, and W. George. Boosted decision tree analysis of surface-enhanced laser desorption/ionization mass spectral serum profiles discriminates prostate cancer from noncancer patients. *Clinical Chemistry*, 48(10) :1835–1843, 2003.

[98] J. Quackenbush. Computational analysis of microarray data. *Nature Reviews — Genetics*, 2, 2001.

[99] Rakotomamonjy. Variable selection using svm-based criteria. *Journal of Machine Learning Research*, 3, 2003.

[100] T. W. Randolph and Y. Yasui. Multiscale processing of mass spectrometry data. *Biometrics*, 62(2) :589–597, 2006 Jun.

[101] H. W. Ressom, R. S. Varghese, M. Abdel-Hamid, S. A.-L. Eissa, D. Saha, L. Goldman, E. F. Petricoin, T. P. Conrads, T. D. Veenstra, C. A. Loffredo, and R. Goldman. Analysis of mass spectral serum profiles for biomarker selection. *Bioinformatics*, 21(21) :4039–4045, 2005 Nov 1.

[102] M. Rogers and et al. Proteomic profiling of unrinary proteins in renal cancer by surface enhanced laser desorption ionization and neural-network analysis : identification of key issue affecting potential clinical utility. *Cancer Res.*, 63 :6971–6983, 2003.

[103] Y. Saeys, I. Inza, and P. Larranaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19) :2507–2517, 2007 Oct 1.

[104] Y. Sayes, T. Abeel, and Y. Peer. Robust feature selection using ensemble feature selection techniques. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases*, 2008.

[105] O. Semmes, Z. Feng, B.-L. Adam, L. Banez, W. Bigbee, D. Campos, L. Cazares, D. Chan, W. Grizzle, and et al. Evaluation of serum protein profiling by surface-enhanced laser desorption/ionization time-of-flight mass spectrometry for the detection of prostate cancer : I. assessment of platform reproducibility. *Clinical Chemistry*, 51(1) :102–112, 2005.

[106] X.-G. SHAO, A. K.-M. Leung, and F.-T. Chau. Wavelet : A new trend in chemistry. *Acc. Chem. Res.*, 36 :276–283, 2003.

[107] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[108] D. Slotta, L. Heath, N. Ramakrishnan, R. Helm, and M. Potts. Clustering mass spectrometry data using order statistics. *Proteomics*, 3 :1667–1672, 2003.

[109] C. A. Smith, E. J. Want, G. O'Maille, R. Abagyan, and G. Siuzdak. Xcms : processing mass spectrometry data for metabolite profiling using nonlinear peak alignment, matching, and identification. *Anal Chem*, 78(3) :779–787, 2006 Feb 1.

[110] P. Somol and J. Novovicova. Evaluating the stability of feature selectors that optimize feature subset cardinality. In *Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, volume 5342, pages 956–966, 2008.

[111] J. Sorase and M. Zhan. A data review and re-assessment of ovarian cancer serum proteomic profiling. *BMC Bioinformatics*, 4(24) :1471–2105, 2003.

[112] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *J. Mach. Learn. Res.*, 3 :1399–1414, 2003.

[113] G. Strubel. *Reconstruction de profils moléculaires : modélisation et inversion d'une chaîne de mesure protéomique*. PhD thesis, Institut Polytechnique de Grenoble, December 2008.

[114] M. Sturm, A. Bertsch, C. Gropl, A. Hildebrandt, R. Hussong, E. Lange, N. Pfeifer, O. Schulz-Trieglaff, A. Zerck, K. Reinert, and O. Kohlbacher. Openms - an open-source software framework for mass spectrometry. *BMC Bioinformatics*, 9 :163, 2008.

[115] C. H. Teo, Q. Le, A. Smola, and S. Vishwanathan. A scalable modular convex solver for regularized risk minimization. *KDD*, 2007.

[116] R. Tibshirani, T. Hastie, B. Narasimhan, S. Soltys, A. Koong, and Q.-T. Le. Sample classification from protein mass spectroscopy, by peak probability contrasts. Technical report, Departement of Health and Research Policy, Standford University, 2004.

[117] N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In *Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

[118] G. Tomasi, F. Berg, and C. Andersson. Correlation optimized warping and dynamic time warping as preprocessing methods for chromatographic data. *Journal of Chemometrics*, 18 :231–241, 2004.

[119] A. Ultsch. Is log ratio a good value for identifying differential expressed genes in microarray experiments ? *Statistical Computing 2006*, 2006.

[120] B. van Breukelen, H. van den Toorn, M. Drugan, and A. Heck. Statquant : A post quantification analysis toolbox for improving quantitative mass spectrometry. *Bioinformatics*, 2009 Mar 31.

[121] W. Wallace, A. Kearsley, and G. CM. An operator-independent approach to mass spectral peak identification and integration. *Anal Chem*, 76 :2446–2452, 2004.

[122] M. Wang, B. Howard, M. Campa, E. J. Patz, and M. Fitzgerald. Analysis of human serum proteins by liquid phase isoelectric focusing and matrix-assisted laser desorption/ionization-mass spectrometry. *Proteomics*, 3 :1661–1666, 2003.

[123] M. West-Nielsen, E. Hogdall, E. Marchiori, C. Hogdall, C. Schou, and N. Heegaard. Sample handling for mass spectrometric proteomic investigations of human sera. In *Anal Chem*, 2005.

[124] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping. Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3, 2003.

[125] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. *Proceedings of NIPS*, 13, 2000.

[126] I. Witten and E. Frank. *Data Mining : Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

[127] I. Witten and E. Frank. *Data mining : Pratical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, 2005.

[128] Y. Yasui, M. Pepe, M. L. Thompson, A. Bao-ling, G. Wright, Y. Qu, J. Potter, M. Winget, M. Thornquist, and Z. Feng. A data-analytic strategy for protein biomarker discovery : profiling of high-dimensional proteomic data for cancer detection. *Biostatistics*, 4(3) :449–453, 2003.

[129] L. Yu, C. Ding, and S. Loscalzo. Stable feature selection via dense feature groups. In *Processing of the KDD 2008*, 2008.

[130] L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5, 2004.

[131] P. Zerefos, J. Prados, A. Kalousis, and A. Vlahou. Sample preparation and bioinformatics in maldi profiling of urinary proteins. *Submitted to Journal of Chromatography B*, 2006.

[132] X. Zhang, X. Lu, and Q. e. a. Shi. Recursive svm feature selection and sample classification for mass-spectrometry and microarray data. *BMC Bioinformatics*, 7(197), 2006.

Quatrième partie

Annexes

Annexe A

Complément à l'ajustement des paramètres de pré-traitement

Cette annexe contient ne figure qui complète la section 7.2.2. La figure A.1 montre pour chaque jeux de données ovarian, prostate et stk2i208 un histogramme qui reflète le nombre de valeurs manquantes dans la matrice d'expression après alignement des pics.

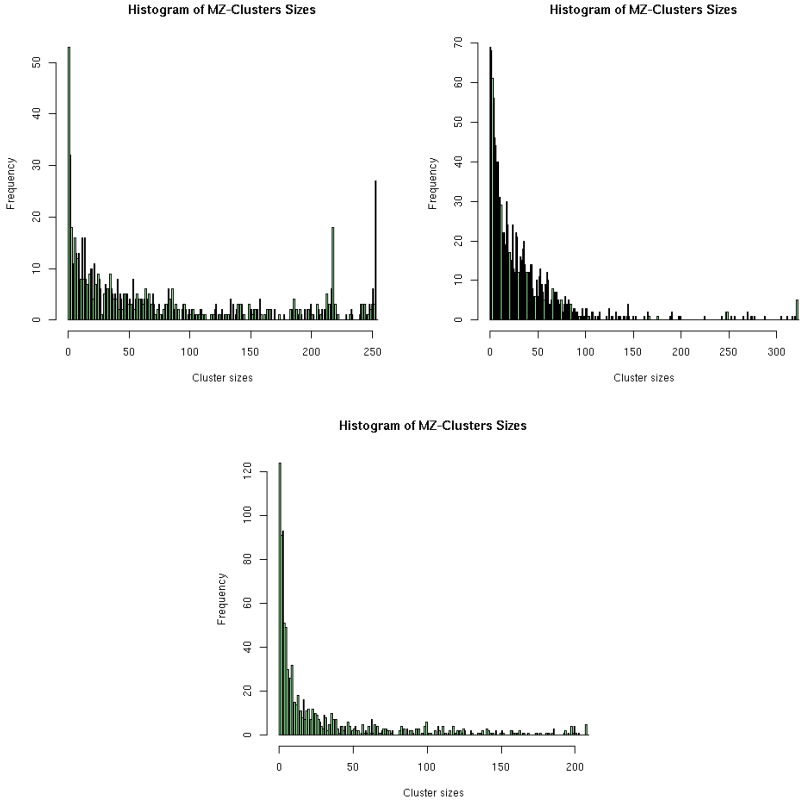


FIGURE A.1 – Histogramme de la taille des clusters résultant de l'alignement par mzcl des pics des différents jeux de donnée dont le rapport signal/bruit est supérieur à 2.5. Rappelons que les clusters sont utilisés pour générer les attributs de la matrice d'expression que l'on utilise pour entrainer les algorithmes d'apprentissage. Les histogrammes nous permettent de caractérisier les valeurs manquantes de la matrice d'expression.

Annexe B

Complément à la comparaison avec PROcess

Cette annexe contient des figures qui complète les analyses de la section 7.3.2. Dans le texte nous n'avions discuté que des résultats de ACCA-UF5kD et DHB-UF5kD, et cette annexe montre les résultats pour tout les jeux de données.

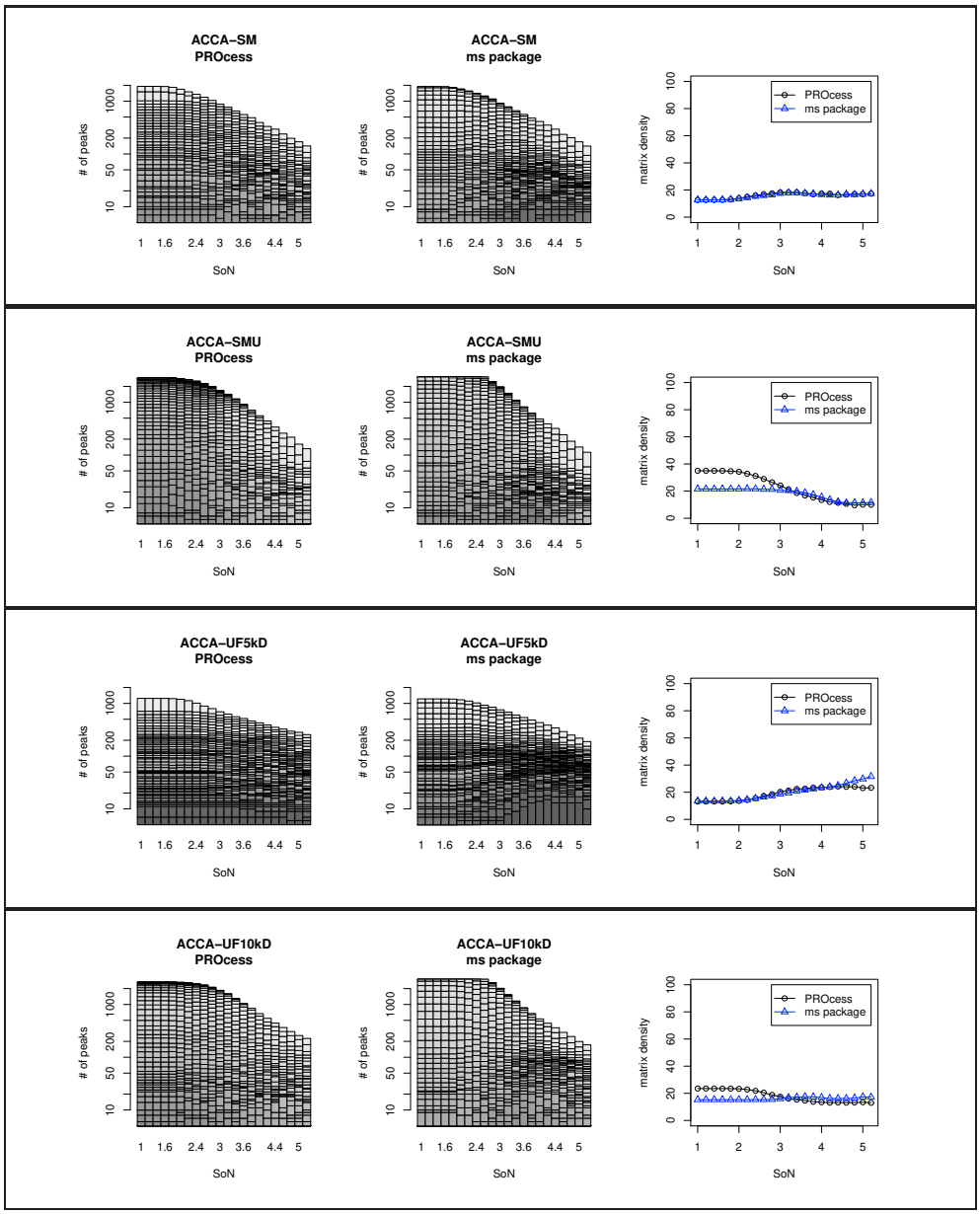


FIGURE B.1 – Ensemble des résultats de stabilité pour les expériences réalisés avec une matrice ACCA, et un paramètre *sm.span=0*

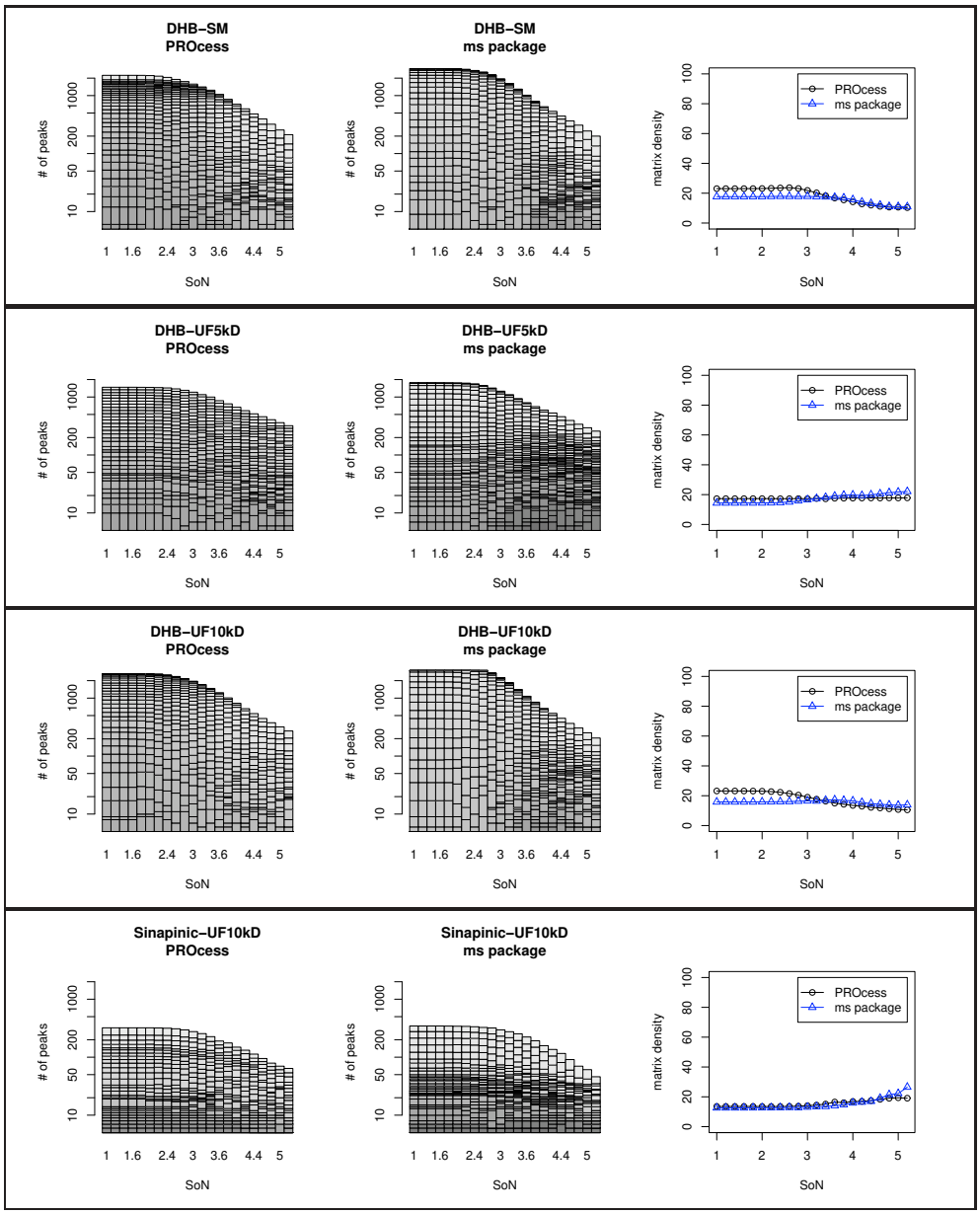


FIGURE B.2 – Ensemble des résultats de stabilité pour les expériences réalisés avec des matrices DHB & Sinapinic, et un paramètre *sm.span=0*

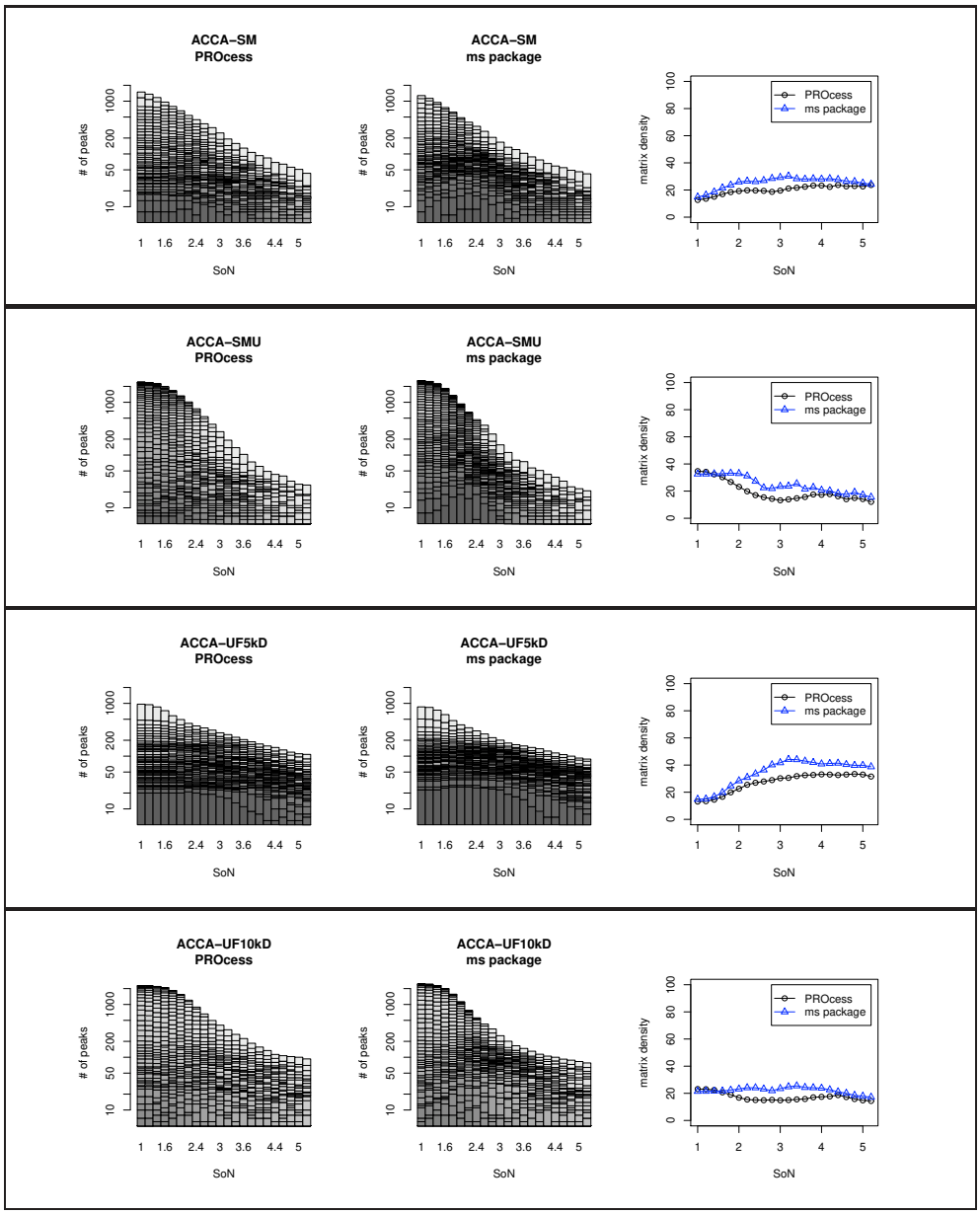


FIGURE B.3 – Ensemble des résultats de stabilité pour les expériences réalisés avec une matrice ACCA, et un paramètre *sm.span=31*

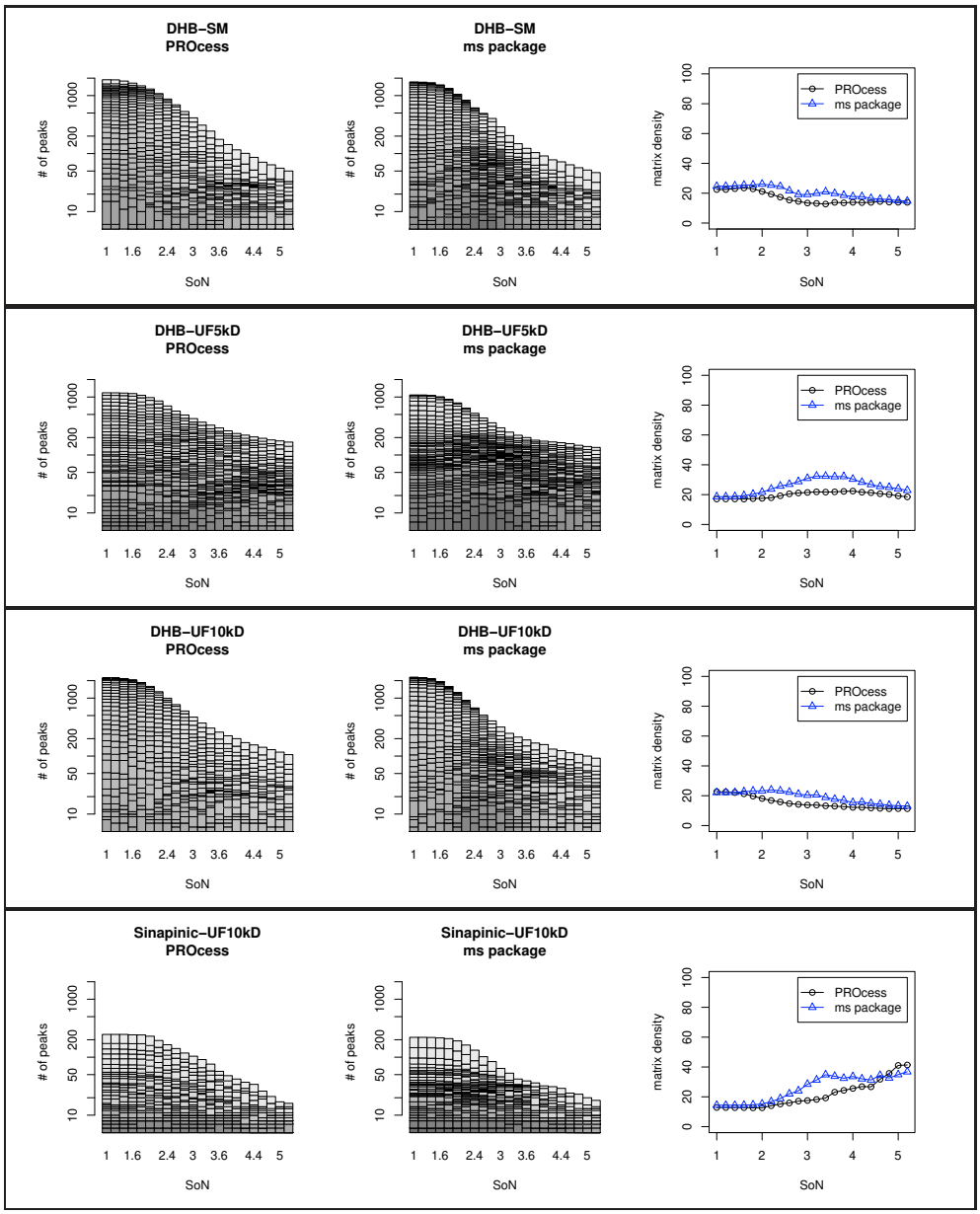


FIGURE B.4 – Ensemble des résultats de stabilité pour les expériences réalisés avec des matrices DHB & Sinapinic, et un paramètre *sm.span=31*

Annexe C

Simplification du noyau *logRatio*

Cette annexe montre le chemin mathématique qui abouti à la simplification du noyau logRatio ($\mathcal{K}$). On part de l'expression du produit scalaire dans l'espace des caractéristiques, et arrive à l'équitation qui exprime le noyau en terme de covariance.

$$\begin{aligned}
\mathcal{K}(\mathbf{x},\mathbf{y}) &= \sum_{i,j} \log\frac{(\mathbf{x})_i}{(\mathbf{x})_j}\log\frac{(\mathbf{y})_i}{(\mathbf{y})_j} && \text{(C.1)}\\
&= \sum_{i,j} (\log(\mathbf{x})_i - \log(\mathbf{x})_j)(\log(\mathbf{y})_i - \log(\mathbf{y})_j) && \text{(C.2)}\\
&= \sum_{i,j} \log(\mathbf{x})_i(\log(\mathbf{y})_i - \log(\mathbf{y})_j) + \log(\mathbf{x})_j(\log(\mathbf{y})_j - \log(\mathbf{y})_i) && \text{(C.3)}\\
&= \sum_{i,j} \log(\mathbf{x})_i(\log(\mathbf{y})_i - \log(\mathbf{y})_j) + \sum_{i,j} \log(\mathbf{x})_j(\log(\mathbf{y})_j - \log(\mathbf{y})_i) && \text{(C.4)}\\
&= \sum_{i,j} \log(\mathbf{x})_i(\log(\mathbf{y})_i - \log(\mathbf{y})_j) + \sum_{j,i} \log(\mathbf{x})_i(\log(\mathbf{y})_i - \log(\mathbf{y})_j) && \text{(C.5)}\\
&= 2\sum_{i,j} \log(\mathbf{x})_i(\log(\mathbf{y})_i - \log(\mathbf{y})_j) && \text{(C.6)}\\
&= 2\sum_{i,j} \log(\mathbf{x})_i \log(\mathbf{y})_i - 2\sum_{i,j} \log(\mathbf{x})_i \log(\mathbf{y})_j && \text{(C.7)}\\
&= 2n\sum_{i=1}^{n} \log(\mathbf{x})_i \log(\mathbf{y})_i - 2\sum_{i,j} \log(\mathbf{x})_i \log(\mathbf{y})_j && \text{(C.8)}\\
&= 2n\sum_{i=1}^{n} \log(\mathbf{x})_i \log(\mathbf{y})_i - 2\left(\sum_{i=1}^{n}\log(\mathbf{x})_i\right)\left(\sum_{j=1}^{n}\log(\mathbf{y})_j\right) && \text{(C.9)}\\
&= 2n\sum_{i=1}^{n} \log(\mathbf{x})_i \log(\mathbf{y})_i - 2\left(\sum_{i=1}^{n}\log(\mathbf{x})_i\right)\left(\sum_{i=1}^{n}\log(\mathbf{y})_i\right) && \text{(C.10)}\\
&= 2n(n-1)\,\mathrm{cov}(\log(\mathbf{x}),\log(\mathbf{y})) && \text{(C.11)}
\end{aligned}$$

Annexe D

Résultats additionnels concernant les performances de classification de SVMRFE-logRatio

Cette section contient des résultats complémentaires pour la section 10.4.3 du chapitre 10. Elle contient les figures qui montrent les détails des résultats obtenus pour les jeux StrokeRaw, OvarianRaw, ProstateRaw.

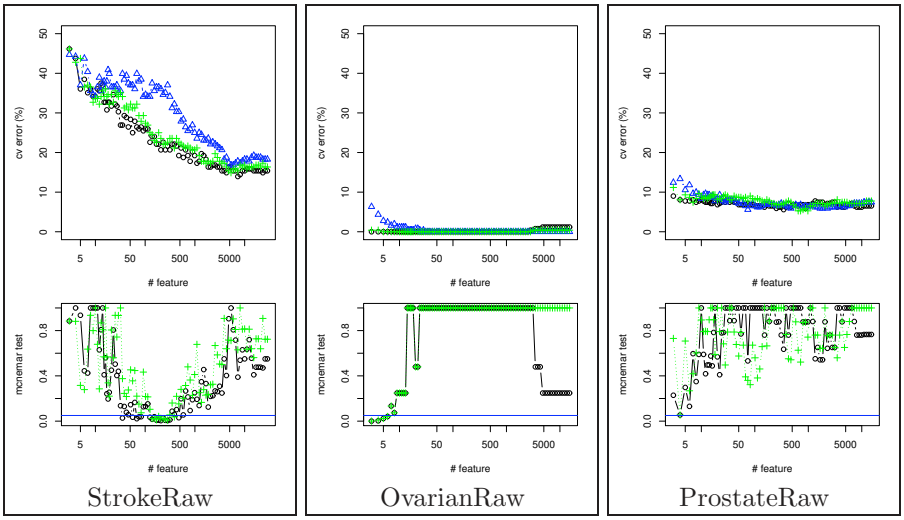


FIGURE D.1 – Résultats addtionnels sur les performances de classification des données MS. Dans chacun des cadres, les trois courbes du haut montrent l'évolution de l'erreur de classification en fonction de la taille (n) des sous-ensembles d'attributs sélectionnés par chaque méthode de sélection d'attribut ; les deux courbes du bas correspondent au test de significativité de McNemar [27] contre les performances du noyau logRatio – lorsque ces courbes sont en-dessous de la ligne 0.05, nous pouvons considérer les erreurs comme significativement différentes.

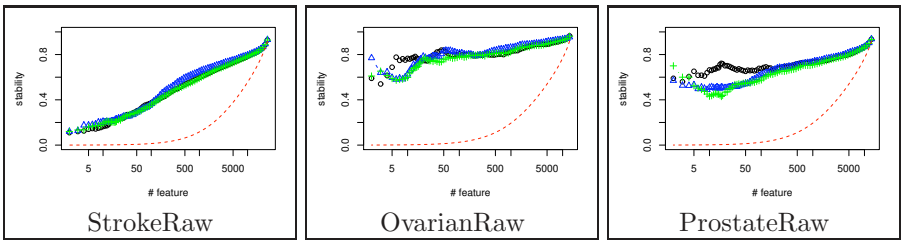


FIGURE D.2 – Évolution de la stabilité des algorithmes de sélection d'attributs SVMRFE en fonction de la taille (n) des sous-ensembles d'attributs sélectionnés sur la totalité des jeux de données. 10% des attributs sont éliminés à chaque itération RFE.

Annexe E

Performance de la sélection d'attributs basée sur un modèle SVM unique (sans itération RFE)

Des expériences identiques à celles menées dans la section 10.4.3 en utilisant des ordonnancements d'attributs d'un seul modèle SVM (sans boucle RFE) à la place de SVMRFE. Les performances de classification des modèles ainsi que la stabilité des modèles obtenus sont données dans les figures qui composent cette annexe. Ces figures sont identiques à celles présentées dans le texte pour SVMRFE.

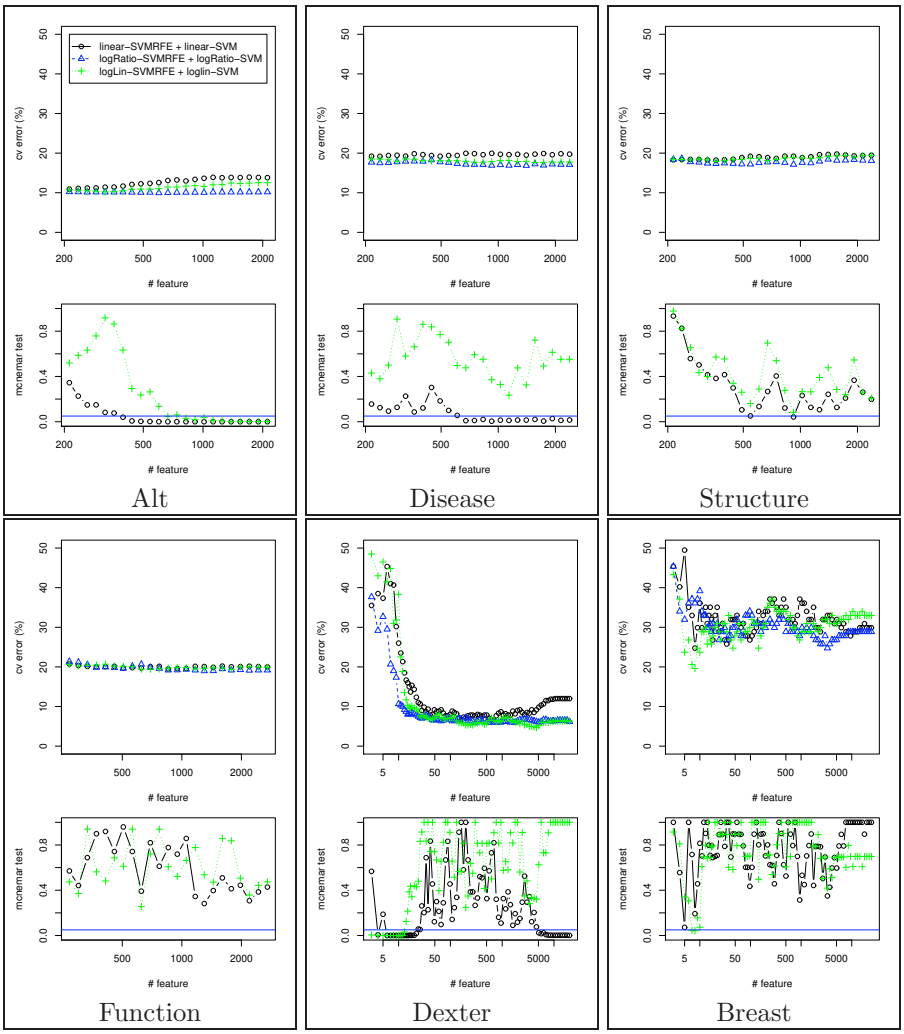


FIGURE E.1 – Evolution de la performance de classification des algorithmes de sélection d'attributs à base de SVM (sans itération RFE) en fonction de la taille (n) des sous-ensembles d'attributs sélectionnés sur les données autres que MS.

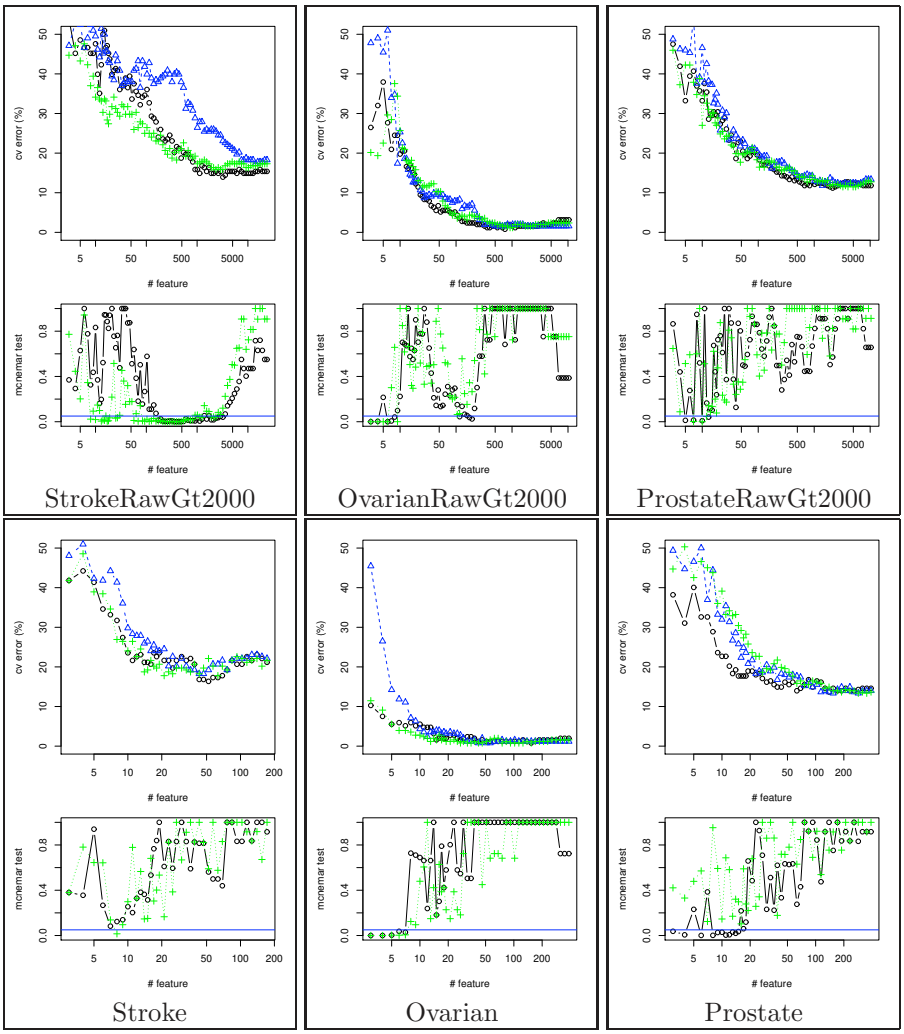


FIGURE E.2 – Evolution de la performance de classification des algorithmes de sélection d'attributs à base de SVM (sans itération RFE) en fonction de la taille (n) des sous-ensembles d'attributs sélectionnés sur une partie des données MS.

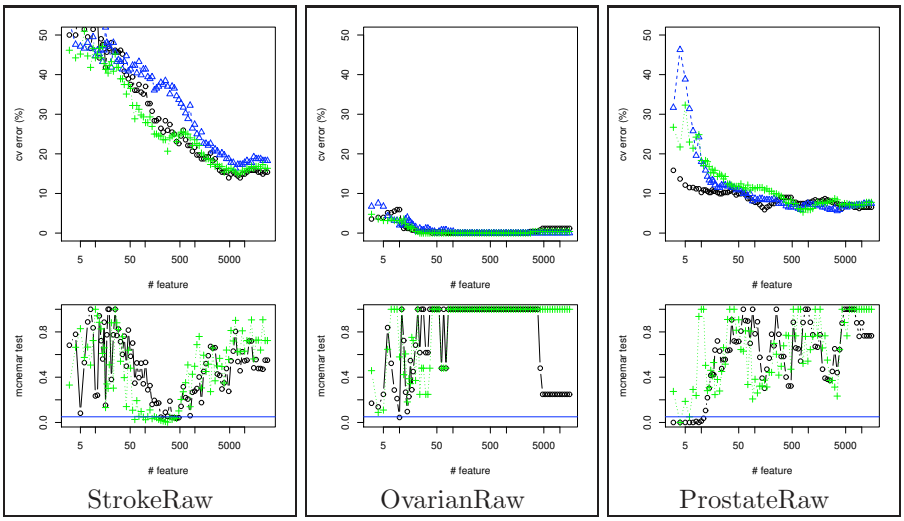


FIGURE E.3 – Evolution de la performance de classification des algorithmes de sélection d'attributs à base de SVM (sans itération RFE) en fonction de la taille (n) des sous-ensembles d'attributs sélectionnés sur une partie des données MS.

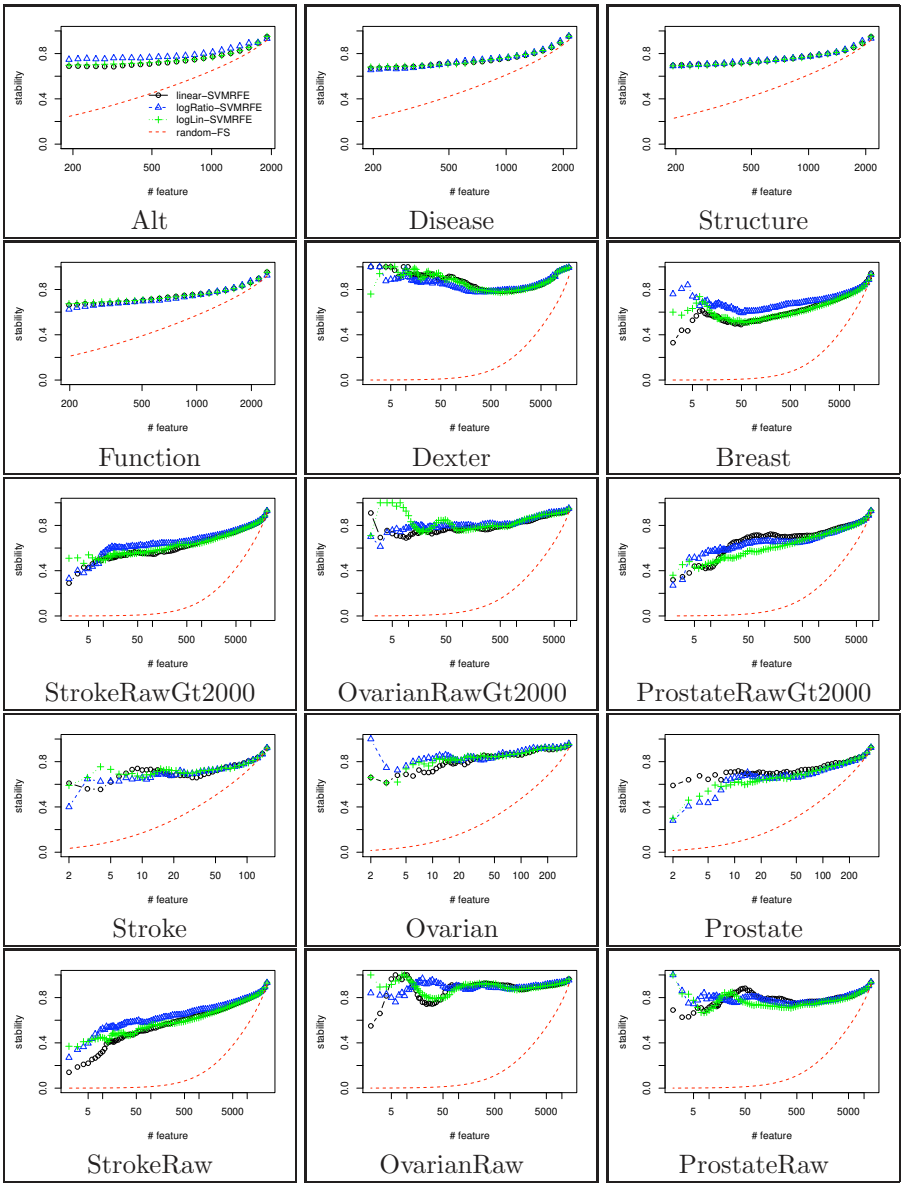


FIGURE E.4 – Évolution de la stabilité des algorithmes de sélection d'attributs à base de SVM (sans itération RFE) en fonction de la taille (n) des sous-ensembles d'attributs sélectionnés sur la totalité des jeux de données.

Annexe F

Attributs sélectionnés par SVMRFE sur données MS

Cette annexe contient les tableaux qui indique les positions des pics sélectionnées par les différentes méthode de sélection d'attributs à base de SVMRFE utilisé dans le texte. Elle complète les résultats de la section 10.4.3, ou seul la version graphique de ces données est présenté.

StrokeRawGt2000			Stroke		
lin	logRatio	logLin	lin	logRatio	logLin
6706.45 (9)	5458.6 (7)	6706.45 (6)	2404.9 (10)	6701.6 (10)	2404.9 (10)
4080.15 (7)	16185.31 (7)	2029.15 (5)	6701.6 (10)	7325.8 (10)	4080.1 (10)
8714.18 (7)	6100.35 (4)	7778.21 (5)	7779 (10)	2024.2 (9)	6701.6 (10)
28349.76 (5)	6706.45 (4)	8717.63 (5)	28201.3 (10)	2404.9 (9)	6952.2 (10)
7778.21 (4)	20021.23 (4)	29949.16 (5)	6952.2 (9)	4080.1 (9)	6958.4 (10)
17345.58 (4)	24814.69 (4)	2404.68 (4)	8715.9 (9)	6958.4 (9)	14384.7 (10)
23451.27 (4)	28051.86 (4)	5075.12 (4)	4080.1 (8)	4639.5 (8)	7325.8 (9)
7781.47 (3)	65337.74 (4)	7567.79 (4)	9142.5 (8)	7779 (8)	28201.3 (9)
8717.63 (3)	2404.68 (3)	28051.86 (4)	14384.7 (8)	9142.5 (8)	8715.9 (8)
10499.91 (3)	2422.83 (3)	28166.49 (4)	5016.3 (7)	6952.2 (7)	11709.4 (8)
12742.33 (3)	2484.16 (3)	65337.74 (4)	7325.8 (7)	14384.7 (7)	2024.2 (7)
24814.69 (3)	4695.46 (3)	24224.42 (3)	2024.2 (6)	4390.5 (6)	3947.4 (7)
28051.86 (3)	4700.53 (3)	27789.4 (3)	9971.4 (6)	8715.9 (6)	4390.5 (7)
30276.24 (3)	4898.95 (3)	28349.76 (3)	15141.7 (6)	8837.8 (6)	7779 (6)
31114.73 (3)	5395.98 (3)	28508.66 (3)	28079.3 (6)	28201.3 (6)	9142.5 (6)
32749.13 (3)	8717.63 (3)	30211.96 (3)	4301.3 (5)	3947.4 (5)	4327.8 (5)
38076.2 (3)	10518.86 (3)	31114.73 (3)	6958.4 (5)	28079.3 (5)	4639.5 (5)
2029.15 (2)	2010.05 (2)	32558.79 (3)	2050.6 (4)	4579.9 (3)	8837.8 (5)
2029.98 (2)	2155.93 (2)	37752.37 (3)	2223.2 (4)	5016.3 (3)	15983.4 (5)
2405.58 (2)	2197.31 (2)	38097.84 (3)	2237.8 (4)	6817.6 (3)	28079.3 (4)

TABLE F.1 – Rapport masse/charge des vingts premiers attributs sélectionnés par les différentes méthodes de sélection à base de SVMRFE sur les deux jeux de données relatifs aux accidents vasculaires cérébraux (StrokeRawGt2000 et Stroke). Les valeurs entre parenthèse indiquent le nombre de fois où l'attribut apparaît parmi les vingt meilleurs dans les différentes validations croisées.

OvarianRawGt2000			Ovarian		
lin	logRatio	logLin	lin	logRatio	logLin
3675.26 (10)	3410.89 (10)	3673 (10)	2870.8 (10)	2318.5 (10)	3533.9 (10)
3676.39 (10)	3675.26 (10)	3674.13 (10)	3674.9 (10)	2419.6 (10)	3674.9 (10)
3674.13 (9)	6681.95 (10)	3675.26 (10)	4003.6 (10)	3674.9 (10)	4438 (10)
4003.64 (9)	3409.8 (9)	3676.39 (10)	5381.3 (10)	4438 (10)	11750.7 (10)
15325.24 (9)	3676.39 (9)	11747.81 (10)	6014.6 (10)	5381.3 (10)	12839.7 (10)
2869.59 (8)	6608.92 (9)	3677.52 (9)	12839.7 (10)	6601.7 (10)	12876.2 (10)
10966.21 (8)	7104.97 (8)	3349.03 (7)	14797.5 (10)	10945.6 (10)	14797.5 (10)
11747.81 (8)	3349.03 (7)	12846.6 (7)	7965.7 (9)	14797.5 (10)	15331.8 (10)
3673 (6)	10974.03 (7)	12848.72 (7)	11750.7 (9)	16426.9 (10)	17092.1 (10)
5377.98 (6)	14723.58 (6)	15334.49 (6)	15331.8 (9)	2451.5 (9)	2419.6 (9)
3410.89 (5)	7283.86 (5)	4441.33 (5)	8038.8 (8)	2870.8 (8)	4003.6 (9)
6014.46 (5)	2022.61 (4)	10966.21 (5)	17092.1 (8)	16585.8 (8)	6014.6 (8)
14807.49 (5)	6584.66 (4)	15385.38 (5)	3201.2 (7)	3533.9 (7)	3412.5 (7)
15322.93 (5)	6598.3 (4)	5377.98 (4)	2419.6 (6)	4890 (7)	5381.3 (7)
2723.52 (4)	6683.48 (4)	12844.49 (4)	3412.5 (6)	4907.7 (7)	7965.7 (7)
14809.76 (4)	7594.55 (4)	12872 (4)	5269.8 (6)	5497.2 (7)	10945.6 (7)
2171.3 (3)	10944.72 (4)	2170.43 (3)	10945.6 (6)	12839.7 (7)	14609.8 (7)
2403.32 (3)	10966.21 (4)	10950.58 (3)	8022.2 (5)	12876.2 (6)	2870.8 (6)
2870.59 (3)	14721.31 (4)	12876.24 (3)	11029.1 (5)	17092.1 (6)	4907.7 (4)
3677.52 (3)	16368.33 (4)	15205.33 (3)	3071.2 (4)	3412.5 (5)	10780 (4)

TABLE F.2 – Rapport masse/charge des vingts premiers attributs sélectionnés par les différentes méthodes de sélection à base de SVMRFE sur les deux jeux de données relatifs aux cancers des ovaires (OvarianRawGt2000 et Ovarian). Les valeurs entre parenthèse indiquent le nombre de fois où l'attribut apparaît parmi les vingts meilleurs dans les différentes validations croisées.

ProstateRawGt2000			Prostate		
lin	logRatio	logLin	lin	logRatio	logLin
3484.31 (10)	3472.2 (6)	4078.39 (8)	2006.2 (10)	12266.5 (10)	4133.4 (10)
12300.38 (7)	3819.22 (6)	2743.04 (5)	4133.4 (10)	6926.5 (9)	9050.1 (10)
4096.29 (6)	2450.21 (5)	3009.23 (5)	6130.4 (10)	13176.9 (9)	12175.9 (9)
4111.83 (6)	2627.94 (5)	3140.67 (5)	11430 (10)	14332.9 (9)	13176.9 (9)
4493.73 (6)	2743.04 (5)	14329.9 (5)	6926.5 (9)	14451.2 (9)	14404.2 (9)
6447.56 (5)	9061.38 (5)	2277.82 (4)	6940.7 (8)	4133.4 (8)	14332.9 (8)
14318.73 (5)	14444.08 (5)	2450.21 (4)	13176.9 (7)	12175.9 (8)	14451.2 (8)
4093.9 (4)	3722.94 (4)	3231.27 (4)	8590.2 (6)	16333.6 (8)	3469.4 (7)
4114.23 (4)	4109.44 (4)	3436 (4)	12175.9 (6)	2053.6 (7)	5249.9 (6)
5353.37 (4)	5250.08 (4)	3472.2 (4)	13910.3 (6)	5249.9 (7)	6926.5 (6)
12298.31 (4)	5511.57 (4)	4077.2 (4)	14332.9 (6)	6130.4 (7)	8741.5 (6)
19777.05 (4)	14455.3 (4)	11959.15 (4)	19501.8 (6)	2310 (5)	12403.8 (6)
2006.69 (3)	2637.52 (3)	16109.08 (4)	3840.1 (5)	6914.5 (5)	16333.6 (6)
3154.28 (3)	2815.86 (3)	18876.71 (4)	8954.1 (5)	7127.3 (5)	12266.5 (5)
3483.21 (3)	3462.31 (3)	2924.85 (3)	12805.9 (5)	9050.1 (5)	13910.3 (5)
3562.98 (3)	4111.83 (3)	3819.22 (3)	14451.2 (5)	12403.8 (5)	2053.6 (4)
4095.09 (3)	4435.12 (3)	4508.76 (3)	16110.4 (5)	3469.4 (4)	6130.4 (4)
4296.99 (3)	4493.73 (3)	9061.38 (3)	19814.1 (5)	13910.3 (4)	11430 (4)
5354.73 (3)	13825 (3)	12172.37 (3)	5981.3 (4)	14404.2 (4)	13455.2 (4)
6323.76 (3)	14448.57 (3)	14444.08 (3)	6914.5 (4)	16110.4 (4)	16110.4 (4)

TABLE F.3 – Rapport masse/charge des vingts premiers attributs sélectionnés par les différentes méthodes de sélection à base de SVMRFE sur les deux jeux de données relatifs aux cancer de la prostate (ProstateRawGt2000 et Prostate). Les valeurs entre parenthèse indiquent le nombre de fois où l'attribut apparaît parmi les vingts meilleurs dans les différentes validations croisées.