Interpretable Machine Learning for Biomarker Discovery in Imaging Mass Spectrometry Data

Leonoor Ella Marie Tideman

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Interpretable Machine Learning for Biomarker Discovery in Imaging Mass Spectrometry Data

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Leonoor Ella Marie Tideman

December 18, 2019

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

VANDERBILT
UNIVERSITY

We kindly acknowledge the Mass Spectrometry Research Center of Vanderbilt University in Nashville, Tennessee, United States of America, for providing the two imaging mass spectrometry datasets used for this thesis. Our special thanks go to Dr. Caprioli and Dr. Spraggins.

TUDelft Delft University of Technology

DCSC

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Interpretable Machine Learning for Biomarker Discovery in Imaging Mass Spectrometry Data

by

Leonoor Ella Marie Tideman

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: December 18, 2019

Supervisor: Dr. R. Van de Plas

Readers: Dr. R. Van de Plas

Dr. S. Wahls

Dr. O. Soloviev

Dr. W. Pan

Abstract

Imaging mass spectrometry (IMS) is a multiplexed chemical imaging technique that enables the spatially targeted molecular mapping of biological samples at cellular resolutions. Within a single experiment, IMS can measure the spatial distribution and relative concentration of thousands of distinct molecular species across the surface of a tissue sample. The large size and high-dimensionality of IMS datasets, which can consist of hundreds of thousands of pixels and hundreds to thousands of molecular ions tracked per pixel, have made computational approaches necessary for effective analysis. This thesis focuses primarily on biomarker discovery in IMS data using supervised machine learning algorithms. Biomarker discovery is the identification of molecular markers that enable the recognition of a specific biological state, for example recognizing diseased tissue from healthy tissue. Biomarkers are increasingly used in biology and medicine for diagnostic and prognostic purposes, as well as for driving the development of new drugs and therapies. Traditionally, the focus has been on maximizing the predictive performance of supervised machine learning models, without necessarily examining the models' internal decision-making processes. Yet, in order to generate insight into the underlying chemical mechanism of disease or drug action, we must go beyond the scope of just prediction and learn how these empirically trained models make their decisions and who are the primary chemical drivers of this prediction process. Machine learning model interpretability is the ability to explain a model's predictions, and can practically be translated into the ability to explicitly report the relative predictive importance of each of the dataset's features. When analyzing IMS data, interpretability is crucial for understanding how the spatial distribution and relative concentration of certain molecular features relate to the labeling of pixels into different physiological classes. The key to our data-driven approach to biomarker discovery in IMS data is to establish (in relation to a specific biomedical recognition task) a means of ranking the molecular features of supervised machine learning models according to their respective predictive importance scores. Ensuring model interpretability and feature ranking in supervised machine learning allows empirical model building to be used as a filtering mechanism to rapidly determine, among thousands of features, those features that exert a large amount of relevance to a specific class determination. With regards to biology, the top-ranking features can help empirically highlight important molecular drivers in the biological process under examination, and can help generate new hypotheses. In terms of translational medicine, such top-ranking features can yield a shortlist of candidate biomarkers worthy of further clinical investigation. Three different classifiers, namely logistic regression, random forests, and support vector machines, are implemented and their performance is compared in terms of accuracy, precision, recall, scale invariance, sensitivity to noise, and computational efficiency. Subsequently, several approaches to explaining these classifiers' predictions are implemented and investigated: model-specific interpretability methods are tied to intrinsically interpretable classifiers, such as generalized linear models and decision trees, whereas model-agnostic interpretability methods can also explain the predictions of black-box models, such as support vector machines with nonlinear kernels or deep neural networks. In addition to three model-specific methods, we present two post-hoc model-agnostic interpretability methods: permutation importance and Shapley importance. Our implementation of Shapley importance, based on Shapley values from cooperative game theory, is novel. Having observed a variability between the rankings of different interpretability methods, we investigate improving the inter-method reliability of feature rankings by decorrelating the features prior to training the classifiers. We also propose a robust ensemble approach to interpretability that aggregates the importance scores attributed to each feature by different model-specific interpretability methods. We demonstrate our methodology on two biomedical case studies: one MALDI-FTICR IMS dataset taken from the coronal section of a rat brain, and one MALDI-TOF IMS dataset taken from the sagittal section of a mouse-pup.

Acknowledgements

I would first like to thank my supervisor Dr. R. Van de Plas for introducing me to imaging mass spectrometry data analysis and guiding me in my work during the past year. I would also like to thank my friends and family for accompanying me and supporting me throughout the course of my engineering studies.

Delft, University of Technology
December 18, 2019

Leonoor Ella Marie Tideman

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The aim of this thesis is to propose a machine learning framework for discovering biomarkers in large-scale high-dimensional imaging mass spectrometry data. A biomarker is a measurable indicator of a specific biological condition, for example a disease, that may be used for diagnostic, prognostic, or screening purposes [11]. We first introduce imaging mass spectrometry and then briefly explain the fundamentals of supervised and unsupervised machine learning. Rather than focus exclusively on predictive performance, we stress the importance of classifier interpretability for biomarker discovery.

1-1 Imaging Mass Spectrometry Data

1-1-1 Mass Spectrometry

Imaging mass spectrometry (IMS) is an imaging technique, based on mass spectrometry (MS), that enables the spatially localized chemical analysis of a sample. IMS is essentially the addition of spatial information to the chemical information provided by MS. The idea of MS is to generate ions from either inorganic or organic compounds, to separate these ions by their mass-to-charge ratios (m/z), and to detect their abundance per m/z [1]. MS provides high chemical specificity and a broad analyte coverage: it enables the concurrent detection, characterization and identification of a wide range of analytes, including metabolites, pharmaceutical drugs, lipids, peptides, and proteins [12]. The working principle of MS is based on the ability that magnetic and/or electric fields have to influence the motion of charged atoms or molecules, usually in a vacuum, in relation to their mass and charge [13]. The force developed by an electromagnetic field relates directly to the particle's charge: the motion of a higher-charged particle will be more affected by a given field than that of a lesser-charged particle, assuming both particles have the same mass [12]. Conversely, the motion of a heavier particle will be more affected by a field than that of a lighter particle, assuming both particles have the same charge [12]. We can therefore determine the mass-to-charge ratio (m/z value) of a particle by examining its trajectory under the influence of a given electromagnetic field.

MS is essentially a four step process that involves sample preparation, desorption/ionization, separation, and detection [14]. As illustrated in Figure 1-1, a mass spectrometer consists of an ion source, where the analytes are ionized, a mass analyzer, where the ions are sorted according to their mass-to-charge ratios, and a detector, which records the ions' relative abundances [15]. A typical MS experiment starts with the vaporization and ionization of the analytes from the sample's surface. The ionized analytes are then directed to a mass spectrometer, where they are separated and detected based on their mass-to-charge ratios. The output of an MS experiment is a mass spectrum, where m/z values are plotted along the x-axis and the relative intensity of the analyte signals are plotted against the y-axis.

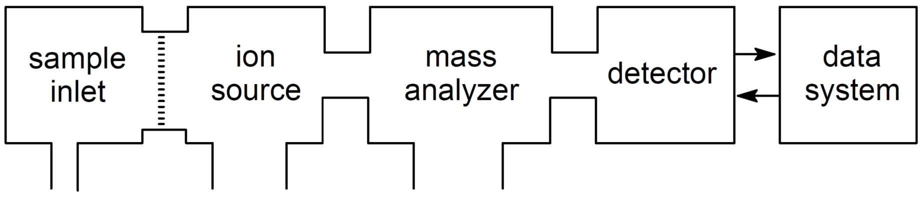


Figure 1-1: Diagram of the general setup of a mass spectrometer. Figure adapted from [1].

Three commonly used ionization/desorption methods, called matrix assisted laser desorption ionization (MALDI), secondary ion mass spectrometry (SIMS), and desorption electrospray ionization (DESI), are illustrated in Figure 1-2. The target plate on which the sample (i.e. usually a tissue section) is mounted must be conductive for MALDI and SIMS, but not for DESI [2]. MALDI and SIMS require the sample to be ionized in a vacuum, whereas DESI allows for ionization under ambient conditions [16].

- MALDI uses pulsed laser light (i.e. ultraviolet or infrared) as a source of energy for the desorption and ionization of analytes [1]. MALDI requires that the tissue section be coated by a light-absorbent solution, called a chemical matrix, prior to ionization. Different matrix types promote the desorption and ionization of different classes of analytes [12]. Analyte molecules are extracted from the sample and incorporated into the matrix during evaporation and crystallization [3], yielding analyte-doped matrix crystals as illustrated in Figure 1-3. Ionization is achieved through exposure of the crystallized matrix to pulsed laser light. The energy intake upon laser irradiation causes the matrix to rapidly heat up and desorb [3]. MALDI is a soft ionization technique that enables the ionization and desorption of heavy macro-molecules, such as proteins, with minimal fragmentation [1]. Thanks to its wide mass range and tolerance to varying sample geometry, MALDI is widely used for imaging mass spectrometry of biological samples, such as animal and human tissue sections [16].

- SIMS is the oldest ionization/desorption method. Under high vacuum, a beam of pulsed high-energy primary ions is focused onto the sample, physically perturbing the surface by a process known as "sputtering", and generating secondary ions [16]. SIMS performs hard ionization and is therefore prone to fragmentation [16], which is especially problematic for the analysis of large molecular species.

- DESI is a relatively new soft ambient ionization method that allows samples to be analyzed without the need for extensive sample preparation [12]. A pneumatically assisted electrospray ionization source is used to impact charged solvent droplets on the surface to be analyzed, resulting in analytes being solvated and ionized from the sample's surface [13].

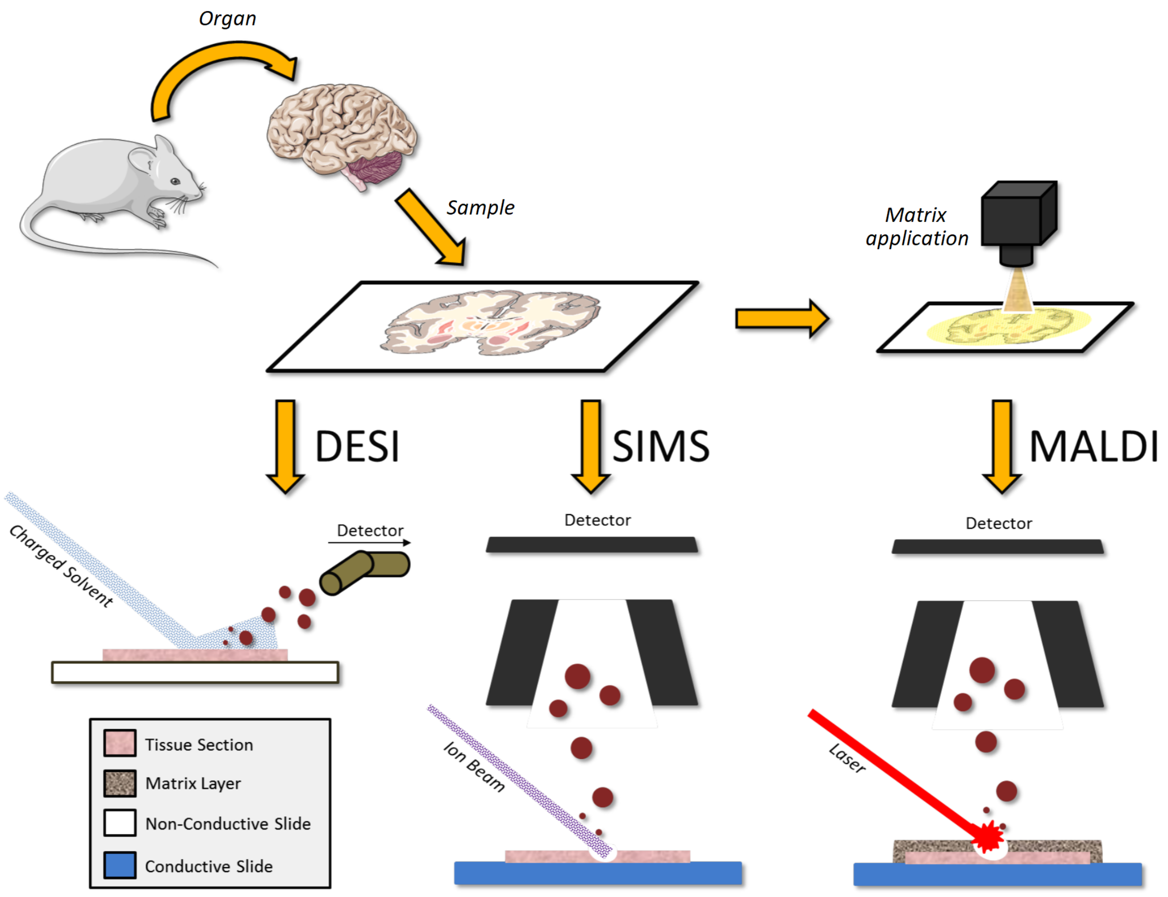


Figure 1-2: Overview of three mass spectrometry ionization/desorption methods: desorption electrospray ionization (DESI), secondary ion mass spectrometry (SIMS) and matrix assisted laser desorption ionization (MALDI). Figure adapted from [2]. Copyright © Bentham Science Publishers.

Two commonly used mass analyzers are time-of-flight (TOF) analyzer and the Fourier transform ion cyclotron resonance (FTICR) analyzer [12]. We briefly explain their respective modes of operation.

- A TOF mass analyzer separates ions with different m/z values by their respective times-of-flight. The ionized analytes are unidirectionally accelerated by an electrostatic field and move through a field-free drift path [12]. Since all ions are accelerated with the same kinetic energy, lighter ions of a certain charge will travel faster than heavier ones and, therefore, reach the detector earlier [12]. We consider an ion with mass m and charge z subject to an acceleration voltage U. The potential electric energy P is converted into kinetic energy K such that the velocity v attained by the ion is that of Equation

1-1. In Equation 1-2, the time t needed to travel a field-free distance d is shown to be proportional to the square root of m/z [1].

$$P = K \iff zU = \frac{1}{2}mv^2 \iff v = \pm\sqrt{\frac{2zU}{m}} \tag{1-1}$$

$$t = \frac{d}{v} = \frac{d}{\pm\sqrt{\frac{2zU}{m}}} \iff \frac{m}{z} = \frac{2Ut^2}{d^2} \tag{1-2}$$

- An FTICR mass analyzer separates ions with different m/z values by their respective cyclotron frequencies. The ions are trapped inside an ion cyclotron resonance cell where they are subject to a uniform magnetic field [1]. The ions orbit perpendicularly to the magnetic field, written B, by action of the Lorentz force F_L defined in Equation 1-3 [1].

  $$F_L = zv \times B \tag{1-3}$$

  The working principle of FTICR is based on cyclotron motion. The radius R of an ion with mass m and charge z, subject to a magnetic field of strength B, increases proportionally to its velocity v. The cyclotron angular frequency ω_c of such an ion is given by Equation 1-5. Equation 1-6 shows that the cyclotron frequency f_c of an ion is proportional to its charge z and inversely proportional to its mass m [1]. A group of ions with the same m/z value will therefore have the same cyclotron frequency f_c.

  $$R = \frac{mv}{zB} \tag{1-4}$$

  $$\omega_c = \frac{v}{R} = \frac{zB}{m} \tag{1-5}$$

  $$f_c = \frac{\omega_c}{2\pi} = \frac{zB}{2\pi m} \iff \frac{m}{z} = \frac{B}{2\pi f_c} \tag{1-6}$$

  The selective acceleration of groups of ions with the same m/z value is achieved by resonant excitation, that is the application of a transverse electric field that alternates at their cyclotron frequency f_c [1]. The cyclotron frequency of each group of ions is recorded by image current detection and converted into a mass spectrum using a Fourier transform [1].

Mass spectrometer performance is generally measured in terms of sensitivity and specificity. TOF is known for its high sensitivity and FTICR is know for its high specificity [12]. Sensitivity is a measure of the ability to discriminate between small differences in analyte concentrations or intensities. It corresponds to the slope of the calibration curve, relating signal intensity to the mass or concentration of analyte [1]. Sensitivity is not to be confused with the limit of detection, which is the minimum amount of analyte required to obtain a signal that can be distinguished from the background noise [1]. Mass resolution defines the degree of chemical specificity [17]. It is a measure of the uniqueness of MS peak assignment, that is the ability to distinguish analytes of slightly different mass-to-charge ratios [17]. Mass resolution usually refers to the division of the observed mass-to-charge ratio by the smallest mass-to-charge difference of two separable ions. Note the difference between mass spectral measurement sensitivity/specificity and statistical sensitivity/specificity (see section 1-2).

1-1-2 Imaging Mass Spectrometry

Imaging mass spectrometry (IMS) is essentially the use of mass spectrometry (MS) to visualize the spatial distribution and relative concentration of a wide range of analytes, ranging from atomic to macromolecular ions, in a sample [15]. IMS consists of scanning the sample's surface with a mass spectrometer: one small, localized region of the sample, termed a pixel, is analyzed at a time and the resulting mass spectrum is recorded along with its corresponding (x,y) spatial coordinates [18]. This process is repeated pixel-by-pixel until the entire sample surface has been examined. The result is an IMS data cube of localized mass spectra.

Matrix assisted laser desorption/ionization (MALDI) is one of the most generally used ion sources for imaging mass spectrometry (IMS) [15]. The workflow of MALDI-IMS is illustrated by Figure 1-3: we start by thinly sectioning the sample, mounting it onto a conductive slide and coating it with a light absorbing matrix. Matrix application refers to the deposition and subsequent evaporation of a matrix on the tissue section [14]. What follows is the desorption and ionization of the analyte-doped crystallized matrix by laser irradiation of designated regions of the tissue section [3]. A mass spectrometer, for example a TOF or FTICR instrument, sequentially generates a mass spectrum for each pixel. The spatial resolution is determined by the size of the laser spot at the sample surface, the spacing between pixels, and the sample preparation processes [19].

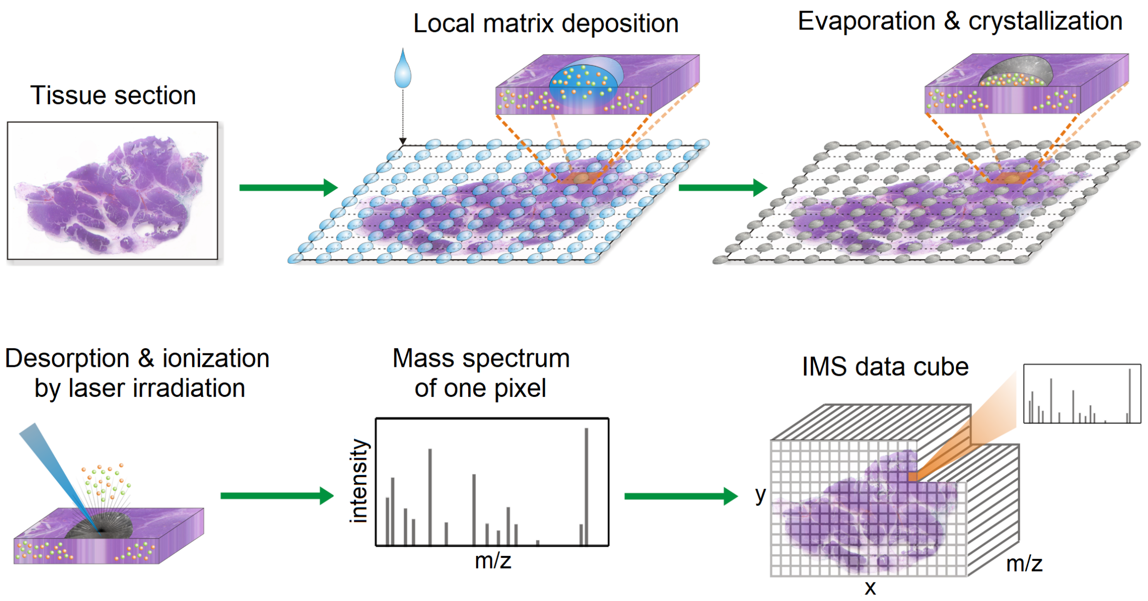

Local matrix deposition
Evaporation & crystallization
Tissue section
Desorption & ionization by laser irradiation
Mass spectrum of one pixel
intensity
m/z
IMS data cube
y
x
m/z

Figure 1-3: Schematic of a MALDI-IMS experiment: sample preparation, matrix deposition, desorption/ionization using a laser, and analyte detection by a mass analyzer. Figure adapted from [3]. Copyright© 2011 Jones et al. Creative Commons Attribution License.

Figures 1-4 and 1-5 show the two possible manners of examining IMS data cubes: one mass spectrum per pixel as in Figure 1-4, or one ion image per m/z value as in Figure 1-5. Figure 1-4 represents the IMS data cube as a collection of mass spectra recording the chemical composition of each pixel. However, IMS also produces ion images that map the spatial distribution and relative concentration of each of its analytes across the sample surface [13]. In Figure 1-5, each ion image provides information about the localization and signal intensity of one specific analyte, which is characterized by a specific m/z value. Ion images are displayed using a pseudo-color scale whose brightness is indicative of the signal intensity measured at a given location. Signal intensity correlates with the relative analyte concentration at that location [12].

IMS 3-D tensors may be reshaped into 2-D matrices where each row is the mass spectrum of a given pixel and each column provides the spatial distribution and relative abundance of a given analyte. It is the 2-D representation of IMS data, illustrated in Figure 1-6, that we use from here-onward.

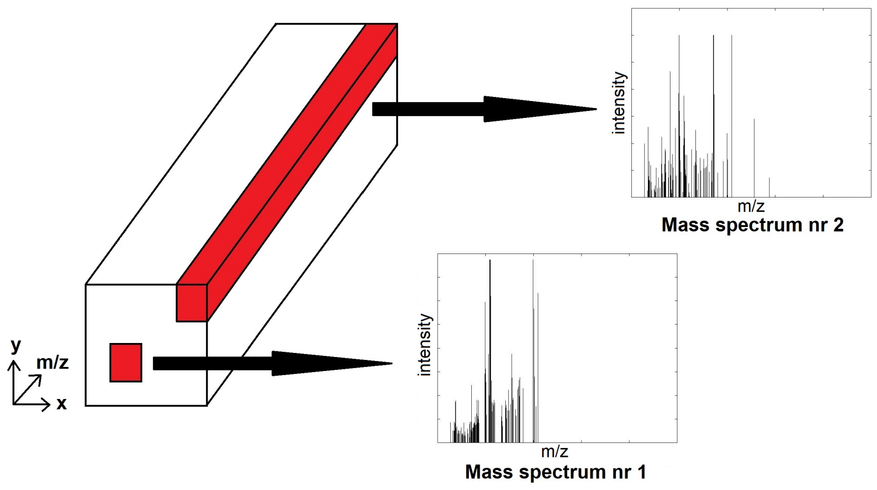


Figure 1-4: IMS data cube: Each pixel of the sample, defined by its spatial coordinates x and y, is chemically characterized by a mass spectrum. The mass spectrum plots the measured signal intensity of all analytes against their respective m/z values.

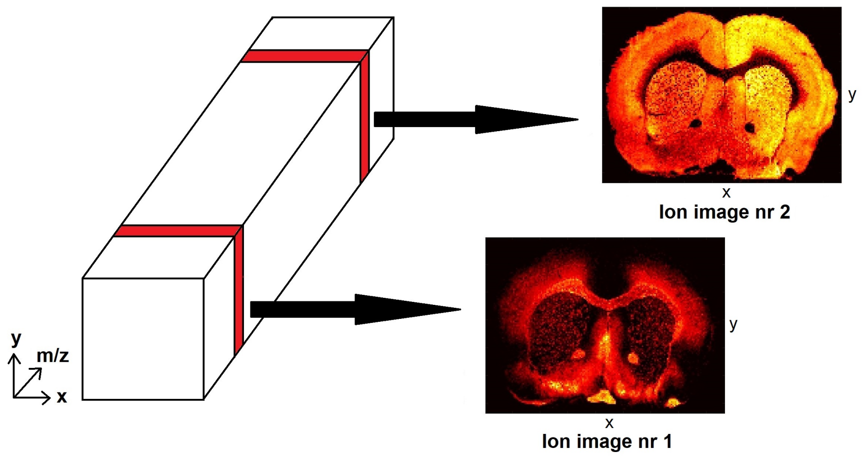


Figure 1-5: IMS data cube: Each analyte's spatial distribution is represented by an ion image whose pseudo-color scale indicates increasing relative concentration by increasing brightness.

1-2 Machine Learning

1-2-1 Basic Terminology

Machine learning is a field of artificial intelligence (AI) concerned with the development and application of computer algorithms that automatically learn from data and improve with experience [20]. The data instances and predictor variables used in machine learning tasks are respectively called observations (or examples) and features. Imaging mass spectrometry (IMS) data is represented as a m-by-n matrix X in which each row is an observation (i.e. mass spectrum) and each column is a feature (i.e. m/z value). As illustrated in Figure 1-6, the rows of X are written $x_i = X_{(i,:)}$, for $i = 1, 2...m$ and the columns of X are written $x^j = X_{(:,j)}$, for $j = 1, 2...n$. Supervised and unsupervised learning differ from each other in that supervised learning requires the observations to be labeled, whereas unsupervised learning does not [20]. Supervised methods focus on modeling a specific recognition task, whereas unsupervised methods focus on discovering patterns in the data. We provide an overview of machine learning methods for the analysis of IMS data: Chapter 2 presents two unsupervised methods, namely principal component analysis (PCA) and non-negative matrix factorization (NNMF), and Chapter 3 presents three supervised methods, namely logistic regression (LR), support vector machine (SVM), and random forest (RF) classifiers.

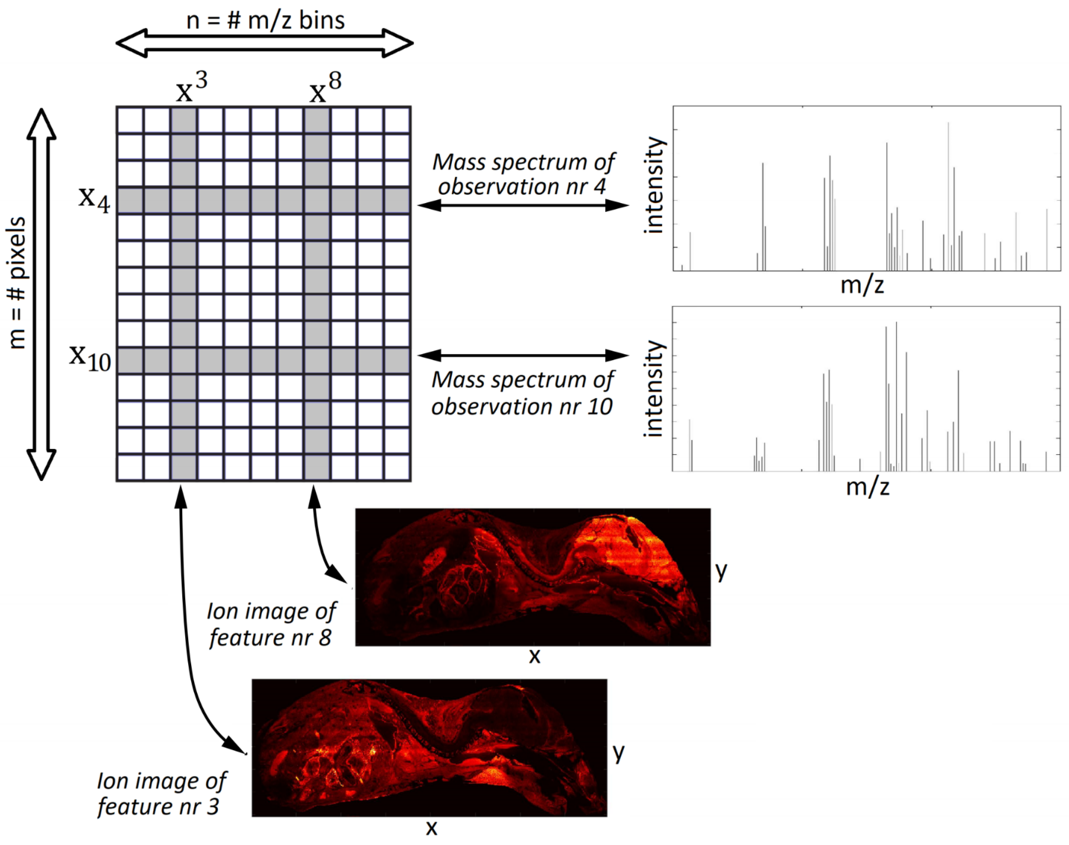

n = # m/z bins
x3
x8
m = # pixels
x4
x10
Mass spectrum of observation nr 4
Mass spectrum of observation nr 10
intensity
m/z
intensity
m/z
Ion image of feature nr 8
Ion image of feature nr 3
y
x
y
x

Figure 1-6: IMS data matrix X where each observation $x_i = X_{(i,:)}$, for $i = 1, 2...m$ corresponds to a pixel and each feature $x^j = X_{(:,j)}$, for $j = 1, 2...n$, corresponds to an m/z bin.

Classification is a form of supervised learning in which observations x_i are annotated with discrete class labels y_i for $i = 1, 2...m$ [20]. We focus on binary classification problems involving a positive class, labeled $y_i = +1$, and a negative class, labeled $y_i = -1$. Training a model requires a learning algorithm to produce a function $f : \mathbb{R}^n \to \{-1, +1\}$ that assigns each observation x_i to either the positive or negative class [4]. The machine learning model f is called a classifier: it performs a mapping from inputs (i.e. observations) to outputs (i.e. class labels). Given an observation x_i, the classifier's prediction $\hat{y}_i = f(x_i)$ is correct if and only if $\hat{y}_i = y_i$. Learning is a matter of tuning the model's parameters to minimize a target function that is proportional to the prediction error of the model, given an input X, called the training data, and output y [21]. Parametric models assume the input-output mapping f to have a certain form, which is determined by a fixed set of parameters, whereas non-parametric models do not [22]. The performance of a model is measured by its ability to generalize, that is to correctly predict the labels of new data instances, called the testing data [21]. Overfitting is a common pitfall that occurs when a model adapts too closely to the training data and memorizes not only the relationship between inputs and outputs but also the noise [20]. Such a model cannot generalize well to new data and will therefore perform poorly on the testing set. The risk of overfitting increases when handling high-dimensional datasets, that is datasets with a large number of features, or when the algorithm is not provided with enough training instances [20]. The difficulty of analyzing high-dimensional data is referred to as a the "curse of dimensionality" [22]. Regularization refers to any modification we make to a machine learning algorithm that is intended to reduce the testing error but not the training error of the resulting model: it is therefore an effective tool to limit overfitting [4].

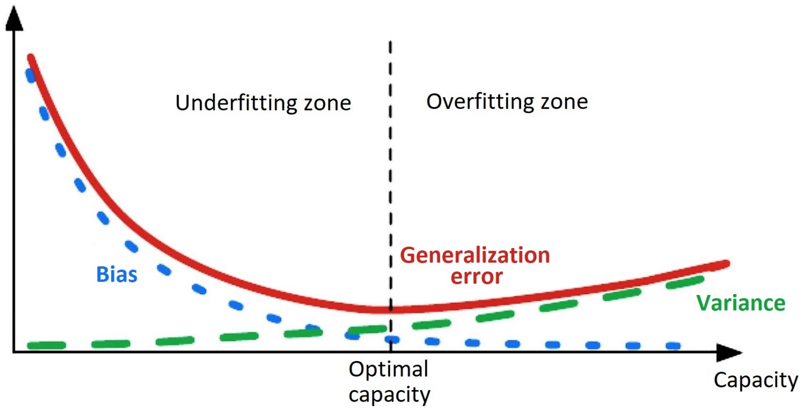


Figure 1-7: Typical relationship between classifier capacity, bias, and variance: bias decreases and variance increases as capacity increases. Models with below-optimal capacity risk underfitting (i.e. high training and testing error) whereas models with above-optimal capacity risk overfitting (i.e. low training error but high testing error). Figure adapted from [4].

The problem of overfitting comes from the bias-variance trade-off. Bias and variance are two sources of predictive error: bias is a measure of how accurately the model estimates the relationship between features and target labels on the training set, whereas variance is a measure of how specific a model is to its training set [23]. Low bias comes at the expense of high variance and, conversely, low variance comes at the expense of high bias. In Figure 1-7, bias and variance are plotted against capacity. Capacity, which is usually quantified by the Vapnik–Chervonenkis (VC) dimension, is a measure of a model's complexity and flexibility [4]. The VC dimension of a model generally coincides with the number of degrees of freedom, or parameters, in the model [24]. Simple models have low variance but high bias and therefore tend to underfit the training data, whereas complex models have low bias but high variance and therefore tend to overfit the training data [23].

We now review a few commonly used measures of predictive performance for supervised learning algorithms used to solve binary classification problems. Classification problems with multiple target classes can be decomposed into multiple binary tasks, each of which involves evaluating one class against the remaining classes, according to the one-versus-all (OVA) approach [23].

- Accuracy: accuracy is the proportion of all predictions that are correct. Maximizing accuracy is equivalent to minimizing the prediction error rate.
- Precision, recall and the F-score: precision is the proportion of all positive predictions that are correct, whereas recall is the proportion of positive instances identified [20]. Precision and recall are defined by Equations 1-7 and 1-8 respectively. A correct positive prediction is called a true positive (TP), a correct negative prediction is called a true negative (TN), an incorrect negative prediction is called a false negative (FN), and an incorrect positive prediction is called a false positive (FP). The F-score, defined in Equation 1-9 as the harmonic mean of precision and recall, is generally recognized to be a better measure of predictive performance than accuracy on imbalanced datasets (i.e. datasets with unequal class cardinality) [25].

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1-7}$$

$$\text{Recall} = \text{TPR} = \text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1-8}$$

$$\text{F-score} = 2\,\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\ \text{TP}}{2\ \text{TP} + \text{FN} + \text{FP}} \tag{1-9}$$

- Sensitivity and specificity: sensitivity, also known as the true positive rate (TPR) or recall, measures the proportion of positive examples that are correctly identified as such. Sensitivity is defined in Equation 1-8. Specificity, which is defined in Equation 1-11 as one minus the false positive rate (FPR), measures the proportion of negative examples that are correctly identified as such. Maximizing sensitivity implies minimizing the false negative rate, whereas maximizing specificity implies minimizing the false positive rate. In the context of biomarker discovery (see subsection 1-2-2), maximizing sensitivity guarantees the exhaustive identification of all candidate biomarkers, whereas maximizing specificity guarantees the identification of reliable biomarkers [11]. Sensitivity and specificity are better measures of classifier performance than accuracy on imbalanced datasets.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{1-10}$$

$$\text{Specificity} = 1 - \text{FPR} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{1-11}$$

1-2-2 Model Interpretability & Feature Importance

Interpretability is core to the field of explainable artificial intelligence (XAI). Research in XAI is driven by the need to understand the inner workings of complex AI systems and explain their decision-making process to users [26]. Being able to explain and present the inner workings of an AI system in a form that humans can understand is key to guaranteeing user trust and social acceptance, both of which are crucial for the AI system's successful deployment [26]. Interpretable AI systems are easier for developers to debug, tune, and monitor. Since the European Union's General Data Protection Regulation (GDPR) went into effect in 2018, algorithmic explainability and accountability became legal requirements for the deployment of AI systems that involve autonomous decision making processes [27]. The "right to explanation" allows those impacted by the decision of an algorithm to demand an explanation [27]. Since there is no agreed-upon formal technical definition for interpretability in machine learning [28], we proceed to define what we mean by interpretability in our work. We use the terms interpretability and explainability interchangeably.

Model interpretability is the ability of a supervised machine learning model to explain its predictions by explicitly reporting the relative importance of its features. Note that interpretability is a quality of a model, rather than of the machine learning algorithm that is used to learn the machine learning model from data. We define the importance of a feature as the degree to which it influences the model's prediction, considering both its direct effect (i.e. correlation with the prediction) and its indirect effect (i.e. correlation between features). Feature importance, which is also referred to as relevance or predictive power, is measured differently depending on the choice of classifier and interpretability method. Feature ranking refers to the task of ordering the features by their respective importance. Note that feature ranking differs from feature selection in that none of the original features, no matter how irrelevant or redundant, are removed [29]. An irrelevant feature is one that does not contribute to the predictive performance of the classifier under study. A feature is said to be redundant if one or more of the other features are highly correlated with it, and its relevance is reduced by the knowledge of any one of these features [30]. Many of the approaches commonly used in scientific literature to evaluate the relative predictive importance of features fail to correctly account for feature correlation [31]. Yet, it is by studying the inter-dependencies of features that we can accurately identify the critical risk factors and biomarkers of a disease, and hence improve our understanding of the features' biochemical functions and interactions [32]. The issue of feature correlation makes the ranking of IMS features, many of whom are biologically related and therefore highly correlated, a challenging task [33].

We differentiate between global and local interpretability: **global interpretability** methods aim to explain the model's behavior holistically by investigating the relationship between the features and the model's predictions, whereas **local interpretability** methods aim to explain specific decisions [34]. Global interpretability is recommended to improve our understanding of the phenomenon being modeled, whereas local interpretability is recommended to justify the predictions produced by the model for individual instances [35]. We prioritize global interpretability for the analysis of IMS data because our aim is ultimately to discover explanatory principles for how the spatial distribution and relative concentration of certain molecular features relate to the classification of a tissue sample. Interpretability methods can further be categorized as model-specific or model-agnostic [26]: **model-specific interpretability** methods derive explanations by examining the model's structure and parameters, whereas **model-agnostic interpretability** methods derive post-hoc explanations by observing how perturbing the model's inputs affects it output. Model-specific methods owe their computational efficiency to the fact they perform feature ranking in the process of training the model [36]. Unlike model-specific methods, model-agnostic methods are applicable across different types of predictive models. By treating the model as a black-box input-output mapping, model-agnostic approaches provide flexibility both in the choice of model and in the form of explanation [37]. In practical applications, model-agnostic interpretability methods also remove the need for rebuilding a model-specific explanation framework every time the underlying predictive model is replaced [37]. Yet, our main reason for using post-hoc model-agnostic methods is that, by separating explanation from prediction, these methods can achieve interpretability without having to sacrifice performance [37]. Post-hoc model-agnostic interpretability methods effectively provide a solution to the trade-off between predictive performance and interpretability of machine learning models. The **performance versus interpretability trade-off** can be summarized as follows: low-capacity models, such as decision trees or generalized linear models (e.g. linear and logistic regression), tend to be intrinsically transparent, whereas high-capacity models, such as deep neural networks or support vector machines with non-linear kernels, tend to behave like opaque black-boxes [34]. Unlike black-box models, the predictions of intrinsically interpretable models can be easily explained using their respective model-specific explanations. The problem is that, when dealing with high-dimensionality, large-scale complex datasets, like those generated by IMS, black-box high-capacity models often outperform intrinsically interpretable low-capacity models. Restricting the analysis of IMS data to intrinsically interpretable low-capacity models may therefore be limiting, and that is why we resort to post-hoc model-agnostic interpretability methods for explaining the decision-making process of black-box high-capacity high-performance models.

1-3 Research objectives

The aim of this thesis is to propose a machine learning framework for discovering biomarkers in high-dimensional, large-scale imaging mass spectrometry data. A biomarker is an objectively measurable molecular indicator of a specific biological state that may be used to screen for, diagnose, or monitor a disease, to identify new drug targets, or to assess therapeutic response [11]. Research into biomarkers is expected to advance the development of translational medicine, and hence facilitate the transfer of medical or biological advances from the laboratory to the clinic [38]. In healthcare, biomarkers are frequently used to predict and detect the development of a disease and to determine the best course of action for patient care [39]. In the pharmaceutical industry, biomarkers enable the evaluation of whether drugs have sufficient exposure to the site of action, engagement with the drug target, activity in line with the mechanism of action, pharmacological efficacy, clinical efficacy, and potential risks [39]. Biomarkers are also increasingly being used to predict therapeutic outcomes, driving personalized medicine [39]. Biomarker discovery refers to the process by which the differential expression of specific molecules between biological states is first observed [11]. The aim of biomarker discovery is to identify molecular markers that enable the recognition and characterization of a specific biological state, for example diseased versus healthy tissue. Biomarker discovery is the key to empirically identifying the molecular drivers of physiological and pathophysiological processes, and better understanding the effect of pharmacological interventions [40].

Our computational workflow for biomarker discovery encompasses unsupervised and supervised methods: unsupervised machine learning algorithms are used for exploratory analysis and dimensionality reduction, whereas supervised machine learning algorithms are used for classification. Traditionally, the focus has been on maximizing the predictive performance of supervised machine learning models, without necessarily examining the models' internal decision-making processes. Yet, in order to generate insight into the underlying mechanism of disease and drug action, we must go beyond the scope of predictive modeling and extract actionable scientific knowledge from our models. We argue that interpretability is crucial for understanding how the spatial distribution and relative concentration of certain molecular features relate to the labeling of pixels into different classes. We believe that it is crucial for the user to understand why a supervised machine learning model makes a certain decision, and how that model captures underlying biological, pathogenic, or pharmacological mechanisms. In this thesis, we develop a data-driven approach to biomarker discovery based on the identification of highly discriminative features using interpretable supervised machine learning algorithms for classification. The key to biomarker discovery in IMS data is therefore the ranking of features according to their respective discriminative importance: the top-ranking features can then be used as candidate biomarkers worthy of further clinical investigation.

Chapter 2

Methodology: Preprocessing & Dimensionality reduction

The datasets generated by imaging mass spectrometry (IMS) are large and complex. Nowadays, a typical IMS dataset consists of tens of thousands of pixels, each with a mass spectrum encoding thousands of distinct mass-to-charge ratios. As IMS technology development progresses, the attainable mass and spatial resolution, sample surface area, and analyte range keep rising, yielding a drastic increases in the amount of raw data generated. Analyzing IMS data therefore requires dimensionality reduction methods that compress the data, while retaining most of its information, prior to classification. We will be focusing on two matrix factorization methods, namely principal component analysis (PCA) and non-negative matrix factorization (NNMF). These are unsupervised machine learning methods that may also be used for noise removal and exploratory analysis. Dimensionality reduction by matrix factorization is based on the assumption that low-rank structures are embedded in high-dimensional data [41]. Both PCA and NNMF take advantage of the redundancies in the data in order to find a low-rank model describing its underlying structure [41]. The data is expressed as a linear combination of basis vectors, each of which encodes a latent pattern in a lower-dimensional space. We will see that extracting latent trends and patterns from IMS data significantly facilitates human understanding. The essential difference between PCA and NNMF arises from their respective constraints: PCA imposes an orthogonality constraint, whereas NNMF imposes a non-negativity constraint on the matrix factors. Note that, as part of this thesis, both PCA and NNMF were implemented from scratch in MATLAB using different solvers.

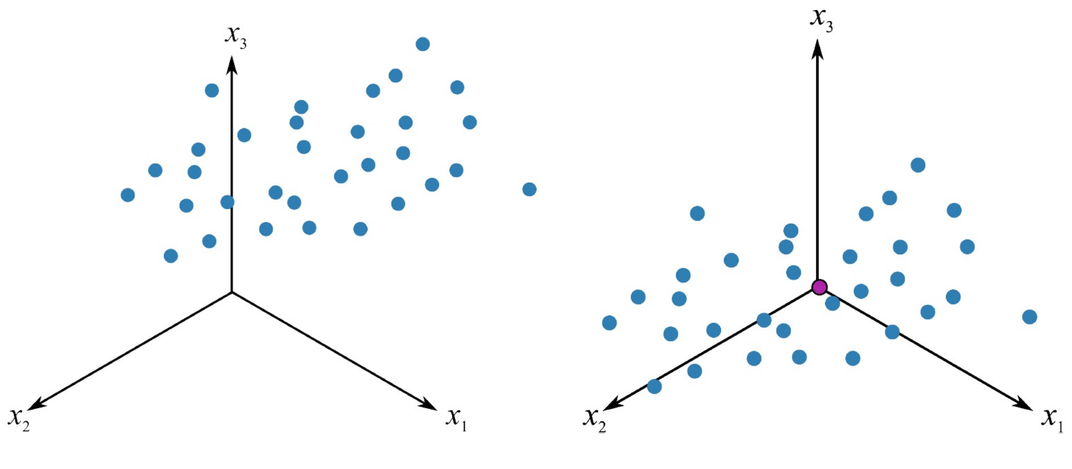
2-1 Principal Component Analysis

Principal component analysis (PCA) is a matrix factorization method for reducing the dimensionality of a dataset while retaining most of its variance [42]. The variance of a dataset is a measure of its statistical dispersion or spread. PCA transforms the original, possibly correlated, variables into a new set of uncorrelated variables, called the principal components (PCs), which are linear combinations of the original variables. The principal components are orthogonal to each other and together form an orthonormal basis onto which is projected the data. PCs are calculated in decreasing order of explained variance: the first PC accounts for the maximum variance contained in the original dataset, while subsequent PCs account for the maximum residual variance [43]. The number of PCs retained is determined by the desired percentage of total variance retained in the transformed dataset [44]. It is the number k of retained PCs that defines the rank of the reduced data.

2-1-1 Introductory Example to Principal Component Analysis

We consider a three-dimensional example dataset plotted in Figure 2-1a. The dataset has $m = 30$ observations and $n = 3$ variables, defining an $(m \times n)$ matrix $\tilde{X}$. Before performing PCA, the data is moved to the center of the coordinate system as in Figure 2-1b. Centering consists in subtracting the column means μ_j from the columns of $\tilde{X}$ so that the columns of the centered data matrix X have zero mean [45]. The observations (i.e. rows) and variables (i.e. columns) of X are respectively written $x_i = X_{(i,:)}$ and $x^j = X_{(:,j)}$ with $i = 1, 2...30$ and $j = 1, 2, 3$. Note that exponent j in notation x^j does not represent a power but rather a feature index.

$$X = \tilde{X} - \mu \qquad \text{with} \qquad \forall j, \mu_j = \frac{1}{m}\sum_{i=1}^{m} \tilde{X}_{(i,j)} \tag{2-1}$$



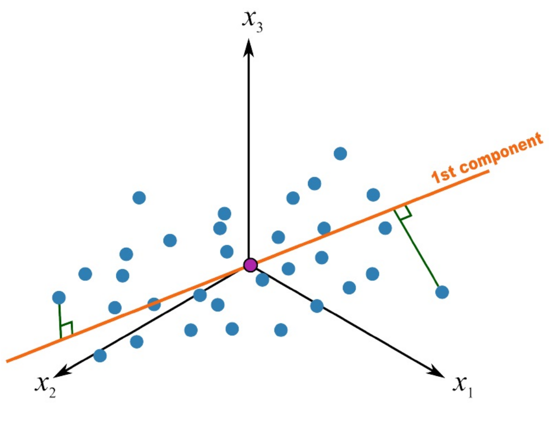
(a) Original data $\tilde{X}$ **(b)** Centred data X

Figure 2-1: Centering the original data prior to principal component analysis. Figure adapted from [5]. Copyright © 2010 to 2019 Kevin G. Dunn - creative commons attribution license.

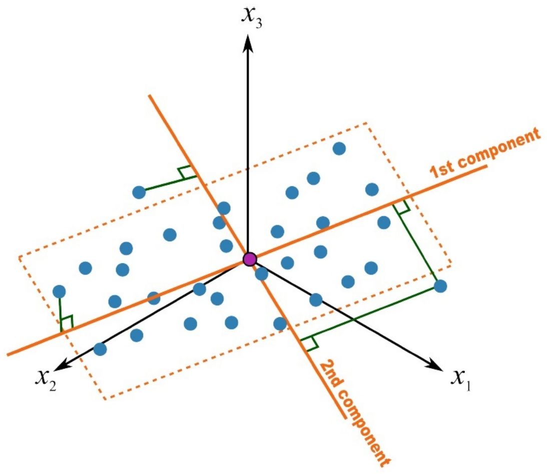
The first principal component is drawn in Figure 2-2a: it's loading is the three-dimensional unit vector v_1 whose direction maximizes the variance of the data point projections. Vector v_1 is a normalized linear combinations of the original variables x^1, x^2 and x^3: $v_1 = \alpha_{1,1}x^1 + \alpha_{2,1}x^2 + \alpha_{3,1}x^3$ where loading coefficients $\alpha_{j,1}$ represent the contribution of x^j to v_1. Since v_1 is oriented primarily in the x^2 direction, $\alpha_{2,1} >> \alpha_{1,1}$ and $\alpha_{2,1} >> \alpha_{3,1}$. The most important variable in understanding the data's overall variability is x^2. Each observation x_i has a score: it is the distance to the origin of the observation's projection on the PC. Maximizing the scores' variance is equivalent to minimizing the error, which is the sum of the residual distances from the original data points to the principal component. The perpendicular distances from two data points to the first PC are indicated by dotted lines in Figure 2-2a. The projections of all 30 points form a score vector z_1 of dimension 30. The score value of the i^{th} observation along the first principal component is a linear combination of the i^{th} row of the data matrix X and the PC's direction vector v_1: $z_{i,1} = x_i^T v_1$ where $i = 1, 2...30$.

The second principal component must be orthogonal to the first principal component as shown in Figure 2-2b. The second PC's loading is the three-dimensional unit vector v_2 pointing in the direction of second largest variance such that $v_2 = \alpha_{1,2}x^1 + \alpha_{2,2}x^2 + \alpha_{3,2}x^3$ where loading coefficients $\alpha_{j,2}$, where $j = 1, 2, 3$, represent the contribution of original variable x^j to v_2. The corresponding score vector z_2 is computed by projecting the data points onto v_2 such that $z_{i,2} = x_i^T v_2$ for $i = 1, 2...30$.

The first and second principal components define a 2D plane within the 3D original coordinate system as shown in Figure 2-2b. These two PCs form a new coordinate system whose axes coincide with the directions of maximum variation of the data. Reducing the data from three to two dimensions by projecting it on a plane exposes the data's latent structure and facilitates visualization.



(a) First principal component



(b) Second principal component

Figure 2-2: First and second principal components. Figure adapted from [5]. Copyright © 2010 to 2019 Kevin G. Dunn - creative commons attribution license.

What follows is the mathematical derivation of the first and second principal components. We consider only one of the 30 observations of X, that is one row $x_i = X_{(i,:)}$. Vector x_i has zero mean and variance-covariance matrix Σ_i whose diagonal terms $\sigma_1^2, \sigma_2^2, \sigma_3^2$ are the variances of

x_i^1, x_i^2, x_i^3 respectively and whose non-diagonal terms are the covariances between x_i^1, x_i^2, x_i^3.

$$x_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ x_{i,3} \end{pmatrix} \qquad v_1 = \begin{pmatrix} v_{1,1} \\ v_{2,1} \\ v_{3,1} \end{pmatrix} \qquad v_2 = \begin{pmatrix} v_{1,2} \\ v_{2,2} \\ v_{3,2} \end{pmatrix} \tag{2-2}$$

$$\Sigma_i = \begin{pmatrix} \sigma_1^2 & \sigma_{1,2} & \sigma_{1,3} \\ \sigma_{2,1} & \sigma_2^2 & \sigma_{2,3} \\ \sigma_{3,1} & \sigma_{3,2} & \sigma_3^2 \end{pmatrix} \tag{2-3}$$

The first PC maximizes the variance of score $z_{i,1}$, which is a linear combination of x_i and the PC's direction vector v_1. The second PC's direction vector v_2 is perpendicular to v_1 and maximizes the variance of score $z_{i,2}$. The scores are given by Equation 2-4, their variance and covariance are defined by Equation 2-5 [42].

$$\begin{aligned} z_{i,1} &= x_i^T v_1 = x_{i,1}v_{1,1} + x_{i,2}v_{2,1} + x_{i,3}v_{3,1} \\ z_{i,2} &= x_i^T v_2 = x_{i,1}v_{1,2} + x_{i,2}v_{2,2} + x_{i,3}v_{3,2} \end{aligned} \tag{2-4}$$

$$\begin{gathered} \text{var}(z_{i,1}) = v_1^T \Sigma_i v_1 \qquad \text{var}(z_{i,2}) = v_2^T \Sigma_i v_2 \\ \text{cov}(z_{i,1}, z_{i,2}) = v_1^T \Sigma_i v_2 = v_2^T \Sigma_i v_1 \end{gathered} \tag{2-5}$$

The loading vector v_1 of the first PC must maximize $\text{var}(z_{i,1})$ subject to the normalization constraint $v_1^T v_1 = 1$. The method of Lagrange multipliers is used to maximize variance subject to a normalization equality constraint as per Equation 2-6 where λ_1 is a Lagrange multiplier [42]. Differentiation with respect to v_1 yields Equation 2-7 where I is the 3×3 identity matrix.

$$\max_{v_1}(v_1^T \Sigma_i v_1) \text{ subject to } v_1^T v_1 = 1 \iff \begin{cases} \nabla(v_1^T \Sigma_i v_1) - \lambda_1 \nabla(v_1^T v_1 - 1) = 0 \\ v_1^T v_1 - 1 = 0 \end{cases} \tag{2-6}$$

$$\nabla(v_1^T \Sigma_i v_1) - \lambda_1 \nabla(v_1^T v_1 - 1) = 0 \iff \Sigma_i v_1 - \lambda_1 v_1 = 0 \iff (\Sigma_i - \lambda_1 I)v_1 = 0 \tag{2-7}$$

Hence λ_1 is an eigenvalue of variance-covariance matrix Σ_i and v_1 is the corresponding eigenvector. Since the variance can be rewritten as $\text{var}(z_{i,1}) = v_1^T \Sigma_i v_1 = v_1^T \lambda_1 v_1 = \lambda_1 v_1^T v_1 = \lambda_1$, maximizing the variance is equivalent to maximizing λ_1 [42]. Therefore λ_1 is the largest eigenvalue of Σ_i and v_1 is the corresponding eigenvector.

The loading vector v_2 of the second PC must maximize $\text{var}(z_{i,2})$ subject to the normalization constraint $v_2^T v_2 = 1$ along with the additional constraint that $z_{i,1}$ and $z_{i,2}$ are uncorrelated such that $\text{cov}(z_{i,1}, z_{i,2}) = 0$. The zero covariance constraint can be rewritten as per Equation 2-8 using the equality $\Sigma_i v_1 = \lambda_1 v_1$ demonstrated in Equation 2-7.

$$\text{cov}(z_{i,1}, z_{i,2}) = 0 \iff v_2^T \Sigma_i v_1 = 0 \iff v_2^T \lambda_1 v_1 = 0 \iff \lambda_1 v_2^T v_1 = 0 \iff v_2^T v_1 = 0 \tag{2-8}$$

The method of Lagrange multipliers is once again used to maximize $\text{var}(z_{i,2})$ subject to two equality constraints as per Equation 2-9 where λ_2 and ϕ are Lagrange multipliers [42].

Differentiation of Equation 2-9 with respect to v_2 yields Equation 2-10.

$$\max_{v_2}(v_2^T\Sigma_i v_2) \text{ subject to } \begin{cases} v_2^T v_2 = 1 \\ v_2^T v_1 = 0 \end{cases} \iff \begin{cases} \nabla(v_2^T\Sigma_i v_2) - \lambda_2\nabla(v_2^T v_2 - 1) - \phi\nabla(v_2^T v_1) = 0 \\ v_2^T v_2 - 1 = 0 \\ v_2^T v_1 = 0 \end{cases} \tag{2-9}$$

$$\nabla(v_2^T\Sigma_i v_2) - \lambda_2\nabla(v_2^T v_2 - 1) - \phi\nabla(v_2^T v_1) = 0 \iff \Sigma_i v_2 - \lambda_2 v_2 - \phi v_1 = 0 \tag{2-10}$$

Multiplying Equation 2-10 on the left by v_1 yields Equation 2-11 hence $\phi = 0$. Therefore Equation 2-10 reduces to Equation 2-12 which demonstrates that λ_2 is also an eigenvalue of Σ_i and v_2 is its corresponding eigenvector. Since $\lambda_2 = v_2^T\Sigma_i v_2$, maximizing the variance is equivalent to maximizing λ_2. Assuming that Σ_i does not have repeated eigenvalues, λ_2 is the second largest eigenvalue of variance-covariance matrix Σ_i.

$$\underbrace{v_1^T\Sigma_i v_2}_{0} - \underbrace{\lambda_2 v_1^T v_2}_{0} - \phi\underbrace{v_1^T v_1}_{1} = 0 \tag{2-11}$$

$$\Sigma_i v_2 - \lambda_2 v_2 = 0 \iff (\Sigma_i - \lambda_2 I)v_2 = 0 \tag{2-12}$$

We demonstrated that the variance accounted for by the j^{th} principal component (PC) is equal to the j^{th} largest eigenvalue λ_j of the variance-covariance matrix Σ_i for $j = 1, 2, 3$. The total variance of observation x_i is the trace of Σ_i, that is the sum of the original variables' variances. This is equal to the sum of the eigenvalues of variance-covariance matrix Σ_i as shown in Equation 2-13.

$$\text{Tr}(\Sigma_i) = \sigma_1^2 + \sigma_2^2 + \sigma_3^2 = \lambda_1 + \lambda_2 + \lambda_3 \tag{2-13}$$

The proportion of total variance Ω_j accounted for by the j^{th} principal component (PC) is the j^{th} eigenvalue divided by the sum of eigenvalues as per Equation 2-14. Consequently, the cumulative percentage of variance P_k explained for by the first k PCs is the sum of the first k eigenvalues divided by the sum of all eigenvalues. In this example, the number of retained PCs is $k = 2$. Equation 2-15 indicates that the first and second PCs explain more than 95% of the total variance. Computing a third PC is therefore not necessary.

$$\Omega_j = \frac{\lambda_j}{\lambda_1 + \lambda_2 + \lambda_3} \tag{2-14}$$

$$P_2 = 100(\Omega_1 + \Omega_2) = 100\frac{\lambda_1 + \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3} \geqslant 95\% \tag{2-15}$$

The dimensionality reduction of vector x_i achieved by PCA can be summarized by the following two equations. Note that since only two out of three PCs are retained, PCA actually computes a low-rank approximation of x_i. Equation 2-16 expresses PC score z_i by an orthonormal linear transformation of x_i. The columns of matrix V are the eigenvectors of variance-covariance matrix Σ_i. Equation 2-17 relates Σ_i to the diagonal eigenvalue matrix Λ_i such that $\lambda_j = \text{var}(z_{i,j})$.

$$z_i \approx V^T x_i \iff \begin{pmatrix} z_{i,1} \\ z_{i,2} \end{pmatrix} \approx \begin{pmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \\ v_{3,1} & v_{3,2} \end{pmatrix}^T \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ x_{i,3} \end{pmatrix} \tag{2-16}$$

$$\Sigma_i V \approx V\Lambda_i \iff \begin{pmatrix} \sigma_1^2 & \sigma_{1,2} & \sigma_{1,3} \\ \sigma_{2,1} & \sigma_2^2 & \sigma_{2,3} \\ \sigma_{3,1} & \sigma_{3,2} & \sigma_3^2 \end{pmatrix} \begin{pmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \\ v_{3,1} & v_{3,2} \end{pmatrix} \approx \begin{pmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \\ v_{3,1} & v_{3,2} \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \tag{2-17}$$

2-1-2 Two Methods for Principal Component Analysis

The imaging mass spectrometry dataset is represented as a $(m \times n)$ matrix X of m observations, and n features, where each observation corresponds to the mass spectrum of a pixel, and each variable corresponds to the mass-to-charge ratio of a molecular ion. It is assumed here that $n < m$, hence the rank of X is equal to the number of variables n. The objective of PCA is to reduce the rank of X from n to $k << n$ such that most of X's variance is accounted for by the k first principal components. We assume that the n columns of X have be centered prior to performing PCA.

There are two approaches to performing principal component analysis of IMS data: PCA by eigen-decomposition of the variance-covariance matrix, which is the original method developed by Pearson in 1901 [42], and PCA by singular value decomposition (SVD) of the data matrix, which is preferred in practice due to its higher computational efficiency and precision [42]. The main advantage of the singular value decomposition (SVD) approach is that it does not require computing the large variance-covariance matrix of data matrix X.

2-1-2-1 Eigen-decomposition of the Variance-covariance Matrix

PCA by eigen-decomposition of the variance-covariance matrix Σ is a generalization of the method explained in the introductory example: the PCs are computed by solving an eigenvalue-eigenvector problem [42]. The $(n \times n)$ variance-covariance matrix of X is calculated as per Equation 2-18. The eigen-decomposition of Σ is given by Equation 2-19 where V is an orthogonal matrix and Λ is a diagonal matrix. The columns of V are the eigenvectors of covariance matrix Σ as well as the principal directions of the data. The columns of V are orthogonal to each other. The diagonal elements of Λ are the eigenvalues of Σ as well as the variances explained by the principal directions.

$$\Sigma = \frac{1}{m-1} X^T X \tag{2-18}$$

$$\Sigma = V \Lambda V^T \tag{2-19}$$

If we were to proceed iteratively as in the introductory example, we would start with the first PC whose score is z_1 and whose loading vector v_1 is chosen to maximize $\text{var}(z_1)$ subject to the normalization constraint $v_1^T v_1 = 1$. Coefficients $z_{i,1} = v_1^T x_i$ are the scores of the i^{th} observation projected onto the first PC, where $i = 1, 2...m$. Geometrically, maximizing the variance of z_1 can be interpreted as maximizing the distance between the projections of the m observations on the first PC. The second PC is computed as follows: its score z_2 has coefficients $z_{i,2} = v_2^T x_i$ for $i = 1, 2...m$ and its direction vector v_2 maximizes $\text{var}(z_2)$ subject to the normalization constraint $v_2^T v_2 = 1$ and to the zero correlation constraint $\text{cov}(z_1, z_2) = 0$. The following PCs are computed similarly.

For $j = 1, 2...n$, the j^{th} PC's direction vector v_j is the eigenvector of Σ with the j^{th} largest eigenvalue λ_j and the variance explained for by this j^{th} PC is equal to $\lambda_j = \Lambda_{j,j}$. The matrix of principal component scores is defined by Equation 2-20. It's size is $(m \times n)$. The coordinates of the i^{th} data point in the new subspace spanned by the PCs are given by z_i, that is the i^{th} row of Z.

$$Z = XV \tag{2-20}$$

2-1-2-2 Singular Value Decomposition of the Data Matrix

The SVD of data matrix X is presented in Equation 2-21: U and V are $(m \times m)$ and $(n \times n)$ orthogonal matrices respectively and S is an $(m \times n)$ diagonal matrix. The diagonal elements of S are the singular values of X in descending order. Since $n < m$, only the n first singular values are positive: the last $m - n$ singular values are zero and can therefore be removed. The corresponding $m - n$ columns of U need not be computed. We obtain the following matrices: $U\ (m \times n)$, $S\ (n \times n)$ and $V\ (n \times n)$.

$$X = USV^T \tag{2-21}$$

The variance-covariance matrix Σ defined in Equation 2-18 can be rewritten as per Equation 2-22 using the SVD of X and the orthogonal matrix property $U^T U = I$. The singular values s_j are related to the eigenvalues λ_j of Σ by Equation 2-23 where $j = 1, 2...n$. The variance explained by principal component j is proportional to the squared singular value s_j [42].

$$\Sigma = \frac{1}{m-1} X^T X = \frac{1}{m-1} VSU^T USV^T = V \frac{S^2}{m-1} V^T \tag{2-22}$$

$$\lambda_j = \frac{s_i^2}{m-1} \tag{2-23}$$

The score matrix Z, defined in Equation 2-20, can also be rewritten using the SVD of X and the orthogonal matrix property $V^T V = I$. Equation 2-24 expresses Z as the product of U and S.

$$Z = XV = USV^T V = US \tag{2-24}$$

2-1-3 Determining the Optimum Number of Principal Components

Once the scores matrix Z and loading matrix V have been computed by either performing the singular value decomposition of X or the eigen-decomposition of Σ, we must determine how many principal components should be retained for the low-rank approximation of X.

The most frequently used method to estimate the number k of necessary principal components consists in determining the percentage of total variation $P^\star$ one desires be retained in the low-rank approximation of the data [42]. The number k of retained PCs is the smallest integer $j = 1, 2...n$ for which $P^\star$ is exceeded [42]. The percentage of variance accounted for by the first k PCs is defined by Equation 2-25, which is a generalization of Equation 2-15.

$$P_k = 100 \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{n} \lambda_j} \geqslant P^\star \tag{2-25}$$

The original data matrix is approximated by the product of Z_k and V_k in Equation 2-26. The reduced scores matrix Z_k, of dimension $(m \times k)$, and the reduced loadings matrix V_k, of dimension $(n \times k)$, are obtained by removing the last $n-k$ columns from Z and V respectively. When applying PCA to an IMS dataset, only the k first PCs encode relevant biochemical information, whereas the last $n-k$ PCs are assumed to be noise. Noise does not provide useful information and can therefore be omitted. Equation 2-26 therefore effectively performs noise removal.

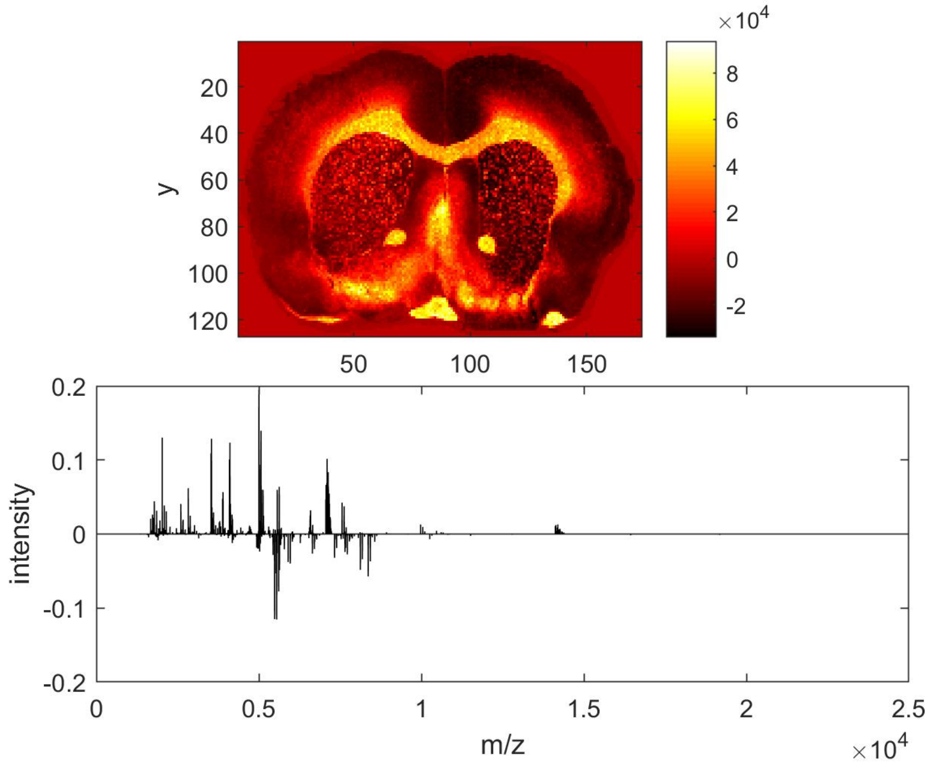
$$X = ZV^T \Rightarrow X \approx Z_k V_k^T \tag{2-26}$$

In the context of IMS data analysis, the reduced scores matrix Z_k and the reduced loadings matrix V_k may be interpreted as follows:

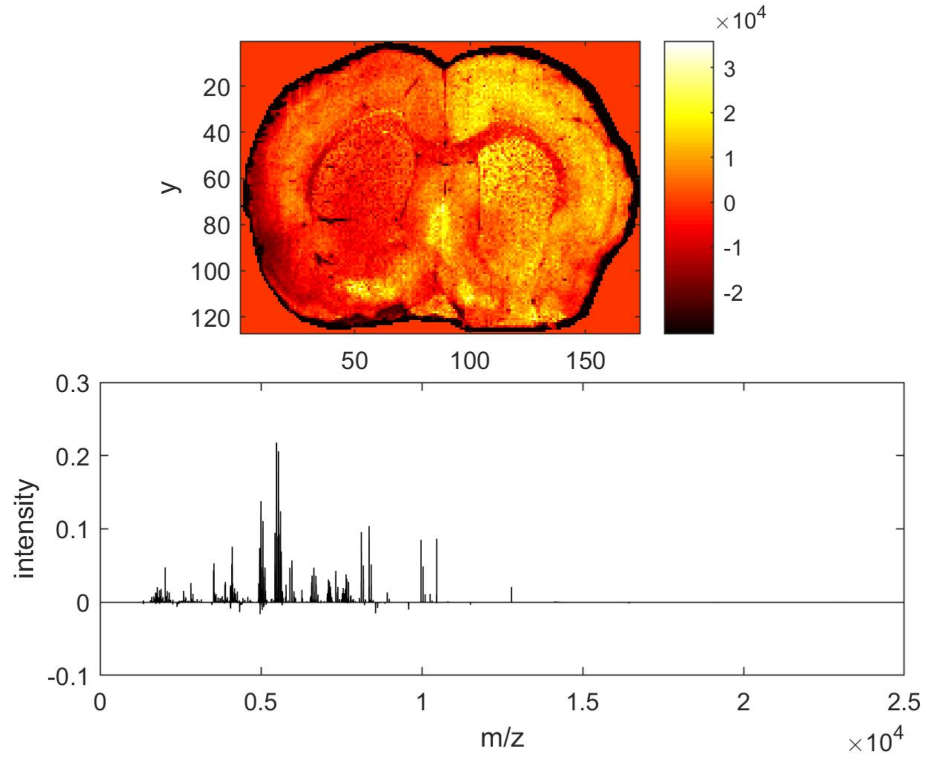
- Matrix Z_k of dimension $(m \times k)$: the scores are the transformed observations. Each of the m observations will have a k-dimensional score vector corresponding to the k retained principal components [17]. The scores are the coordinates of the observations in the new coordinate system formed by the principal components. Geometrically, the scores can be interpreted as the projections of the observations onto the principal components [5]. The i^{th} row of the scores matrix represents the coefficients of the linear combination of principal components describing the mass spectrum of the i^{th} pixel. The larger the (i^{th}, j^{th}) entry of the scores matrix is, the more important the contribution of the j^{th} principal component to the mass spectrum of the i^{th} pixel. The columns of Z_k can also be used to plot score images showing the degree of expression of each principal component throughout the tissue section, which provides useful insight into the data's underlying structure.
- Matrix V_k of dimension $(n \times k)$: the loadings represent the contribution of the original coordinate system to the new coordinate system formed by the principal components [17]. Each of the loadings are n-dimensional unit vectors that span a k-dimensional hyperplane within in the n-dimensional space of original variables [5]. The loadings matrix serves as a mapping from the original coordinate system to the PC's coordinate system. The j^{th} column of the loadings matrix lists the coefficients of the linear combination of original variables describing the j^{th} principal component. The larger the (i^{th}, j^{th}) entry of the loadings matrix is, the more important the contribution of the ion with the i^{th} mass-to-charge ratio to the j^{th} principal component. The j^{th} column of V_k therefore represents the mass spectrum of the j^{th} principal component.

In Figure 2-3, we study IMS data whose sample is the coronal section of a rat-brain. PCA produces PCs that extract latent biochemical patterns from the data. The score images in Figure 2-3 show the degree of expression of the two first PCs across the sample's surface and enable the visualization of different types of cerebral tissue and nuclei, such as the corpus striatum, corpus callosum, and cortex [1]. Note that the score images and pseudo-spectra take on negative values in Figure 2-3, unlike Figure 2-6 that is obtained by non-negative matrix factorization. In Figure 2-4, PCA is performed on another IMS dataset taken from the whole-body section of a mouse-pup. Please refer to section 4-1 for more information regarding these two IMS datasets.

[1]The following anatomical atlas was used to recognize rat brain structures in mass spectrometry images: https://instruct.uwo.ca/anatomy/530/ratpix.pdf

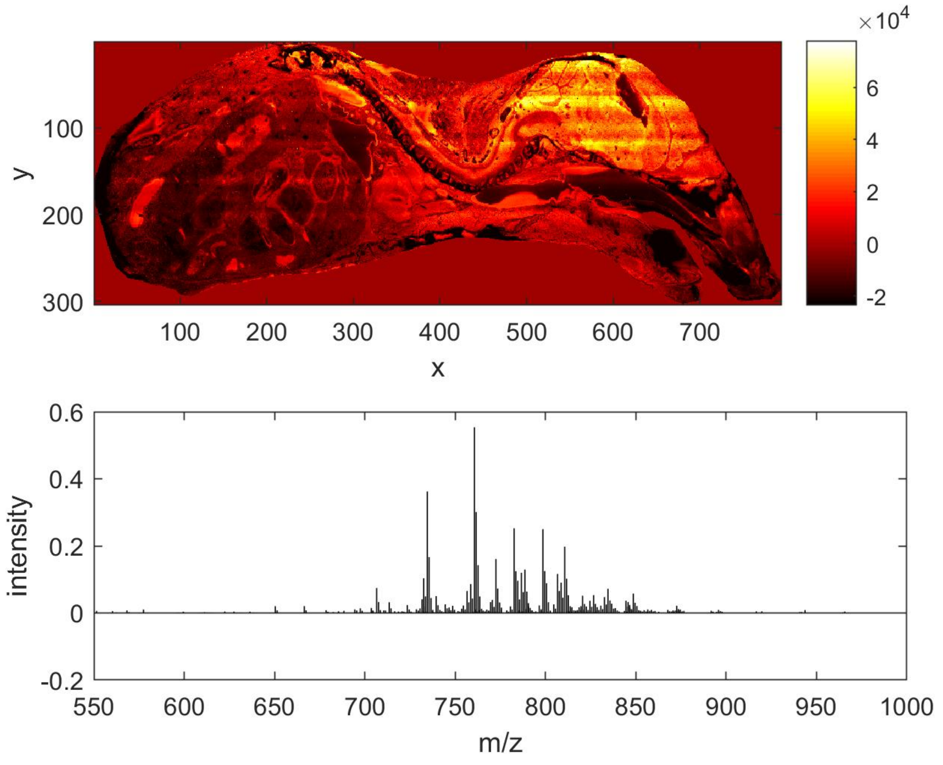


(a) First principal component: score image and pseudo-spectrum

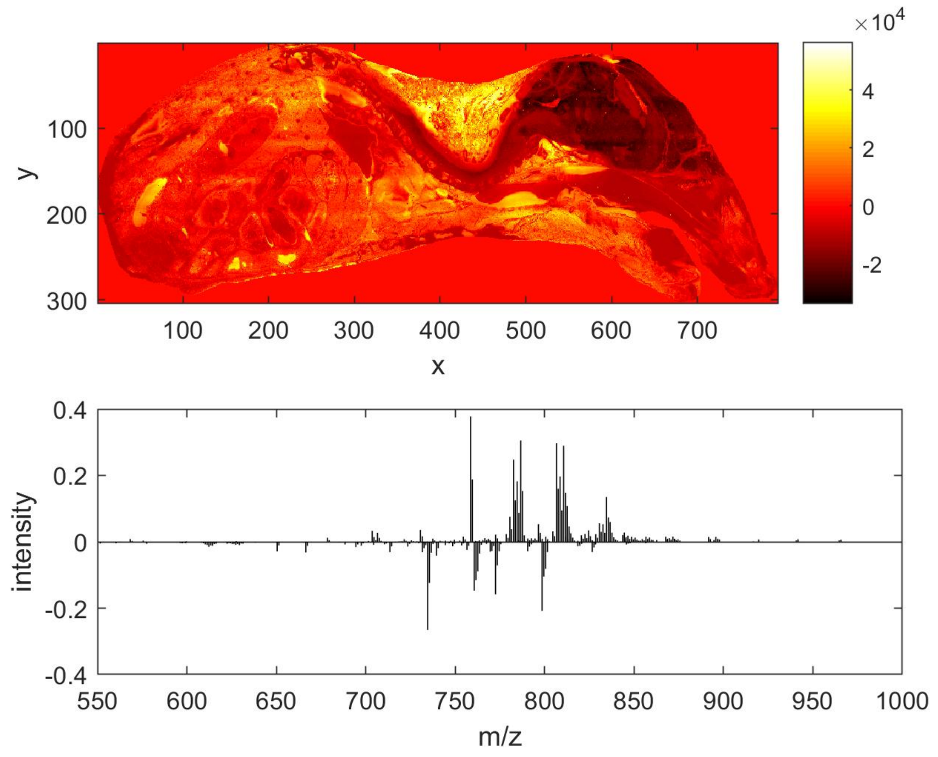


(b) Second principal component: score image and pseudo-spectrum

Figure 2-3: Two first principal components (PCs) obtained by performing principal component analysis (PCA) of an IMS dataset: the score images illustrate the spatial distribution and relative abundance of each PC across the surface of the sample, whereas the loading pseudo-spectra define each PC as a linear combination of the original variables (i.e. m/z bins)



(a) First principal component: score image and pseudo-spectrum



(b) Second principal component: score image and pseudo-spectrum

Figure 2-4: Two first principal components (PCs) obtained by performing principal component analysis (PCA) of another IMS dataset: the score images illustrate the spatial distribution and relative abundance of each PC across the surface of the sample, whereas the loading pseudo-spectra define each PC as a linear combination of the original variables (i.e. m/z bins)

2-2 Non-negative Matrix Factorization

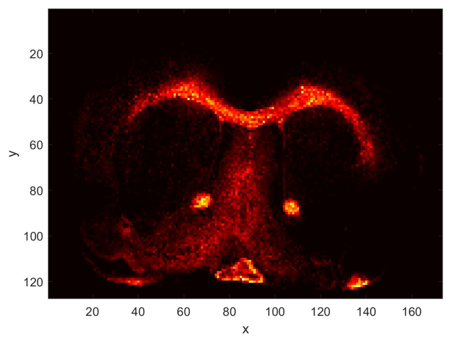
Non-negative matrix factorization (NNMF) is a matrix factorization method that consists of approximating a non-negative matrix X by a lower-rank product of two non-negative matrices W and H. Similarly to principal component analysis (PCA), NNMF expresses the data as a linear combination of basis vectors in a lower-dimensional space. The non-negativity constraint of NNMF is especially useful for analyzing imaging mass spectrometry (IMS) data. Unlike PCA whose score and loading matrices contain both positive and negative terms, NNMF approximates the original data by a lower-rank product of two positive matrices. The original data is expressed as an additive linear combination of latent features [46]. Since imaging mass spectrometry data is inherently non-negative, decomposing IMS data matrices into non-negative factors avoids contradicting physical reality and facilitates human interpretation of the matrix factors.

Unlike PCA whose representation of the data is holistic, NNMF achieves a parts-based representation of the data by expressing it as an additive linear combination of biologically relevant latent features [46]. The importance of a parts-based representation was demonstrated by Lee and Seung who proposed NNMF as an image decomposition method in 1999 [46]. The image of a human face was approximated by a linear superposition of basis images: the basis images obtained by NNMF represented individual features such as the nose and mouth, whereas PCA generated "eigenfaces" without any obvious visual interpretation [46]. In Figure 2-5, we perform NNMF on an IMS dataset whose sample is the coronal section of a rat brain. Figures 2-5a, 2-5b and 2-5c represent specific anatomical regions of the rat's brain, namely the striatum (i.e. caudate and putamen) and the cerebral cortex [2]. Note that in Figure 2-5a, NNMF successfully differentiates the white matter (i.e. nerve tracks including the corpus callosum, anterior commissure, optic chiasma and olfactory nerve tracks) from the rest of the brain tissue. The basis images of Figures 2-5d, 2-5e, and 2-5f represent instrumental noise that NNMF has extracted from the IMS data. In Figure 2-7, we perform NNMF on an IMS dataset whose sample is the whole-body section of a mouse-pup: the basis image of Figure 2-7a highlights the brain and spine, whereas the basis image of 2-7b highlights the intestines and liver. In Figure 2-7, the parts-based representation of the data achieved by NNMF extracts latent features that capture biochemical information relevant to different organs.

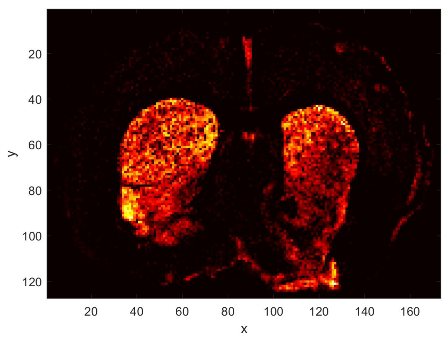
NNMF is a form of additive bi-clustering [47]. Bi-clustering is the simultaneous clustering of variables and observations in order to identify subsets of variables that exhibit similar behavior across subsets of observations and, conversely, subsets of observations that exhibit similar behavior across subsets of variables [48]. Bi-clustering produces a local partitioning of the data by organizing it into sub-matrices, called bi-clusters, whose rows (i.e. observations) and columns (i.e.variables) are highly correlated [48]. Each variable is ascribed a bi-cluster using but a subset of all observations, and each observation is ascribed a bi-cluster using but a subset of all variables. Bi-clustering by NNMF has proven useful for the extraction of localized, biologically relevant patterns in genomic data [49]. NNMF-driven bi-clustering is also a promising method for the exploratory analysis of IMS data because it enables the

[2]The following anatomical atlas was used to recognize rat brain structures in mass spectrometry images: https://instruct.uwo.ca/anatomy/530/ratpix.pdf

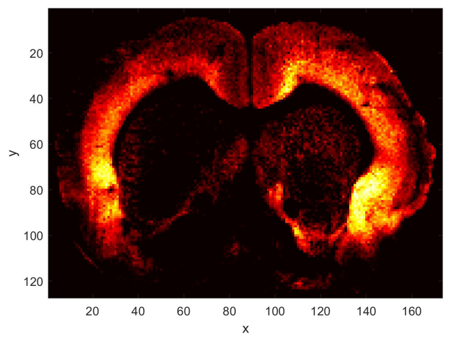
discovery of subsets of biomolecules, characterized by their respective mass-to-charge ratios, that exhibit similar behavior across but a subset of pixels and vice versa [48]. For example, co-regulated biomolecules may only be co-expressed in certain regions of the IMS sample, and the mass spectra of neighboring pixels may only have a small number of biomolecules in common [50].



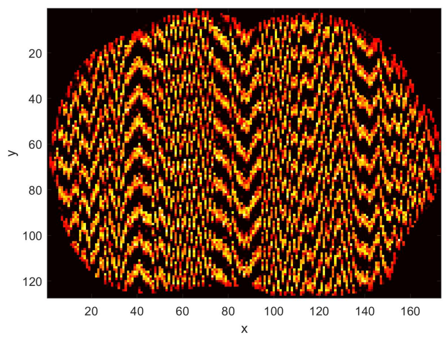
(a) Basis image of nerve tracks



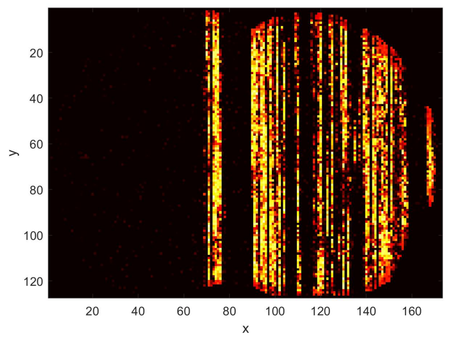
(b) Basis image of the striatum



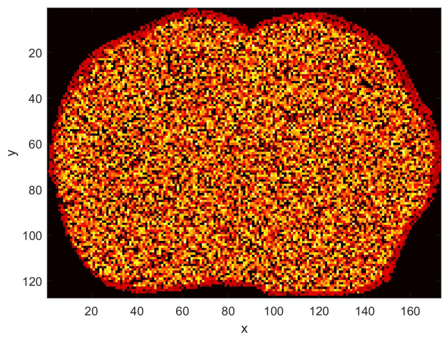
(c) Basis image of the cerebral cortex



(d) Basis image of instrumental noise



(e) Basis image of instrumental noise



(f) Basis image of instrumental noise

Figure 2-5: Parts-based representation of an IMS dataset using non-negative matrix factorization: some latent features extract biochemical information specific to different anatomical structures, whereas other latent features extract noise

2-2-1 Three Methods for Non-negative Matrix Factorization

We consider an $(m \times n)$ IMS data matrix X containing m mass spectra, hence m observations, with n mass-to-charge bins each, hence n variables. The rows of X are the mass spectra measured at each pixel, whereas the columns of X can be reshaped into ion distribution images. In order to reduce the dimensionality of X, NNMF approximates X by the product of two matrices: the basis matrix W, of size $(m \times k)$, and the coefficient matrix H, of size $(k \times n)$ according to Equation 2-27. The target number k of factors, or latent features, is predefined. The non-negativity constraint imposed by NNMF forbids negative entries in the matrix factors. As a result, W and H are positive matrices: $W \geq 0, H \geq 0$. Assuming for X that $n < m$ and $\text{rank}(X) = n$, NNMF generates a matrix product whose rank is at most k such that $1 < k < n$.

$$X \approx WH \tag{2-27}$$

NNMF extracts the biochemical structure of a sample by finding k factors, each defined as a non-negative linear combination of ions, and approximating the mass spectra of each pixel by a non-negative linear combination of factors. In Equation 2-27, matrices W and H serve purposes similar to those of PCA's scores matrix Z_k and loadings matrix V_k respectively. Furthermore, if we were to look at NNMF through the lens of bi-clustering, matrices W and H would perform the clustering of observations and variables respectively. Figure 2-6 displays the basis images (i.e. columns of W) and pseudo-spectra (i.e. rows of H) of two latent features obtained by NNMF on an IMS dataset whose sample is the coronal section of a rat brain. In this section about non-negative matrix factorization, we consider indices $i = 1, 2...m$ for the observations (i.e. rows of X), $j = 1, 2...k$ for the latent features (i.e. columns of W and rows of H) and $p = 1, 2...n$ for the variables (i.e. columns of X).

- Matrix W of dimension $(m \times k)$: The rows of W are the projections of the observations onto the subspace spanned by the k basis vectors. Each of the m observations is expressed as a linear additive combination of these factors. The i^{th} row of W lists the non-negative coefficients of the linear combination of basis vectors describing the mass spectrum of the i^{th} pixel. The columns of W are the vectorized basis images of the data: the j^{th} column of W can be reshaped into a basis image showing the j^{th} factor's spatial distribution and degree of expression throughout the sample.
- Matrix H of dimension $(k \times n)$: Multiplying by H performs a mapping from the original coordinate to the new coordinate system. The rows of H express the factors, or basis vectors, as a non-negative linear combination of the n original variables. The j^{th} row of H is actually the characteristic mass spectra of the j^{th} factor.

Non-negative matrix factorization starts by constructing random non-negative initial estimates for matrix factors W and H. The W and H matrices are iteratively updated to minimize the cost function defined in Equation 2-28 as the mean-squared-error or squared Euclidean distance between X and WH [51]. Note that the root-mean-squared-error could also be used as cost function.

$$f(W, H) = \frac{1}{2}||X - WH||_F^2 \tag{2-28}$$

The Frobenius norm (i.e. Euclidean norm) of matrix $(X - WH)$ is defined in Equation 2-29 as the square root of the trace of $(X - WH)(X - WH)^T$, or equivalently as the square root of the sum of its squared elements.

$$||X - WH||_F = \sqrt{\operatorname{Tr}\left((X - WH)(X - WH)^T\right)} = \sqrt{\sum_{i=1}^{m}\sum_{p=1}^{n}(X - WH)^2} \tag{2-29}$$

Solving the non-negative matrix factorization problem can be formulated as the following optimization problem [51]: cost function $f(W, H)$ is minimized subject to non-negative constraints.

$$\min_{W,H} \frac{1}{2}||X - WH||_F^2 \text{ subject to } W \geq 0, H \geq 0 \tag{2-30}$$

The cost function $f(W, H)$ of Equation 2-28 is convex with respect to W or H, but it is not convex with respect to both W and H. The existence of local minima and the lack of a unique solution are the two most important challenges affecting the optimization problem stated in Equation 2-30 [52]. The problem is therefore divided into two convex non-negative least-squares problems which are solved by keeping one factor fixed while updating the other, alternating and iterating until convergence [41]. Although there are many different numerical approaches to solving the NNMF optimization problem of Equation 2-30, all the NNMF methods we use adhere to the same general framework: start by randomly initializing factors $W^{(0)}$ and $H^{(0)}$, then at each iteration t, fix one of the two factors and update the other in such a way that the objective function is reduced [53].

- Fix $H^{(t)}$ and find $W^{(t+1)} \geq 0$ such that $||X - W^{(t+1)}H^{(t)}||_F^2 < ||X - W^{(t)}H^{(t)}||_F^2$
- Fix $W^{(t+1)}$ and find $H^{(t+1)} \geq 0$ such that $||X - W^{(t+1)}H^{(t+1)}||_F^2 < ||X - W^{(t+1)}H^{(t)}||_F^2$

Two of the NNMF methods studied hereafter, the multiplicative update algorithm [46] and the alternating least squares (ALS) algorithm [52] are frequently used standard methods for computing NNMF that can be found in MATLAB's Statistics and Machine Learning Toolbox. The hierarchical alternating least squares (HALS) algorithm is more efficient and returns a more precise low-rank approximation [54].

2-2-1-1 Multiplicative Update Algorithm

Non-negative matrix factorization was approached by Lee and Seung using the multiplicative update algorithm [46]. Minimization of the objective function $f(W, H)$ defined in Equation 2-28 is done through the alternative update of W and H. First W and H must randomly be initialized and the columns of W must be scaled to unit norm. What follows is one iteration of the multiplicative update algorithm to be repeated until convergence:

1. Update matrix H with $\epsilon << 1$:

$$H \leftarrow H\frac{(W^TX)}{(W^TWH) + \epsilon}$$

2. Update matrix W with $\epsilon << 1$:

$$W \leftarrow W \frac{(XH^T)}{(WHH^T) + \epsilon}$$

3. Normalize the columns of W

The multiplicative update algorithm is not guaranteed to converge to a local minimum, but only to a stationary point where the cost function ceases to decrease [53]. It is also very sensitive to its initial conditions and should therefore be run multiple times, with different initial conditions, until the solution is satisfactory [52]. Another disadvantage of the multiplicative update algorithm is it does not have the flexibility to escape from a poor minimization path: once an element in W or H becomes zero, it will remain zero [52].

2-2-1-2 Alternating Least Squares Algorithm

The alternating least squares (ALS) algorithm exploits the objective function's convexity in either W or H (not both) to solve the optimization problem in Equation 2-30. The ALS algorithm was developed by Paatero and Tapper in 1994 [52]. The matrix factors W and H must be randomly initialized before implementing ALS. W and H are then alternatively updated by solving unconstrained least-squares problems. After each update of W or H, non-negativity is insured by setting the negative elements obtained from the least squares solution to zero. What follows is one iteration of ALS:

1. Solve for H in $W^TWH = W^TX$ while fixing W
2. Set negative elements of H to 0
3. Solve for W in $HH^TW^T = HX^T$ while fixing H
4. Set negative elements of W to 0

The ALS algorithm converges faster and more consistently than the multiplicative update algorithm [52]. It is however not guaranteed to converge to a global optimum [54]. Unlike the multiplicative update algorithm, the ALS algorithm can recover from a poor minimization path [52]: if an element of W or H becomes zero, it does not necessarily have to remain zero.

2-2-1-3 Hierarchical Alternating Least Squares

The hierarchical alternating least squares (HALS) algorithm was developed by Cichocki and Phan in 2009 [54] in order to improve the convergence rate of the alternating least squares (ALS) algorithm and reduce its computational complexity. The HALS method is robust to noise and suitable for large scale problems [54].

The idea of hierarchical alternating least squares is to decompose the NNMF optimization problem of Equation 2-30 using "local learning rules" to sequentially process blocks of data. In order to approximate X by matrix product WH of reduced rank k, factors W and H are updated by fixing all the variables except for the block comprised of the j^{th} column of W, written $W_{(:,j)}$, and j^{th} row of H, written $H_{(j,:)}$ [41]. HALS minimizes the cost function of Equation 2-28 with respect to the remaining $k-1$ blocks [41]. We obtain the following local optimization problem where $R^{(j)}$ is the j^{th} residual, for $j = 1, 2...k$.

$$\min_{W_{(:,j)},H_{(j,:)}} f_{\text{local}}^{(j)} = ||R^{(j)} - W_{(:,j)}H_{(j,:)}||_F^2 \tag{2-31}$$

$$R^{(j)} = X - \sum_{i\neq j}^{k} W_{(:,j)}H_{(j,:)} \tag{2-32}$$

The local cost function $f_{\text{local}}^{(j)}$ defined in Equation 2-31 is expanded in Equation 2-33 and its gradient with respect to the unknown vectors $W(:,j)$ and $H(j,:)$ is computed in Equations 2-34 and 2-35 respectively [41].

$$\begin{aligned} f_{\text{local}}^{(j)} &= \text{Tr}\left(R^{(j)T}R^{(j)} - 2R^{(j)T}W_{(:,j)}H_{(j,:)} + H_{(j,:)}^T W_{(:,j)}^T W_{(:,j)}H_{(j,:)}\right) \\ f_{\text{local}}^{(j)} &= ||R^{(j)}||_F^2 - 2\,\text{Tr}\left(R^{(j)T}W_{(:,j)}H_{(j,:)}\right) + \text{Tr}\left(H_{(j,:)}^T W_{(:,j)}^T W_{(:,j)}H_{(j,:)}\right) \end{aligned} \tag{2-33}$$

$$\frac{\partial f_{\text{local}}^{(j)}}{\partial W_{(:,j)}} = W_{(:,j)}H_{(j,:)}H_{(j,:)}^T - R^{(j)}H_{(j,:)}^T \tag{2-34}$$

$$\frac{\partial f_{\text{local}}^{(j)}}{\partial H_{(j,:)}} = H_{(j,:)}^T W_{(:,j)}^T W_{(:,j)} - R^{(j)T}W_{(:,j)} \tag{2-35}$$

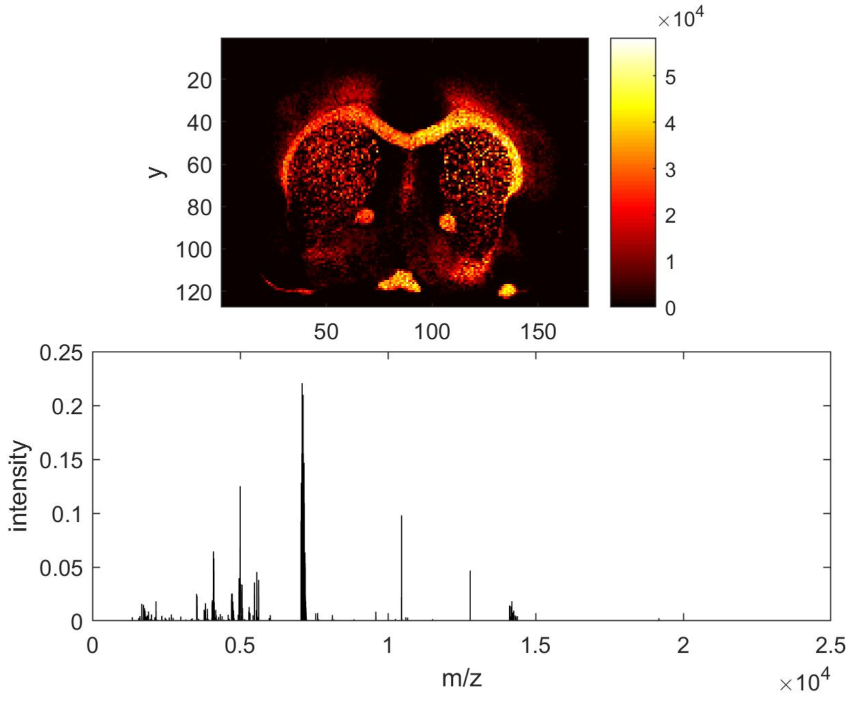
The sequential update rules for the j^{th} components of W and H are obtained by setting Equations 2-34 and 2-35 to zero and enforcing non-negativity as per Equation 2-36 [41]:

$$\begin{aligned} W_{(:,j)}^+ &= \max\left(\frac{R^{(j)}H_{(j,:)}^T}{H_{(j,:)}H_{(j,:)}^T}, 0\right) \\ H_{(j,:)}^+ &= \max\left(\frac{R^{(j)T}W_{(:,j)}}{W_{(:,j)}^T W_{(:,j)}}, 0\right) \end{aligned} \tag{2-36}$$

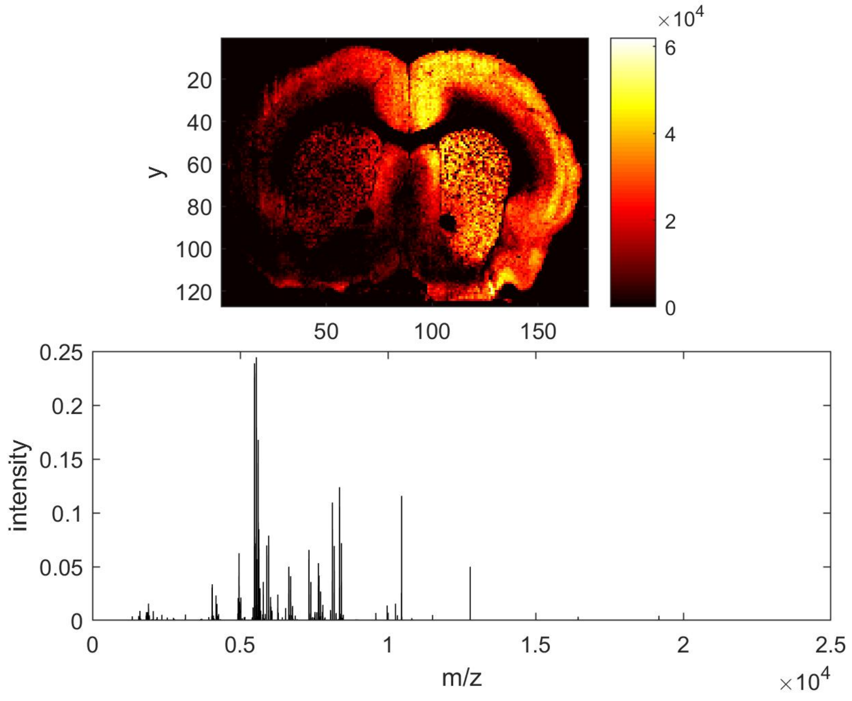
Note that by rewriting the residual of 2-32 as per Equation 2-37, the update rules of 2-36 can be simplified. The simplified update rules, stated in Equation 2-38, avoid having to explicit compute of the residual $R^{(j)}$ at each iteration [41]. In practice, it is recommended to normalize vectors $W_{(:,j)}$ and $H_{(j,:)}$ at the end of each iteration [54].

$$R^{(j)} = X - WH + W_{(:,j)}H_{(j,:)} \tag{2-37}$$

$$\begin{aligned} W_{(:,j)}^+ &= \max\left(W_{(:,j)} + \frac{[XH^T]_{(:,j)} - W[HH^T]_{(:,j)}}{[HH^T]_{(j,j)}}, 0\right) \\ H_{(j,:)}^+ &= \max\left(H_{(j,:)} + \frac{[X^TW]_{(:,j)} - H^T[W^TW]_{(:,j)}}{[W^TW]_{(j,j)}}, 0\right) \end{aligned} \tag{2-38}$$

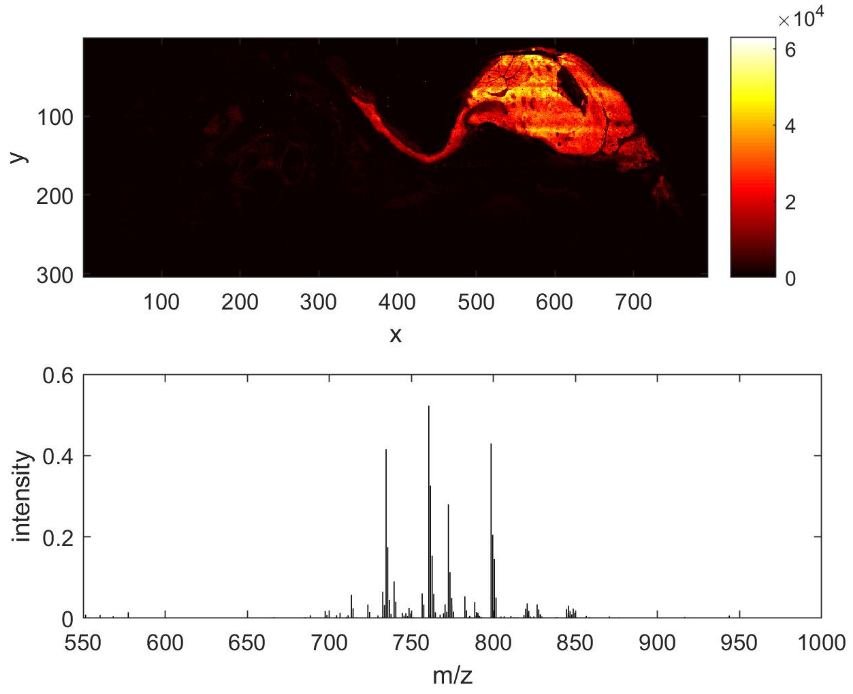


(a) First NNMF latent feature: basis image and pseudo-spectrum

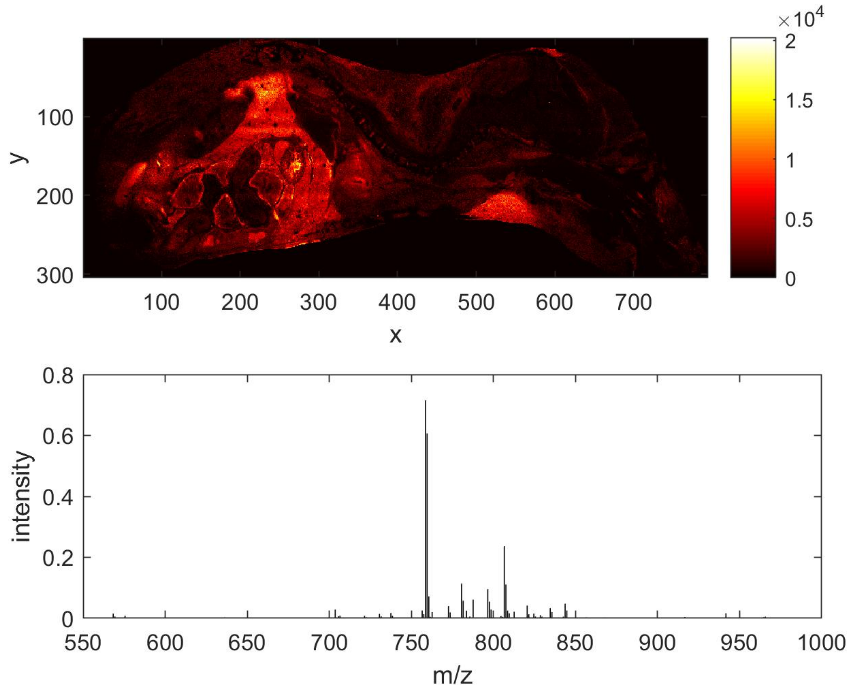


(b) First NNMF latent feature: basis image and pseudo-spectrum

Figure 2-6: Two latent features obtained by performing non-negative matrix factorization of an IMS dataset: the basis images illustrate the spatial distribution and relative abundance of each latent feature across the surface of the sample, whereas the pseudo-spectra define each latent feature as a linear combination of the original variables (i.e. m/z bins)



(a) First NNMF latent feature: basis image and pseudo-spectrum



(b) First NNMF latent feature: basis image and pseudo-spectrum

Figure 2-7: Two latent features obtained by performing non-negative matrix factorization of another IMS dataset: the basis images illustrate the spatial distribution and relative abundance of each latent feature across the surface of the sample, whereas the pseudo-spectra define each latent feature as a linear combination of the original variables (i.e. m/z bins)

2-2-2 Determining the Optimum Number of Latent Features

Choosing the number of factors k prior to performing non-negative matrix factorization (NNMF) determines the rank of data matrix X's approximation. The importance of determining k so as to maximize dimensionality reduction while minimizing loss of information has been thoroughly examined in scientific literature, and yet there is no blueprint on how to do so. The methods we use for optimizing the choice of k make use of NNMF's bi-clustering properties. Given the non-negative matrix factorization $X \approx WH$, we can choose to either cluster the m observations (i.e. pixels) or the n variables (i.e. mass-to-charge ratio bins) of X into k latent features. We consider indices $i = 1, 2...m$ for the observations (i.e. rows of X), $j = 1, 2...k$ for the latent features (i.e. columns of W and rows of H), and $p = 1, 2...n$ for the variables (i.e. columns of X).

- In order to cluster observations, we look at the rows of basis matrix W: observation $x_i = X_{(i,:)}$ is assigned to latent feature number j if max $\left(W_{(i,:)}\right) = W_{(i,j)}$.
- In order to cluster variables, we look at the columns of coefficient matrix H: variable $x^p = X_{(:,p)}$ is assigned to latent feature number j if max $\left(H_{(:,p)}\right) = H_{(j,p)}$.

For the sake of simplicity, we hereafter assume that we use NNMF to cluster observations, rather than variables. Each observation x_i is assigned to exclusively one cluster (i.e. one latent feature). The distance in n-dimensional feature space between two observations x_i and x_j, with $i \neq j$, is a measure of the biochemical dissimilarity between the corresponding pixels of the IMS sample. We choose the squared Euclidean distance as our preferred distance metric.

2-2-2-1 Silhouette Analysis

Silhouette analysis may be used for determining the optimal number k of latent features for non-negative matrix factorization by assessing the quality of the resulting data partition. Clustering quality is quantified by comparing cluster compactness to cluster separation [55]. An observation is considered to be well clustered if it is close to the other observations in the same cluster and far from observations in other clusters. Ideally, within-cluster dissimilarities should be small and between-cluster dissimilarities should be large. For each observation $x_i = X_{(i,:)}$, for $i = 1, 2...m$, the silhouette index s_i is computed as follows [55]:

1. For each observation x_i, calculate the average dissimilarity α_i between x_i and all other observations in the cluster $\mathcal{C}_p$ to which x_i belongs. Note that $\#\mathcal{C}_p = \text{card}(\mathcal{C}_p)$.
$$\alpha_i = \frac{\sum_{l=1}^{\#\mathcal{C}_p} ||x_i - x_l||^2}{\#\mathcal{C}_p} \tag{2-39}$$
2. For all other clusters $\mathcal{C}_j$ to which x_i does not belong, calculate the average dissimilarity $d_{i,j}$ of x_i to all observations of $\mathcal{C}_j$. Equation 2-40 is reiterated for $j = 1, 2...p-1, p+1...k$ as a result of which a $(k-1)$-dimensional vector D_i is obtained. Vector D_i's j^{th} entry is the average dissimilarity of x_i to the elements of cluster $\mathcal{C}_j$ where $j \neq p$.
$$d_{i,j} = \frac{\sum_{l=1}^{\#\mathcal{C}_j} ||x_i - x_l||^2}{\#\mathcal{C}_j} \tag{2-40}$$

3. Define β_i as the dissimilarity between x_i and its neighbor cluster, that is the closest cluster to which x_i does not belong. The neighbor cluster is the second most similar cluster to observation x_i, after cluster $\mathcal{C}_p$.

$$\beta_i = \min_{\forall \mathcal{C}_j \neq \mathcal{C}_p}(D_i) \tag{2-41}$$

4. The silhouette index s_i of observation x_i is defined by Equation 2-42 [55]:

$$s_i = \frac{\beta_i - \alpha_i}{\max(\alpha_i, \beta_i)} \iff \begin{cases} 1 - \frac{\alpha_i}{\beta_i} & \text{if} \quad \alpha_i < \beta_i \\ 0 \quad \text{if} \quad \alpha_i = \beta_i \\ \frac{\beta_i}{\alpha_i} - 1 & \text{if} \quad \alpha_i > \beta_i \end{cases} \tag{2-42}$$

The silhouette index s_i ranges from -1 to 1 and can be interpreted as follows [55]:

- If $s_i \approx 1$, the within-cluster dissimilarity α_i is much smaller than the minimum between-cluster dissimilarity β_i. So observation x_i has been assigned to the right cluster.
- If $s_i \approx 0$, observation x_i lies equally far from two clusters so it is unclear to which cluster x_i should be assigned.
- If $s_i \approx -1$, $\beta_i >> \alpha_i$ implies that x_i is closer to a different cluster than $\mathcal{C}_p$. Observation x_i has therefore been assigned to the wrong cluster.

Given a targeted reduced rank k, the cluster mean silhouette of a cluster $\mathcal{C}_j$, for $j = 1, 2...k$, is defined in Equation 2-43 as the average of the s_i silhouette indices of all observations belonging to $\mathcal{C}_j$. $S_j(k)$ is effectively the ratio of the compactness of cluster $\mathcal{C}_j$ to its separation. If k is too low, some of the data's natural clusters will be artificially combined in order to divide the data into k groups. The artificially combined clusters will have large within-cluster dissimilarities and $S_j(k)$ will be reduced as a consequence. On the other hand, if k is too high, some of natural clusters will be artificially partitioned in order to conform to the specified number of clusters k. Since these artificially partitioned clusters resemble their natural cluster, their between-cluster dissimilarity will be small, which also results in a reduced $S_j(k)$.

$$S_j(k) = \frac{1}{\#\mathcal{C}_j} \sum_{i \in \mathcal{C}_j} s_i \tag{2-43}$$

Finally, the global silhouette index $\bar{S}(k)$ is the average of the mean silhouettes of all clusters, or, equivalently, the average of the s_i silhouette indices of all m observations in the dataset [55]. The global silhouette index, defined in Equation 2-44, verifies $-1 \leq \bar{S}(k) \leq 1$. The closer the average silhouette is to 1, the more stable the clustering and the better the choice of k [55].

$$\bar{S}(k) = \frac{1}{k} \sum_{j=1}^{k} S_j(k) = \frac{1}{m} \sum_{i=1}^{m} s_i \tag{2-44}$$

2-2-2-2 Consensus Clustering

Consensus clustering may be used to determine the optimal number of latent features k to use for non-negative matrix factorization by assessing cluster stability [49]. For a given value of k, we compute how frequently different observations are grouped together (i.e. assigned to the same latent feature) in repeated runs of NNMF. The resulting pairwise consensus rate is used to measure the robustness of NNMF to its random initialization. If the number k of clusters reflects the inherent structure of the data, the assignment of observations (i.e. pixels) to latent features should vary little from run to run [49]. The more stable the clusters are with respect to random initialization, the more reliable the resulting partition and the better the choice of k [49]. The advantage of consensus clustering is that, unlike silhouette analysis, it exploits the stochastic nature of NNMF driven bi-clustering.

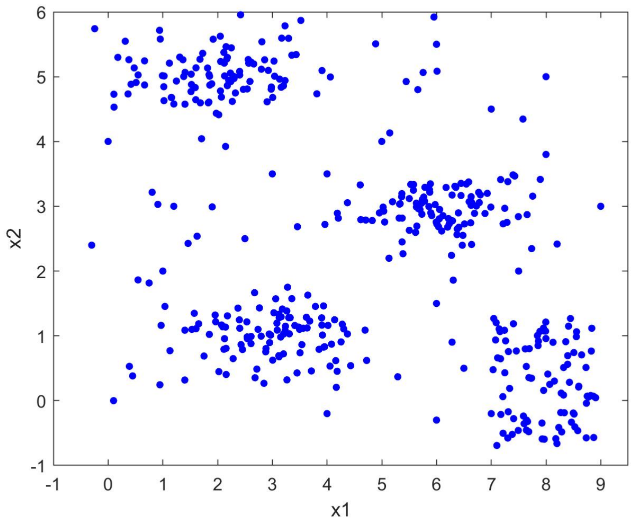
For each run of NNMF, the pixel assignment is defined by a connectivity matrix C of size $m \times m$: $C_{i,j} = 1$ if pixels i and j belong to the same cluster, and $C_{i,j} = 0$ if they belong to different clusters [49]. The agreement among several clustering runs is represented using a consensus matrix: the consensus matrix $\bar{C}$ stores, for each pair of observations, the proportion of clustering runs in which two observations are clustered together [56]. It is computed by taking the average over the connectivity matrices of each run of NNMF. The entries of $\bar{C}$ range from 0 to 1 and reflect the probability that two observations are grouped together: $\bar{C}_{i,j} = 1$ if observations x_i and x_j are always assigned to the same latent feature, $\bar{C}_{i,j} = 0$ if x_i and x_j are never assigned to the same latent feature. If k is well chosen, C should vary little from run to run and the entries of $\bar{C}$ will be close to either 0 or 1 [49]. The two following consensus clustering metrics may be used for determining the optimal value of k for NNMF:

- Dispersion coefficient [57]: the consensus matrix's dispersion is a measure of the reproducibility of the cluster assignment for a given value of k. The dispersion coefficient ρ of the consensus matrix $\bar{C}$ is defined by Equation 2-45. The number of latent features k should be chosen to maximize ρ.

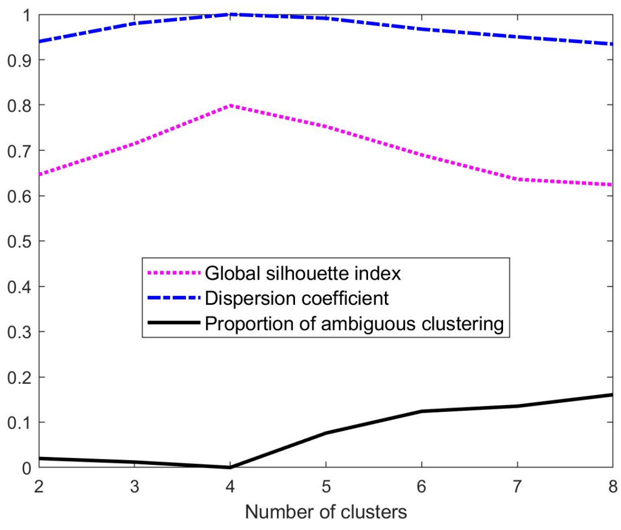
$$\rho(k) = \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} 4 \left(\bar{C}_{i,j} - \frac{1}{2} \right)^2 \tag{2-45}$$

- Proportion of ambiguous clustering [58]: the proportion of ambiguous clustering is defined as the fraction of sample pairs whose consensus coefficients fall between 0 and 1. The number of latent features k should be chosen to minimize the proportion of ambiguous clustering. Ideally, all entries of consensus matrix $\bar{C}$ are either equal to 0 or 1 and the proportion of ambiguous clustering is 0.

Figure 2-8 shows that, for the toy dataset of Figure 2-8a that is obviously made up of four latent clusters, the three clustering measures agree: the global silhouette index and the dispersion coefficient are maximum for $k = 4$, whereas the proportion of ambiguous clustering is minimum for $k = 4$.



(a) Scatter plot of a 2-D dataset whose number of natural latent clusters is four



(b) Plot of the global silhouette index, dispersion coefficient, and proportion of ambiguous clustering versus the number of clusters

Figure 2-8: Toy clustering example: grouping the data into four clusters, which is the natural number of clusters inherent to the data, maximizes the global silhouette index, and the dispersion coefficient, and minimizes the proportion of ambiguous clustering.

2-3 Whitening

Statistical whitening, or sphering, is a common preprocessing step for linearly decorrelating variables by imposing an identity covariance matrix to a zero-centered dataset [21]. We have an m-by-n imaging mass spectrometry dataset X whose observations and variables are written $x_i = X_{(i,:)}$ and $x^j = X_{(:,j)}$ for $i = 1, 2...m$ and $j = 1, 2...n$ respectively. We assume that centering has been performed as per Equation 2-1 prior to whitening. The covariance matrix Σ of X is related to covariance matrix P and diagonal variance matrix V by Equation 2-46. We have $V = \mathrm{diag}(\sigma_1^2, \sigma_2^2...\sigma_n^2)$ such that $\sigma_j^2 = \mathrm{var}(x^j)$ for $j = 1, 2...n$. The eigendecomposition of Σ and P are presented in Equation 2-47: U and G are the eigenvectors of Σ and P respectively; Λ and Θ are their respective eigenvalues.

$$\Sigma = V^{1/2}\, P\, V^{1/2} \tag{2-46}$$

$$\Sigma = U\, \Lambda\, U^T \qquad\qquad P = G\, \Theta\, G^T \tag{2-47}$$

Whitening transforms the original variables x^j into new orthogonal variables z^j using the n-by-n whitening matrix W [59]. Equation 2-48 defined the linear transformation through which whitening is achieved. The original covariance $\mathrm{cov}(X) = \Sigma$ becomes the identity matrix: $\mathrm{cov}(Z) = I$. The new sphered variables z^j are centered, such that $\mathrm{E}[z^j] = 0$, and scaled to have unit variance, such that $\mathrm{var}(z^j) = 1$ for $j = 1, 2...n$.

$$Z = XW \tag{2-48}$$

The whitening transformation of Equation 2-48 requires choosing of a whitening matrix W that makes $\mathrm{cov}(Z) = I$. Equation 2-49 states the condition that W must satisfy [60].

$$\mathrm{cov}(Z) = I \iff \mathrm{E}[ZZ^T] = I \iff \mathrm{E}[XWW^TX^T] = I \iff W\Sigma W^T = I \iff W^TW = \Sigma^{-1} \tag{2-49}$$

Due to rotational freedom, W is not uniquely determined by Equation 2-49. Rewriting W in polar form demonstrates that whitening combines multivariate rescaling by $\Sigma^{-1/2}$ with a rotation by Q_1 [59]. Q_1 is an orthogonal matrix such that $Q_1^TQ_1 = I$. So W will satisfy Equation 2-49 regardless of the choice of Q_1 [59]. There are therefore infinitely many possible whitening procedures: each procedure uses a different whitening matrix to linearly decorrelate the variables.

$$W = Q_1\, \Sigma^{-1/2} \tag{2-50}$$

Depending on the application, it may be necessary to start by standardizing the original variables by multiplying by $V^{-1/2}$. Whitening matrix W can be rewritten as per Equation 2-51 where Q_2 is another orthogonal rotation matrix.

$$W = Q_2\, P^{-1/2}\, V^{-1/2} \tag{2-51}$$

We discuss five commonly used statistical whitening procedures based either on the zero-phase component analysis (ZCA), principal component analysis (PCA), or Cholesky decomposition of X. Our five whitening methods arise from specific constraints on the cross-covariance Φ and cross-correlation Ψ between the sphered data Z and the original data X [59]. Equation

2-52 demonstrates that the cross-covariance Φ and cross-correlation Ψ are related to rotation matrices Q_1 and Q_2 of Equation 2-50 and 2-51 respectively [59].

$$\Phi = \mathrm{cov}(Z, X) = W\,\Sigma = Q_1\,\Sigma^{1/2} \qquad \Psi = \mathrm{cor}(Z, X) = \Phi\,V^{-1/2} = Q_2\,P^{1/2} \tag{2-52}$$

Zero-phase component analysis (ZCA) is an image processing technique introduced by Bell and Sejnowski in 1997 [60]. ZCA whitening, also known as Mahalanobis whitening, is the only method to produce a symmetric whitening matrix W_{ZCA} [59]. ZCA whitening maximizes the average cross-covariance between the whitened and original variables, and uniquely produces a symmetric cross-covariance matrix Φ [59].

$$W_{\mathrm{ZCA}} = \Sigma^{-1/2} \tag{2-53}$$

ZCA-cor is a scale-invariant alternative to ZCA whitening. ZCA-corr whitening is done by standardizing the original variables by multiplication by $V^{-1/2}$ and then employing ZCA whitening based on the cross-correlation rather than the cross-covariance matrix. ZCA-cor whitening maximizes the average cross-correlation between the whitened and original variables, and uniquely produces a symmetric cross-correlation matrix Ψ [59]. Unlike W_{ZCA}, $W_{\text{ZCA-cor}}$ is asymmetric. ZCA-corr is the only whitening method that ensures that the sphered variables z^j remain maximally correlated with the original variables x^j [59]. Using ZCA-corr is recommended if the aim is to obtain uncorrelated variables that are similar enough to the original variables to maintain their original interpretation.

$$W_{\text{ZCA-corr}} = P^{-1/2}\,V^{-1/2} \tag{2-54}$$

PCA whitening performs decorrelation and dimensionality reduction. PCA whitening is the unique sphering procedure that maximizes the compression of all original variables in each new variable using the cross-covariance Φ as the compression metric [59]. The whitening matrix W_{PCA} is subject to a sign ambiguity due to the eigenvectors U. Enforcing $\mathrm{diag}(U) > 0$ makes PCA whitening unique and guarantees Φ and Ψ to be positive diagonal [59].

$$W_{\mathrm{PCA}} = \Lambda^{-1/2}\,U^T \tag{2-55}$$

PCA-cor is a scale-invariant alternative to PCA whitening: cross-correlation, rather than cross-covariance is the metric to optimize. PCA-cor whitening maximally compresses all dimensions of the original data into each dimension of the whitened data using the cross-correlation Ψ as the compression metric [59]. Like W_{PCA}, $W_{\text{PCA-cor}}$ is subject to a sign ambiguity due to the eigenvectors G. Enforcing $\mathrm{diag}(G) > 0$ makes PCA-cor whitening unique and guarantees Φ and Ψ to be positive diagonal [59].

$$W_{\text{PCA-cor}} = \Theta^{-1/2}\,G^T\,V^{-1/2} \tag{2-56}$$

Cholesky whitening is based on Cholesky factorization $LL^T = \Sigma^{-1}$ where L is the unique lower triangular matrix with positive diagonal values. It produces a lower triangular positive diagonal cross-covariance Φ and cross-correlation Ψ [59].

$$W_{\mathrm{Chol}} = L^T \tag{2-57}$$

Chapter 3

Methodology: Classification & Biomarker Discovery

In subsection 1-2-2, we formulate the problem of biomarker discovery in high-dimensional, large-scale imaging mass spectrometry (IMS) data as a feature ranking problem. The IMS dataset is represented by an m-by-n matrix X whose m rows are observations or instances, and whose n columns are predictor variables or features. We use the superscript j, ranging from 1 to n, to denote a particular feature $x^j = X_{(:,j)}$ and we use subscript i, ranging from 1 to m, to denote a particular observation $x_i = X_{(i,:)}$. We use supervised machine learning algorithms to train predictive models for the biochemical classification of IMS data. Three binary classifiers, namely logistic regression, support vector machines, and random forests, are presented. In order to improve our understanding of the biochemical mechanisms being modeled, we rank the features according to their relative predictive importance using global model-specific and model-agnostic interpretability methods, implemented and extended as part of this thesis. The features with the highest discriminative power are candidate biomarkers that may be useful for recognizing a specific class within IMS data. Model interpretability is the ability to explain a supervised machine learning model's predictions by explicitly reporting the relative importance of its features. Please refer to subsection 1-2-2 for the definition of local versus global interpretability, and model-specific versus model-agnostic interpretability. We will see that logistic regression, linear support vector machines, and random forests are intrinsically interpretable models whose predictions can be explained using model-specific interpretability methods. Nonlinear support vector machines are black-box models that require the use of post-hoc model-agnostic interpretability approaches to produce a ranking of features. We will present, implement and apply two state-of-the-art post-hoc model agnostic methods, namely permutation importance and Shapley importance. Our implementation of Shapley importance as a global interpretability method is novel. Note that every supervised machine learning algorithm and every interpretability method discussed in Chapter 3 was implemented from scratch in MATLAB.

3-1 Logistic Regression

3-1-1 Regularized Logistic Regression

Logistic regression (LR) is a simple classification algorithm for binary problems consisting of an $(m \times n)$ input data matrix X whose m-dimensional output vector y has either positive or negative labels. Each of the n-dimensional observations x_i, where $i = 1, 2...m$, may be labeled $y_i = 1$ or $y_i = 0$ depending on whether it belongs to class $\mathcal{C}_+$ or $\mathcal{C}_-$ respectively. The n features are written x^j, where $j = 1, 2...n$. Hence the j^{th} element of x_i, where $j = 1, 2...n$, is written x_i^j.

Logistic regression models the probability $p(x_i)$ that a given test observation x_i belongs to the positive class $\mathcal{C}_+$ as per Equation 3-1 [44]. Equation 3-2 defines the probability that x_i belongs to the negative class $\mathcal{C}_-$. The hypothesis of the logistic regression (LR) classifier is defined by the sigmoid function as per Equation 3-3 [44]. As illustrated by Figure 3-1, the sigmoid function maps the linear (actually affine) combination of features $\theta_0 + \theta^T x_i$ to the probability interval $[0, 1]$. The steepness of the sigmoid curve is determined by the n-dimensional parameter θ.

$$p(x_i) = p(y_i = 1|x_i) = h_\theta(x_i) \tag{3-1}$$

$$p(y_i = 0|x_i) = 1 - p(y_i = 1|x_i) = 1 - h_\theta(x_i) \tag{3-2}$$

$$\begin{aligned} h_\theta(x_i) &= \frac{\exp(\theta_0 + \theta^T x_i)}{1 + \exp(\theta_0 + \theta^T x_i)} = \frac{1}{1 + \exp(-(\theta_0 + \theta^T x_i))} \\ \iff h_\theta(x_i) &= \frac{1}{1 + \exp(-\theta_0 - \theta_1 x_i^1 - \theta_2 x_i^2 - ... - \theta_n x_i^n)} \end{aligned} \tag{3-3}$$

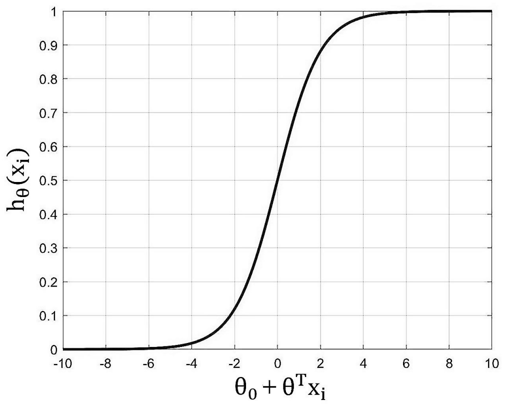


Figure 3-1: Sigmoid function

The odds of x_i belonging to $\mathcal{C}_+$ are defined by Equation 3-4 [61]. The logarithm of the odds, called the log-odds or logit function, is the inverse of the sigmoid function. The logit function maps the probability $p(x_i)$ back to $\theta_0 + \theta^T x_i$. The logarithm of the odds spans from $-\infty$ to $+\infty$ whereas the corresponding probability is bounded between 0 and 1. Since the logarithm of the odds is linear in the features x^j, with $j = 1, 2...n$, of a given observation x_i, the LR classifier is said to be a generalized linear model (GLM) with a logit linking function and a binary response [61]. Equation 3-5 demonstrates that the logistic regression model is a linear model for the log odds.

$$\frac{p(x_i)}{1-p(x_i)} = \exp\left(\theta_0 + \theta_1 x_i^1 + \theta_2 x_i^2 + ... + \theta_n x_i^n\right) \tag{3-4}$$

$$\log\left(\frac{p(x_i)}{1-p(x_i)}\right) = \theta_0 + \theta_1 x_i^1 + \theta_2 x_i^2 + ... + \theta_n x_i^n \tag{3-5}$$

The impact of increasing the j^{th} feature of observation x_i by one unit, while fixing all other features, is a multiplicative increase of the odds ratio by a factor $\exp(\theta_j)$, or - equivalently - an additive increase of the log odds by θ_j [34]. In Equation 3-16, we show that $\frac{\text{odds}_{x^j+1}}{\text{odds}} = \exp(\theta_j)$.

$$\frac{p(x_i)_{x_i^j+1}}{1-p(x_i)_{x_i^j+1}} \cdot \frac{1-p(x_i)}{p(x_i)} = \frac{\exp\left(\theta_0 + \theta_1 x_i^1 + ... + \theta_j(x_i^j+1) + ... + \theta_n x_i^n\right)}{\exp\left(\theta_0 + \theta_1 x_i^1 + ... + \theta_j x_i^j + ... + \theta_n x_i^n\right)} = \exp(\theta_j) \tag{3-6}$$

In logistic regression, the classification threshold is $h_\theta(x_i) = 0.5$ such that if $h_\theta(x_i) > 0.5$, the predicted label is $\hat{y}_i = +1$ and if $h_\theta(x_i) < 0.5$, the LR classifier predicts $\hat{y}_i = 0$ [44]. The decision boundary is the set that verifies Equation 3-7. Logistic regression divides the feature space into two by a hyperplane $\mathcal{H}$ defined in n dimensions by $\theta_0 + \theta^T x_i = 0$ where θ_0 is the intercept and θ is the n-dimensional normal vector. Observation x_i will be classified positive or negative depending on whether it is on the positive or negative side of hyperplane $\mathcal{H}$. Parameters $\theta_1, \theta_2 ... \theta_n$ are the weights, or coefficients, assigned to each of the n variables.

$$\mathcal{H} = \{x \in \mathbb{R}^n | p(y_i = \omega_+|x_i) = p(y_i = \omega_-|x_i) = 0.5\} \tag{3-7}$$

$$\hat{y}_i = \begin{cases} 1 & \text{if} \quad h_\theta(x_i) > 0.5 \iff \theta_0 + \theta^T x_i > 0 \\ 0 & \text{if} \quad h_\theta(x_i) < 0.5 \iff \theta_0 + \theta^T x_i < 0 \end{cases} \tag{3-8}$$

The maximum likelihood approach is used to estimate parameter θ and intercept θ_0 [61]. The likelihood $p(y_i|x_i)$ is the probability that the LR classifier predicts output y_i considering input x_i: $p(y_i|x_i)$ can be rewritten as per Equation 3-9 [61]. Assuming that $(x_1, y_1), (x_2, y_2) ... (x_m, y_m)$ are independent and identically distributed (IID), the total likelihood is the product of the m likelihoods. The aim of the maximum likelihood approach is to estimate θ so that the prediction $\hat{y}_i$ corresponds as accurately as possible to the known output label y_i. Parameter θ is therefore chosen to maximize the logarithm of the total likelihood $\mathcal{L}(\theta_0, \theta)$ as per Equation 3-10. Maximizing $\mathcal{L}(\theta_0, \theta)$ is equivalent to minimizing the convex cost function $J(\theta_0, \theta)$ of Equation 3-11 [44]. Since there is no closed form solution to this problem, minimizing $J(\theta_0, \theta)$ must be done numerically.

$$p(y_i|x_i) = \begin{cases} h_\theta(x_i) & \text{if} \quad y_i = 1 \\ 1 - h_\theta(x_i) & \text{if} \quad y_i = 0 \end{cases} \iff p(y_i|x_i) = (h_\theta(x_i))^{y_i} + (1 - h_\theta(x_i))^{1-y_i} \tag{3-9}$$

$$\mathcal{L}(\theta_0, \theta) = \log\left(\prod_{i=1}^{m} p(y_i|x_i)\right) \tag{3-10}$$

$$\begin{aligned} J(\theta_0, \theta) &= -\frac{1}{m} \log\left(\prod_{i=1}^{m} p(y_i|x_i)\right) = -\frac{1}{m}\left(\sum_{i=1}^{m} \log(p(y_i|x_i))\right) \\ &= -\frac{1}{m}\left(\sum_{i=1}^{m} y_i \log(h_\theta(x_i)) + (1-y_i)\log(1-h_\theta(x_i)\right) \end{aligned} \tag{3-11}$$

Cost function $J(\theta_0, \theta)$ is minimized using the gradient descent method. Parameters $\theta_0, \theta_1, \theta_2...\theta_n$ are iteratively updated as per Equation 3-12 where α is the learning rate [44]. Hyperparameter α should be chosen by validation. The $n+1$ parameters must be simultaneously updated at each iteration. The gradient of the cost function is an n-dimensional vector whose j^{th} element is defined by Equation 3-13.

$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \tag{3-12}$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)x_i^j \tag{3-13}$$

Applying ridge regularization to logistic regression changes the cost function $J(\theta_0, \theta)$ to $J^*(\theta_0, \theta)$ as defined by Equation 3-14 where $\lambda \geq 0$ [44]. Increasing the regularization parameter λ boosts the relative importance of the penalty term in the regularized cost function $J^*(\theta_0, \theta)$ and enforces a preference for weights with small squared norms. When minimizing $J^*(\theta_0, \theta)$ by gradient descent, the penalty term enforces a trade-off between minimizing the training error and reducing the square of the weights' magnitudes $|\theta_1|^2, |\theta_2|^2...|\theta_n|^2$. Smaller weights $\theta_1, \theta_2...\theta_n$ simplify the classifier's decision boundary, which in turn reduces the risk of overfitting and improves generalization [44]. Note the difference between ridge regularization, also called L_2 regularization, in which the penalty is equivalent to the square of the weights' magnitudes, and Lasso regularization, also called L_1 regularization, in which the penalty is equivalent to the absolute value of the weights' magnitudes [62].

$$J^*(\theta_0, \theta) = \frac{1}{m}\sum_{i=1}^{m}\left(-y^i \log(h_\theta(x^i)) - (1-y^i)\log(1-h_\theta(x^i))\right) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{3-14}$$

The regularized cost function is differentiable. The j^{th} entry of the regularized cost function's gradient is given by Equation 3-15 for $j = 1, 2...n$. Note that the intercept θ_0 is not regularized.

$$\begin{aligned} \frac{\partial J^*(\theta)}{\partial \theta_j} &= \frac{1}{m}\sum_{i=1}^{m}\left((h_\theta(x^i) - y^i)x_j^i\right) \quad \text{for } j = 0 \\ \frac{\partial J^*(\theta)}{\partial \theta_j} &= \frac{1}{m}\sum_{i=1}^{m}\left((h_\theta(x^i) - y^i)x_j^i\right) + \frac{\lambda}{m}\theta_j \quad \text{for } j \geq 1 \end{aligned} \tag{3-15}$$

3-1-2 Measuring Feature Importance

3-1-2-1 Magnitude of the Feature Weights

Ranking features according to their relative predictive importance is key to machine learning model interpretability. LR classifiers are intrinsically interpretable models whose predictions can be explained by LR's model-specific interpretability method. Intuitively, if a feature has a larger magnitude than another, it should be more important. Also, the sign of the feature's weight indicates whether the feature contributes towards increasing (if positive) or decreasing (if negative) the model's prediction. If all the features were uncorrelated, the interpretations of the logistic regression classifier's coefficients would be simple: the larger $|\theta_j|$, the stronger the influence of feature x^j on the LR classifier, hence the stronger its predictive power. In Equation 3-16, the predictive importance of feature x^j is determined by a multiplicative increase of the odds ratio by a factor $\exp(\theta_j)$ that is achieved by increasing the value of x^j and fixing the values of all other features. Yet, studying the impact of a unit increase in one feature while holding the other features constant is only relevant in the absence of feature inter-dependencies. In other words, Equation 3-16 implicitly assumes that feature x^j is uncorrelated with the other features. The problem is that the assumption of feature uncorrelatedness does not hold for imaging mass spectrometry (IMS) data. Evaluating the relative importance of features by simply looking at the magnitude of their relative weights may therefore be misleading in the case of imaging mass spectrometry (IMS) data analysis.

$$s_j = |\theta_j| \tag{3-16}$$

Note that evaluating the importance of a feature based on the magnitude of its corresponding weight in the decision boundary of the LR model is affected by correlation bias: features that belong to groups of correlated features receive smaller weights in the decision boundary of the LR model because of their shared contribution to the predictive task [33]. When presented with groups of informative correlated features, the LR training algorithm will tend to assign a large weight $|\theta_j|$ to only one arbitrary representative of each group of correlated features. As a result, feature x^j will appear as being very important but the features that are correlated with x^j are redundant. These features will be discarded as unimportant, not because they do not have discriminative power, but because they do not provide the LR classifier with any more information than x^j.

3-1-2-2 Relative Weights Analysis

Relative weights analysis was developed by Johnson in 2000 for determining the relative importance of predictor variables in multiple linear regression [63]. Johnson defines a feature's importance as a feature's relative contribution to the variance of the prediction, considering its direct effect and its joint effect with other features [63]. Johnson's definition of feature importance is therefore compatible with our definition, stated in subsection 1-2-2. Tonidanel and LeBreton extended relative weights analysis to classification by logistic regression in 2010 [64]. Tonidanel and LeBreton's work enables us to determine the importance of multiple predictor variables while accounting for feature multicollinearity. The central idea of relative weights analysis is to perform logistic regression on a set of maximally related orthogonal features Z, rather than on the original correlated features X, and then map the feature weights obtained for Z back to X. Each new feature z^j is uncorrelated with the other new features but maximally correlated to their own respective original feature x^j, for $j = 1, 2...n$ [63]. Each z^j is essentially a maximally rotated orthogonal variable that represents the relationship of each x^j with the model prediction y [63].

Relative weights analysis for logistic regression involves the four following steps [64]:

- Create orthogonal approximations $z^j = Z_{(:,j)}$ of the original features $x^j = X_{(:,j)}$, for $j = 1, 2...n$, as per Equation 3-17. Refer to Equation 2-21 for the singular value decomposition (SVD) of matrix X. We have $X = USV^T$ where U is the matrix of left singular vectors, S is the diagonal matrix containing the singular values, and V is the matrix of right singular vectors. Equation 3-17 is the best fitting least squares orthonormal approximation of X [64]. It is recommended to center and standardize the orthogonal features z^j [64].
$$Z = UV^T \tag{3-17}$$
- Obtain coefficients linking the original features x^j to the orthogonal features z^j. The n-by-n mapping matrix Λ^* is computed as per Equation 3-18 [64].
$$\Lambda^* = (Z^T Z)^{-1} Z^T X \tag{3-18}$$
- Obtain coefficients $\beta^*_j = \beta^*(j)$ linking the orthogonal predictors z^j to the criterion y. We train an LR model on the orthogonal set Z and obtain feature weights θ_j for all orthogonal features z^j, with $j = 1, 2...n$. Tonidanel refers to the n-dimensional vector θ as the unstandardized logistic regression coefficients [64]. The LR model produces a prediction $\hat{y}$. Equation 3-19 defines the logistic regression analog to R^2 [64]: R^2_{log} is obtained by computing the sum of squared errors (SSE) and the sum of squares total (SST). The log-odds or logit function of $\hat{y}$ is defined by Equation 3-20 and its standard deviation is written $\sigma_{\text{logit}(\hat{y})}$. Finally, Equation 3-21 defines the standardized logistic regression coefficients β^* such that β^*_j represents the relative importance of each uncorrelated feature z^j [64].
$$R^2_{\text{log}} = 1 - \frac{\text{SSE}}{\text{SST}} = 1 - \frac{\sum_{i=1}^{m}(y - \hat{y})^2}{\sum_{i=1}^{m}(y - \bar{y})^2} \tag{3-19}$$
$$\text{logit}(\hat{y}) = \log\left(\frac{\hat{y}}{1 - \hat{y}}\right) \tag{3-20}$$
$$\beta^* = \frac{\theta \, R_{\text{log}}}{\sigma_{\text{logit}(\hat{y})}} \tag{3-21}$$
- Combine the linking coefficients Λ^* and β^* to obtain the relative weights ϵ of the original features in X [64]. Each relative weight ϵ_j provides an estimate of the influence each original feature x^j makes to the model prediction $\hat{y}$. The relative weights sum to R^2_{log} and can be scaled as the percentage of predicted variance accounted for by each x^j [64].
$$\epsilon = \Lambda^{*2} \beta^{*2} \tag{3-22}$$

3-2 Support Vector Machines

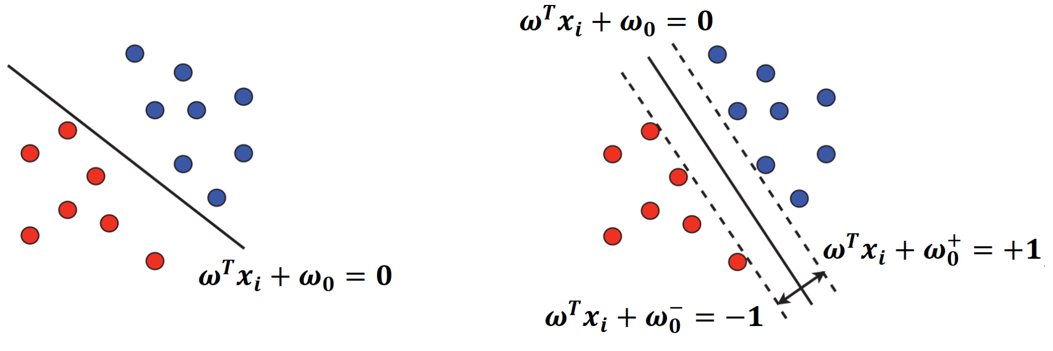
3-2-1 Linear Support Vector Machines

3-2-1-1 Linear Hard-margin Support Vector Machines

Building a classifier for linearly separable data requires choosing the best hyperplane among several possibilities. In Figure 3-2, two different hyperplanes correctly separate the red class from the blue class. Yet, the separating hyperplane of Figure 3-11b is intuitively a better decision boundary than the hyperplane of Figure 3-11a because it has a larger margin. The margin is the minimum distance to the training observations. In Figure 3-11b, the distance between the two dashed lines parallel to the decision boundary is the hyperplane's margin. The larger the margin, the more robust the separating hyperplane is to measurement error (i.e. noise) and the better it generalizes to new observations [7]. The maximal margin hyperplane is the separating hyperplane for which the margin is largest. The corresponding classifier was introduced by Vapnik and Chervonenkis in 1964 [6].

$$\forall x_i \in \mathcal{H}, \;\; g(x_i) = \omega^T x_i + \omega_0 = 0 \tag{3-23}$$

Assuming an $(m \times n)$ input data matrix X and an output vector y, the general equation of separating hyperplane $\mathcal{H}$ is given by Equation 3-23, where $\omega \in \mathbb{R}^n$ and $\omega_0 \in \mathbb{R}$. Note that Equation 3-23 is invariant to scaling. Since the marginal hyperplanes are parallel to the separating hyperplane, they admit the same normal vector ω with different biases ω_0^- and ω_0^+. The margins delimiting the positive and negative classes $\mathcal{C}_+$ and $\mathcal{C}_-$ are denoted $\mathcal{M}_+$ and $\mathcal{M}_-$ respectively.


$\omega^T x_i + \omega_0 = 0$
$\omega^T x_i + \omega_0 = 0$
$\omega^T x_i + \omega_0^+ = +1$
$\omega^T x_i + \omega_0^- = -1$

(a) Random separating hyperplane **(b)** Maximal margin hyperplane

Figure 3-2: Two possible hyperplanes for linearly separable data. Figure adapted from [6]. Copyright © 2018 Massachusetts Institute of Technology.

Computing the separating hyperplane's margin is equivalent to computing the distance from $\mathcal{H}$ to the nearest observation. We start by considering an arbitrary observation $x_i \in \mathbb{R}^n$. Observation x_i can be expressed as per Equation 3-24 where x_i^p is the normal projection of x_i onto $\mathcal{H}$ and r_i is the signed distance - positive or negative depending on which side of $\mathcal{H}$ x_i lies. Since $g(x_i^p) = 0$, we have Equation 3-25. Assuming a binary problem whose classes $\mathcal{C}_+$ and $\mathcal{C}_-$ have labels $y_i = +1$ and $y_i = -1$ respectively, distance d_i from x_i to $\mathcal{H}$ can be written as per Equation 3-26.

$$x_i = x_i^p + r_i \frac{\omega}{\|\omega\|} \tag{3-24}$$

$$g(x_i) = \omega^T x_i + \omega_0 = r_i \|\omega\| \tag{3-25}$$

$$d_i = |r_i| = \frac{|\omega^T x_i + \omega_0|}{\|\omega\|} = \frac{y_i(\omega^T x_i + \omega_0)}{\|\omega\|} \tag{3-26}$$

Marginal hyperplanes $\mathcal{M}_+$ and $\mathcal{M}_-$ are defined by $\omega^T x_i + \omega_0^+ = +1$ and $\omega^T x_i + \omega_0^- = -1$ respectively. Equation 3-27 states that any observation $x_i \in \mathbb{R}^n$ belonging to a marginal hyperplane meets $|\omega^T x_i + \omega_0| = 1$: this is the separating hyperplane's minimum distance to the training observations.

$$\min_{i=1,2...m} y_i(\omega^T x_i + \omega_0) = \min_{i=1,2...m} |\omega^T x_i + \omega_0| = 1 \tag{3-27}$$

The margin ρ is equivalent to the distance from either $\mathcal{M}_+$ or $\mathcal{M}_-$ to the separating hyperplane $\mathcal{H}$. According to Equation 3-28, ρ depends directly on the Euclidean norm, or L2-norm, of vector ω. Maximizing ρ is equivalent to minimizing $\|\omega\|$ [6].

$$\forall x_i \in \mathcal{M}_\pm,\ \ \rho = d_i = \frac{|\omega^T x_i + \omega_0|}{\|\omega\|} = \frac{1}{\|\omega\|} \tag{3-28}$$

The maximum margin classifier, also referred to as the linear hard-margin support vector machine (SVM) [7], implements the decision rule of Equation 3-29. If observation x_i is correctly classified, the prediction verifies $\hat{y}_i = y_i$ hence $y_i(\omega^T x_i + \omega_0) \geq 1$.

$$\hat{y}_i = \begin{cases} +1 & \text{if} \quad \omega^T x_i + \omega_0 \geq +1 \\ -1 & \text{if} \quad \omega^T x_i + \omega_0 \leq -1 \end{cases} \tag{3-29}$$

The maximal margin hyperplane solves the constrained convex optimization problem of Equation 3-30, called the primal problem [6]. The quadratic objective function $f : \omega \to \frac{1}{2}\|\omega\|^2$ is differentiable: it has gradient $\Delta_f = \omega$ and Hessian $H_f = I_n$ [6]. The fact that H_f is positive definite, added to the fact that the m inequality constraints are affine functions, guarantee convexity and the existence of a global minimum. Indeed, there is a unique normal vector ω defining the maximal margin hyperplane. The bias ω_0 does not appear in the cost function, but it is involved in the constraint.

$$\begin{gathered} \text{minimize } J(\omega, \omega_0) = \frac{1}{2}\|\omega\|^2 \\ \text{subject to: } \forall x_i,\ y_i(\omega^T x_i + \omega_0) \geq 1 \end{gathered} \tag{3-30}$$

We introduce Lagrange variables $\lambda_i \geq 0$ for $i = 1, 2...m$, associated to the m constraints and denote by λ the vector $(\lambda_1, \lambda_2...\lambda_m)^T$. The Lagrangian is defined in Equation 3-31 for $\omega \in \mathbb{R}^n$, $\omega_0 \in \mathbb{R}$ and $\lambda \in \mathbb{R}_+^m$ [6]. The inequality constraint is now written in standard form: $y_i(\omega^T x_i + \omega_0) - 1 \geq 0$.

$$\mathcal{L}(\omega, \omega_0, \lambda) = \frac{1}{2}\|\omega\|^2 - \sum_{i=1}^{m} \lambda_i (y_i(\omega^T x_i + \omega_0) - 1) \tag{3-31}$$

The Karush-Kuhn-Tucker (KKT) conditions are obtained by setting the gradient of the Lagrangian with respect to the primal variables ω and ω_0 to zero. The complementary slackness condition states that, for every observation, either the Lagrange multiplier is zero or the constraint is zero.

$$\begin{aligned} \frac{\partial}{\partial \omega}\mathcal{L}(\omega, \omega_0, \lambda) = 0 \;\;&\Rightarrow\;\; \omega = \sum_{i=1}^{m} \lambda_i y_i x_i \\ \frac{\partial}{\partial \omega_0}\mathcal{L}(\omega, \omega_0, \lambda) = 0 \;\;&\Rightarrow\;\; \sum_{i=1}^{m} \lambda_i y_i = 0 \\ \forall i,\; \lambda_i \Big(y_i(\omega^T x_i + \omega_0) - 1\Big) = 0 \;\;&\Rightarrow\;\; \lambda_i = 0 \;\text{ or }\; \omega^T x_i + \omega_0 = 1 \end{aligned} \tag{3-32}$$

Hence, the weight vector ω solution to the problem is a linear combination of the training observations $x_1, x_2 ... x_m$ [6]. Support vectors are observations belonging to the margins $\mathcal{M}_+$ and $\mathcal{M}_-$ that verify $y_i(\omega^T xi + \omega_0) = 1$ where $\lambda_i \neq 0$ [6]. Observations falling outside the region between $\mathcal{M}_+$ and $\mathcal{M}_-$ verify $\lambda_i = 0$. Note that while the solution ω of the SVM problem is unique, the support vectors are not. In n-dimensional feature space, $n+1$ observations are sufficient to define a hyperplane [6]. Thus, when more than $n+1$ observations lie on a marginal hyperplane, different choices are possible for the $n+1$ support vectors.

The support vectors fully define the maximal margin hyperplane. The maximal margin hyperplane does not depend on any other observations: changing any observation not belonging to the marginal hyperplanes would not affect the separating hyperplane, provided that the change does not cause the observation to cross the boundary set by the positive or negative margin [65].

Deriving the dual formulation of Equation 3-30 is done by maximizing the Lagrangian $\mathcal{L}(\omega, \omega_0, \lambda)$ with respect to the Lagrangian multipliers. Equating to zero the gradient of the Lagrangian, with respect to ω and ω_0, in Equation 3-32 resulted in two stationarity constraints: $\omega = \sum_{i=1}^{m} \lambda_i y_i x_i$ and $\sum_{i=1}^{m} \lambda_i y_i = 0$. Substituting these constraints into Equation 3-31 yields Equation 3-33 where the Lagrangian is expressed as a function of $\lambda \geq 0$ [7]. The dual formulation reveals an important property of support vector machines: the solution depends only on inner products between observations and not directly on the observations themselves. Hence $\mathcal{L}(\lambda)$ does not explicitly depend on the dimensionality of the input space [66]: this becomes crucial when extending SVMs to nonlinear decision boundaries thanks to the kernel trick (refer to section 3.4.3).

$$\begin{aligned} \mathcal{L}(\lambda) &= \frac{1}{2}\omega^T\omega - \sum_{i=1}^{m}\lambda_i y_i \omega^T x_i - \sum_{i=1}^{m}\lambda_i y_i \omega_0 + \sum_{i=1}^{m}\lambda_i \\ \mathcal{L}(\lambda) &= \frac{1}{2}\sum_{i=1}^{m}\lambda_i y_i x_i^T \sum_{j=1}^{m}\lambda_j y_j x_j - \sum_{i=1}^{m}\lambda_i y_i \sum_{j=1}^{m}\lambda_j y_j x_j^T x_i + \sum_{i=1}^{m}\lambda_i \\ \mathcal{L}(\lambda) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \lambda_i \lambda_j x_i^T x_j - \sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \lambda_i \lambda_j x_i^T x_j + \sum_{i=1}^{m}\lambda_i \\ \mathcal{L}(\lambda) &= -\frac{1}{2}\sum_{i,j=1}^{m}\lambda_i\lambda_j y_i y_j x_i^T x_j + \sum_{i=1}^{m}\lambda_i \end{aligned} \tag{3-33}$$

The dual optimization problem for linear hard-margin SVM classifiers is given by Equation 3-34 [66]. The cost function is differentiable and the Hessian is $H = -G$ where G is the positive semi-definite Gram matrix such that $G_{i,j} = \langle y_i x_i, y_j x_j \rangle$. Since the objective function is concave and the constraints are affine, minimizing $-\mathcal{L}(\lambda)$ transforms the dual optimization problem into a convex problem that can be solved by quadratic programming (QP) [6]. Although the solution to the dual problem is generally a lower bound on the solution to the primal problem, strong duality holds for convex QP [7]. Therefore, the solution to the dual SVM problem is equal to the solution of the primal SVM problem.

$$\begin{aligned} \text{maximize } \mathcal{L}(\lambda) = \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{m} \lambda_i \lambda_j y_i y_j x_i^T x_j \\ \text{subject to: } \sum_{i=1}^{m} \lambda_i y_i = 0 \\ \lambda_i \geq 0 \end{aligned} \tag{3-34}$$

Once the optimal Lagrange multiplier vector $\lambda^\star$ has been computed by maximizing the Lagrangian, the normal vector $\omega^\star$ to the maximum margin hyperplane is obtained from Equation 3-35 [7].

$$\omega^\star = \sum_{i=1}^{m} \lambda_i^\star y_i x_i \tag{3-35}$$

The Lagrangian multipliers of support vectors verify $\lambda_k > 0$ where $k = 1, 2 ... m_{\text{SV}}$ where m_{SV} is the number of support vectors. According to the complementary slackness condition, the support vectors verify $y_k(\omega^{\star T} x_k + \omega_0^\star) = 1$. Hence $\omega_0^\star = y_k - \omega^{\star T} x_k$ [7]. Averaging the bias over all support vectors, as done in Equation 3-36, guarantees numerical stability [66].

$$\omega_0^\star = \frac{1}{m_{\text{SV}}} \sum_{k \in \text{SV}} (y_k - \omega^{\star T} x_k) \tag{3-36}$$

Since $\lambda_k > 0$ for the support vectors and $\lambda_i = 0$ otherwise, $\omega_0^\star = y_k - \omega^{\star T} x_k$ can be rewritten so as to express the margin ρ as a function of the L1-norm of Lagrangian multiplier vector $\lambda^\star$ [6]:

$$\underbrace{\sum_{i=1}^{m} \lambda_k y_k \omega_0^\star}_{0} = \underbrace{\sum_{i=1}^{m} \lambda_k y_k^2}_{\sum_{i=1}^{m} \lambda_i^\star} - \underbrace{\sum_{i=1}^{m} \sum_{k=1}^{m_{\text{SV}}} \lambda_i^\star \lambda_k y_i y_k x_i^T x_k}_{\|\omega\|^2} \Rightarrow \|\omega\|^2 = \sum_{i=1}^{m} \lambda_i^\star \Rightarrow \rho = \frac{1}{\|\omega\|} = \frac{1}{\sqrt{|\lambda^\star|}} \tag{3-37}$$

The decision rule of the linear SVM classifier is defined by Equation 3-38: predict the output label of test observation x_i requires computing the inner product between x_i and each of the training observations x_j for which $\lambda_j > 0$ [7].

$$\hat{y}_i = \text{sgn}\left(\omega^{\star T} x_i + \omega_0^\star\right) = \text{sgn}\left(\sum_{j=1}^{m} y_j \lambda_j^\star x_j^T x_i + \omega_0^\star\right) = \text{sgn}\left(\sum_{\lambda_j^\star > 0} y_j \lambda_j^\star x_j^T x_i + \omega_0^\star\right) \tag{3-38}$$

3-2-1-2 Linear Soft-margin Support Vector Machines

The maximal margin classifier performs well on linearly separable data. However, the positive and negative classes are seldom linearly separable in practice. The generalization of the maximal margin classifier to non-separable classes is known as the linear soft-margin support vector machine. Even if the classes are linearly separable, an SVM classifier whose hyperplane tolerates margin violation and misclassification on the training set is often preferable in the interest of reduced sensitivity to individual observations and better generalization [65]. Indeed, the soft-margin SVM has proven more robust to outliers and less prone to overfitting than the hard-margin SVM classifier [7].

Rather than seeking to maximize the margin so that every training observation is not only on the correct side of the separating hyperplane $\mathcal{H}$ but also on the correct side of the marginal hyperplanes $\mathcal{M}_+$ and $\mathcal{M}_-$, the soft-margin SVM allows some observations to be on the incorrect side of $\mathcal{M}_\pm$, or even on the incorrect side of $\mathcal{H}$ - in which case they are misclassified [65].

Observations may fall into one of the three following cases. Correctly classified observations that fall outside the region between $\mathcal{M}_+$ and $\mathcal{M}_-$ comply with the constraints of Equation 3-39 [66]. If Equation 3-39 holds, the soft-margin SVM classifier is equivalent to the hard-margin SVM classifier. Correctly classified observations that fall within the region between $\mathcal{M}_+$ and $\mathcal{M}_-$, and hence violate the margins, verify Equation 3-40 [66]. Misclassified observations obey Equation 3-41 [66].

$$y_i(\omega^T x_i + \omega_0) \geq 1 \tag{3-39}$$

$$0 \leq y_i(\omega^T x_i + \omega_0) < 1 \tag{3-40}$$

$$y_i(\omega^T x_i + \omega_0) < 0 \tag{3-41}$$

All three cases can be written in the form of Equation 3-42 by introducing slack variables $\xi_1, \xi_2 ... \xi_m$ [66]. Slack variable ξ_i measures the distance by which observation x_i violates the desired inequality constraint of Equation 3-39. Effectively, ξ_i implements a hinge-loss: $\xi_i = \max\{0, 1 - y_i(\omega^T x_i + \omega_0)\}$. Equation 3-39 is obtained from Equation 3-42 for $\xi_i = 0$ [66]. Equations 3-40 and 3-41 are obtained from Equation 3-42 for $0 < \xi_i \leq 1$ and $\xi_i > 1$ respectively [66]. Therefore, $\xi_i > 0$ indicates that observation x_i either violates the margins or is misclassified. In Figure 3-3, observation $x^\star$ is incorrectly classified hence $\xi^\star > 1$ whereas observation $x^{\star\star}$ is correctly classified but violates the margin hence $0 < \xi^{\star\star} \leq 1$. Observation noted $x^{\star\star\star}$ is correctly classified without violating the margins so $\xi^{\star\star\star} = 0$.

$$y_i(\omega^T x_i + \omega_0) \geq 1 - \xi_i \tag{3-42}$$

Equation 3-42 introduces the following constraints on the soft-margin SVM classifier [9]:

- For $y_i = +1$, $\omega^T x_i + \omega_0 \geq 1 - \xi_i$
- For $y_i = -1$, $\omega^T x_i + \omega_0 \geq -1 + \xi_i$

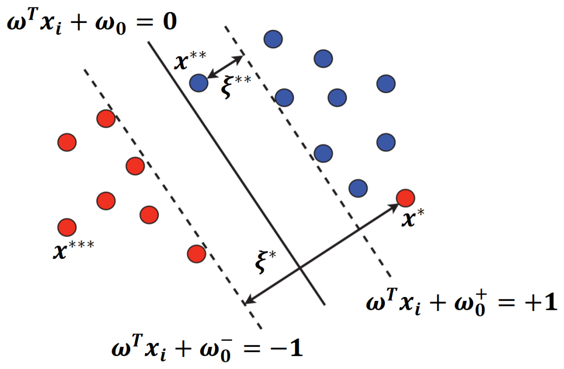

$\omega^T x_i + \omega_0 = 0$
x^{**}
ξ^{**}
x^*
x^{***}
ξ^*
$\omega^T x_i + \omega_0^+ = +1$
$\omega^T x_i + \omega_0^- = -1$

Figure 3-3: Soft-margin support vector margin classifier. Figure adapted from [6]. Copyright © 2018 Massachusetts Institute of Technology.

The goal is to maximize the margin while keeping the number of observations for which $\xi_i > 0$ as small as possible. This is equivalent to the optimization problem of Equation 3-43 where regularization hyperparameter $C \geq 0$ determines the trade-off between margin maximization and outlier-penalty minimization [6]. Similarly to the optimization problem associated with the hard-margin SVM, Equation 3-43 is a convex QP problem [66].

$$\begin{aligned} \text{minimize} \;\; & J(\omega, \omega_0, \xi) = \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \;\; & y_i(\omega^T x_i + \omega_0) \geq 1 - \xi_i \\ & \xi_i \geq 0 \;\; \text{for} \;\; i = 1, 2...m \end{aligned} \tag{3-43}$$

Penalty parameter C determines the number and severity of the violations to the margin and separating hyperplane that are tolerated [65]. Increasing C decreases violation tolerance, hence a narrower margin with fewer support vectors. Conversely, decreasing C increases violation tolerance, resulting in a wider margin with more support vectors. C also controls the bias-variance trade-off of the soft-margin SVM classifier [65]: a classifier whose C is small has high bias and low variance whereas a classifier whose C is large has a low bias but a high variance.

$$\mathcal{L}(\omega, \omega_0, \xi, \lambda, \mu) = \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\mu_i\xi_i - \sum_{i=1}^{m}\lambda_i\left(y_i(\omega^T x_i + \omega_0) - 1 + \xi_i\right) \tag{3-44}$$

The Lagrangian of the soft-margin SVM is given by Equation 3-44 where $\lambda_i \geq 0$ are the Lagrange multipliers of inequality constraint $y_i(\omega^T x_i + \omega_0) \geq 1 - \xi_i$ and $\mu_i \geq 0$ are the Lagrange multipliers of $\xi_i \geq 0$ for $i = 1, 2...m$ [7]. The Karush-Kuhn-Tucker conditions state that the gradient of $\mathcal{L}(\omega, \omega_0, \xi, \lambda, \mu)$ with respect to primal variables ω, ω_0, ξ cancel out [6].

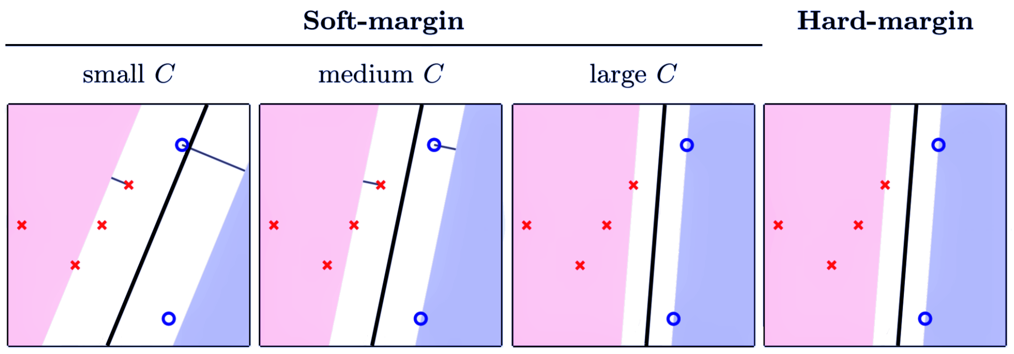


Figure 3-4: Support vector machine classification boundary for different choices of penalty parameter C: soft-margin SVMs allow for some observations to violate the margin, whereas hard-margin SVMs do not. Figure adapted from [7]. Copyright © 2019 Yaser S. Abu-Mostafa, Malik Magdon-Ismail, Hsuan-Tien Lin.

Hence Equation 3-45.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \omega} &= \omega - \sum_{i=1}^{m} \lambda_i y_i x_i = 0 &&\Rightarrow\ \omega = \sum_{i=1}^{m} \lambda_i y_i x_i \\
\frac{\partial \mathcal{L}}{\partial \omega_0} &= -\sum_{i=1}^{m} \lambda_i y_i = 0 &&\Rightarrow\ \sum_{i=1}^{m} \lambda_i y_i = 0 \\
\frac{\partial \mathcal{L}}{\partial \xi_i} &= C - \mu_i - \lambda_i = 0 &&\Rightarrow\ \forall i,\ \mu_i + \lambda_i = C \\
\forall i,\ \lambda_i \left(y_i(\omega^T x_i + \omega_0) - 1 + \xi_i \right) &= 0 &&\Rightarrow\ \lambda_i = 0 \ \text{ or } \ y_i(\omega^T x_i + \omega_0) = 1 - \xi_i \\
\forall i,\ \mu_i \xi_i &= 0 &&\Rightarrow\ \mu_i = 0 \ \text{ or } \ \xi_i = 0
\end{aligned} \tag{3-45}$$

Normal vector ω is a linear combination of the training observations: $\omega = \sum_{i=1}^{m} \lambda_i y_i x_i$. Hence the support vectors are the observations verifying $\lambda_i \neq 0$.

The complementary slackness condition states that $\lambda_i \neq 0 \iff y_i(\omega^T x_i + \omega_0) = 1 - \xi_i$. Therefore, if $\xi_i = 0$, $y_i(\omega^T x_i + \omega_0) = 1$ so x_i lies on the marginal hyperplane $\mathcal{M}_\pm$ [6]. However, if $\xi_i \neq 0$, observation x_i violates the margin and may even be on the wrong side of boundary $\mathcal{H}$ [6]. The support vector of the soft-margin SVM are the observations that lie either on the margin or on the wrong side of the margin for their class.

Since observations that are on the right side of the margin verify $\xi_i = 0 \Rightarrow \mu_i \neq 0$ as well as $y_i(\omega^T x_i + \omega_0) \neq 1$, their Lagrange multipliers are $\lambda_i = 0$. Observations that violate the margin verify $\xi_i \neq 0 \Rightarrow \mu_i = 0$ so $\lambda_i = C$. Observations belonging to margins $\mathcal{M}_\pm$ therefore verify $0 < \lambda_i < C$. The KKT conditions can therefore be written as per Equation 3-46.

$$\begin{aligned}
\lambda_i = 0 &\iff y_i(\omega^T x_i + \omega_0) \geq 1 \\
0 < \lambda_i < C &\iff y_i(\omega^T x_i + \omega_0) = 1 \\
\lambda_i = C &\iff y_i(\omega^T x_i + \omega_0) \leq 1
\end{aligned} \tag{3-46}$$

Since Lagrange multipliers λ_i and μ_i sum to C, we can replace μ_i by $C - \lambda_i$ to simplify the Lagrangian in Equation 3-47 [7]. Furthermore, since both λ_i and μ_i are non-negative, the constraint on Lagrange multipliers $\mu_i \geq 0$ is equivalent to $\lambda_i \leq C$: Lagrange multipliers λ_i are bounded by C [6].

$$\begin{aligned} \mathcal{L}(\omega, \omega_0, \xi, \lambda) &= \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}\lambda_i\left(1 - y_i(\omega^T x_i + \omega_0) - \xi_i\right) - \sum_{i=1}^{m}(C - \lambda_i)\xi_i \\ \mathcal{L}(\omega, \omega_0, \xi, \lambda) &= \frac{1}{2}\|\omega\|^2 + \sum_{i=1}^{m}\lambda_i\left(1 - y_i(\omega^T x_i + \omega_0)\right) \end{aligned} \tag{3-47}$$

Remarkably, the obtained Lagrangian is equal to the Lagrangian of the hard-margin SVM problem (refer to Equation 3-33). The objective function of the soft-margin SVM optimization problem is therefore equal to that of the hard-margin SVM in Equation 3-34. However, the soft-margin SVM problem has an additional box constraint: $0 \leq \lambda_i \leq C$ for $i = 1, 2...m$.

$$\begin{aligned} \text{maximize} \;\; \mathcal{L}(\lambda) &= \sum_{i=1}^{m}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{m}\lambda_i\lambda_j y_i y_j x_i^T x_j \\ \text{subject to:} \;\; &\sum_{i=1}^{m}\lambda_i y_i = 0 \\ &0 \leq \lambda_i \leq C \end{aligned} \tag{3-48}$$

The objective function of Equation 3-48 is concave and infinitely differentiable so minimizing $-\mathcal{L}(\lambda)$ is a convex QP problem. Strong duality holds so the solutions $\omega^\star$ and $\omega_0^\star$ of the primal problem can be determined from that of the dual problem using the KKT conditions. The normal vector to the optimal separating hyperplane is $\omega^\star = \sum_{i=1}^{m}\lambda_i^\star y_i x_i$. Bias $\omega_0^\star$ is computed using only the support vectors lying on the marginal hyperplanes that verify $\xi_i = 0 \iff \omega_0^\star = y_i - \omega^{\star T} x_i$. Finally, the classification hypothesis of the soft-margin SVM is given by $\hat{y}_i = \operatorname{sgn}\left(\omega^{\star T} x_i + \omega_0^\star\right)$ according to Equation 3-38.

3-2-1-3 Measuring Feature Importance

Linear support vector machines are intrinsically interpretable. Since the key idea of SVMs is to maximize the margin separating the two classes while minimizing the classification error, the most important features should maximize the separation between classes [67]. In Equation 3-49, we define a measure of class separation that takes the imbalanced class cardinality into account: $\text{card}(\mathcal{C}_+)$ and $\text{card}(\mathcal{C}_-)$ are the number of observations belonging to the positive class $\mathcal{C}_+$ and negative class $\mathcal{C}_-$ respectively, m_j^+ and m_j^- are their respective means in n-dimensional feature space [67]. In Equation 3-50, we define the contribution s_j of feature x^j for $j = 1, 2...n$: s_j is a measure of feature importance that considers both the feature's weight ω_j in the decision boundary of the linear SVM classifier and the class means [67]. The larger s_j is, the more feature x^j contributed to the class separation and therefore the more it is important to the linear SVM classifier [67]. Note that s_j does not account for inter-dependencies between x^j and the other features.

$$S = \frac{1}{\mathrm{card}(\mathcal{C}_+)} \sum_{x_i \in \mathcal{C}_+} \left(\omega^{\star T} x_i + \omega_0^\star\right) - \frac{1}{\mathrm{card}(\mathcal{C}_-)} \sum_{x_i \in \mathcal{C}_-} \left(\omega^{\star T} x_i + \omega_0^\star\right) = \sum_{j=1}^{n} \omega_j m_j^+ - \sum_{j=1}^{n} \omega_j m_j^- \tag{3-49}$$

$$s_j = |\omega_j^*|(m_j^+ - m_j^-) \tag{3-50}$$

Note that the SVM feature importance measure presented in Equation 3-50 is affected by correlation bias: features that belong to groups of correlated features receive smaller weights in the decision boundary of the linear SVM model because of their shared contribution to the prediction [33]. When presented with groups of informative correlated features, the SVM training algorithm will tend to assign a high weight $|\omega_j^*|$ to only one arbitrary representative of each group of correlated features. As a result, feature x^j will appear as being very important to the SVM model because of its high importance score s_j but features that are correlated with x^j will score poorly because they are redundant. The problem is that these feature will therefore be considered a lot less important than x^j, despite being very similarly associated to the classification target labels. Although these features are unimportant from a statistical point of view, they may contain useful biochemical information. Correlation bias may be very misleading in the context of biomarker discovery.

3-2-2 Nonlinear Support Vector Machines

The kernel trick enables the extension of the SVM classifier to non-separable data. A nonlinear mapping $\Phi : \mathcal{X} \to \mathcal{Z}$ is used to project the input data into a higher-dimensional kernel-space where the classes become linearly separable. Assuming $\dim(\mathcal{Z}) = N \gg n$, observations $x_i \in \mathbb{R}^n$, where $i = 1, 2...m$, are transformed into $z_i \in \mathbb{R}^N$ such that $z_i = \Phi(x_i)$. Performing SVM classification in the higher dimensional kernel-space $\mathcal{Z}$ yields nonlinear boundaries in feature-space $\mathcal{X}$.

A kernel is defined as a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ [6]. Kernels implicitly define the inner product of two observations in high dimensional kernel-space. In Equation 3-51, kernel K quantifies the similarity of x_i and x_j where $i, j = 1, 2...m$. Using a kernel is an efficient way of computing the inner product in high dimensional space: $O(N)$ computations would be necessary for explicitly computing $\langle \Phi(x_i), \Phi(x_j)$ in $\mathbb{R}^N$ whereas only $O(n)$ operations are necessary for computing $K(x_i, x_j)$ [6]. Note that the computational cost of nonlinear SVM is therefore independent of kernel-space dimensionality.

$$\forall x_i, x_j \in \mathcal{X}, \;\; K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \tag{3-51}$$

Since the dual formulation of the soft-margin SVM optimization problem and its resulting hypothesis depend directly on the inner product of observations, the SVM may be extended to accommodate nonlinear decision boundaries by replacing inner product $x_i^T x_j$ by kernel $K(x_i, x_j)$.

$$\begin{aligned} \text{maximize} \;\; \mathcal{L}(\lambda) = \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{m} \lambda_i \lambda_j y_i y_j K(x_i, x_j) \\ \text{subject to:} \;\; \sum_{i=1}^{m} \lambda_i y_i = 0 \\ 0 \leq \lambda_i \leq C \end{aligned} \tag{3-52}$$

Solving Equation 3-52 yields $\lambda^\star$ hence the bias $\omega_0^\star = y_k - \sum_{i=1}^{m} \lambda_i^\star y_i K(x_i, x_k)$ where x_k is a support vector lying on the margin. The optimization does not return the weight vector $\omega^\star$ defining the linear boundary in high dimensional kernel-space. This property of the kernel trick has the advantage of reducing computational cost. Unlike the hypothesis of linear SVMs, the nonlinear SVM hypothesis defined in Equation 3-53 can only be expressed as a function of its support vectors. The kernel $K(x_j, x_i)$ measures the similarity between test observation x_i and all m training observations x_j.

Since weights $\omega_1, \omega_2 ... \omega_N$ are not explicitly computed, the user cannot simply determine how individual features influence the classification outcome by comparing their respective weights. This problem is illustrative of the trade-off between interpretability and capacity in machine learning: by choosing to use support vector machines with a nonlinear kernel, we seem to improve classification performance at the expense of interpretability. Unlike linear SVMs, nonlinear SVMs do not provide the user with a ranking of features according to their importance. Nonlinear SVM classifiers are so-called "black-box" models because their decision-making process is largely opaque to the user. We resort to model-agnostic interpretability methods to explain the predictions of nonlinear SVMs and evaluate the predictive importance of their features.

$$\hat{y}_i = \operatorname{sgn}\left(\sum_{j=1}^{m} \lambda_j^\star y_j K(x_j, x_i) + \omega_0^\star \right) \tag{3-53}$$

The convexity of the SVM optimization problem and its convergence to a unique solution is guaranteed if and only if kernel K verifies Mercer's condition [6]. Any positive definite symmetric (PDS) kernel qualifies [6]. A PDS kernel has a symmetric positive semi-definite Gram matrix. The following kernels are frequently used for nonlinear SVM classification:

- The polynomial kernel K_P of degree d is defined over $\mathbb{R}^n$ for any constant $c > 0$. Note that K_P is PDS [68]. The dimensionality of the polynomial kernel-space is N.

$$N = \binom{n+d}{d} = \frac{(n+d)!}{n!d!}$$

$$\forall x_i, x_j \in \mathbb{R}^n, \;\; K_P(x_i, x_j) = \left(x_i^T x_j + c \right)^d \tag{3-54}$$

- The radial basis function (RBF) Gaussian kernel K_{RBF} defined over $\mathbb{R}^n$ for $\sigma \geq 0$. Hyper-parameter σ determines the width of the Gaussian kernel. Note that K_{RBF} is PDS [68]. The Gaussian-RBF kernel is an infinite-dimensional transform, hence $N = \infty$.

$$\forall x_i, x_j \in \mathbb{R}^n, \quad K_{RBF}(x_i, x_j) = \exp\left(-\frac{||x_j - x_i||^2}{2\sigma^2}\right) \tag{3-55}$$

- The hyperbolic tangent, or sigmoid kernel K_S defined over $\mathbb{R}^n$ for any real constants $a, b \geq 0$. Note that K_S is conditionally positive definite, but not positive definite symmetric (PDS) [68].

$$\forall x_j, x_j \in \mathbb{R}^n, \quad K_S(x_i, x_j) = \tanh\Big(a(x_i^T x_j) + b)\Big) \tag{3-56}$$

Although the choice of kernel is essential for ensuring the good performance of nonlinear SVM classifiers, there is currently no efficient method for selecting the best kernel for a given problem [66]. Once a type of kernel has been chosen, the kernel hyperparameters (i.e. c, d for K_P, a, b for K_S, and σ for K_{RBF}) must be tuned by validation. Changing the hyperparameters modifies kernel-space geometry, hence affecting the margin of the separating hyperplane. This may result in different support vectors and a different boundary in feature space. In practice, the Gaussian kernel is often preferred because one unique hyperparameter controls the model's complexity and flexibility [7].

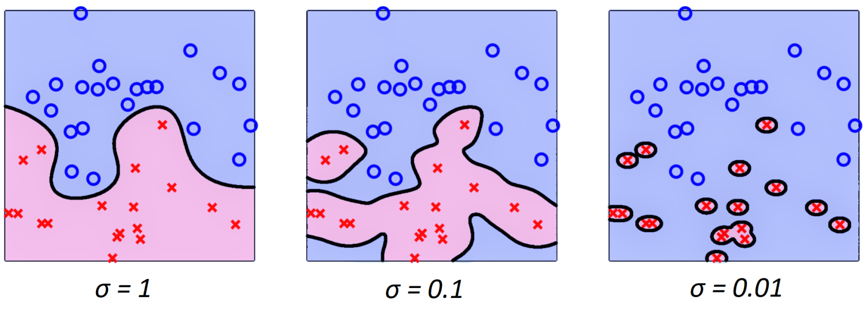


Figure 3-5: Nonlinear support vector machine classification using different Gaussian-RBF kernels: the SVM classifier becomes more prone to overfitting as its kernel becomes narrower. Figure adapted from [7]. Copyright © 2019 Yaser S. Abu-Mostafa, Malik Magdon-Ismail, Hsuan-Tien Lin.

Figure 3-5 demonstrates the need to carefully choose kernel hyperparameters: three Gaussian-RBF kernels with different values for σ are used for nonlinear SVM classification. A small σ makes the Gaussian kernel narrow and put the corresponding classifier at risk of overfitting (rightmost figure). A larger σ yields a wider Gaussian kernel and a more robust boundary will tend to generalize better (leftmost figure).

Figure 3-6 illustrates the decision boundaries obtained by nonlinear SVM classifiers with polynomial kernels. The red and blue classes are separated by a white margin, whose support vectors are boxed. The dimension of the second order polynomial kernel-space is $N = 5$ because $\Phi_2(x) = (x_1, x_2, x_1x_2, x_1^2, x_2^2)$. That of the third order polynomial kernel-space is $N = 9$ because $\Phi_3(x) = (x_1, x_2, x_1x_2, x_1^2, x_2^2, x_1^3, x_2^3, x_1^2x_2, x_2^2x_1)$. Despite the fact the kernel-space dimension near doubles from Φ_2 to Φ_3, the effective complexity of the classifier's decision

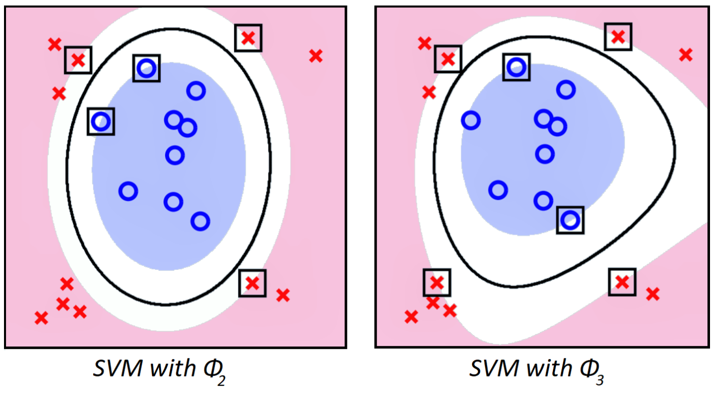


Figure 3-6: Nonlinear support vector machine classification using 2^{nd} and 3^{rd} order polynomial kernels: increasing the dimensionality of the kernel space does not dramatically increase the complexity of the SVM classifier's decision boundary. Figure adapted from [7]. Copyright © 2019 Yaser S. Abu-Mostafa, Malik Magdon-Ismail, Hsuan-Tien Lin.

boundary hardly changes since the number of support vectors increases from 5 to 6. Figure 3-6 illustrates a peculiar advantage of the nonlinear SVM classifier: increasing kernel-space dimensionality does not hinder generalization.

3-2-3 Sequential Minimal Optimization

Solving the SVM optimization problem by standard numerical quadratic programming (QP) is computationally very costly: assuming a dataset of m observations, a QP solver takes $O\left(m^3\right)$ operations and has memory requirements in the order of $O\left(m^2\right)$ [66]. Sequential minimal optimization (SMO) is an optimization algorithm introduced by Platt in 1998 to speed up the training of dual-form linear and nonlinear soft-margin SVM classifiers and to reduce memory requirements [6]. SMO decomposes the large QP optimization problem into a series of small quadratic optimization problems that each involve only two Lagrange multipliers and can therefore be solved analytically [6]. The two key elements of the SMO algorithm are the following: a heuristic method for choosing which two Lagrange multipliers to optimize at each step and an analytic method for solving these two Lagrange multipliers [8]. The algorithm is guaranteed to converge to a global optimum [8]. It stops when the Lagrange multipliers corresponding to all observations meet the KKT optimality conditions of Equation 3-46.

In order to explain the sequential minimal optimization (SMO) algorithm, we refer to subsection 3-2-1-2 about linear soft-margin SVMs. Note that Platt defines the bias $b = -\omega_0$. We briefly summarize the linear soft-margin SVM problem. The primal SVM problem is given by Equation 3-43 where penalty parameter $C \geq 0$ determines the trade-off between margin maximization and outlier-penalty minimization, and slack variables ξ_i measure the distance by which observations x_i violate the margins, for $i = 1, 2...m$. The Lagrangian is given by Equation 3-44 where $\lambda_i \geq 0$ and $\mu_i \geq 0$ are the Lagrange multipliers of constraints $y_i(\omega^T x_i - b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ respectively. The KKT conditions are given by Equation 3-45 and the dual problem is given by Equation 3-48. We focus on solving the dual problem.

The SMO algorithms considers the three following cases with bias $b = -\omega_0$ [9]:

- If x_i is on the right side of the margin, $\lambda_i = 0$, $\mu_i = C$ and $\xi_i = 0$ so $y_i(\omega^T x_i - b) - 1 \geq 0$
- If x_i is on the margin, $0 < \lambda_i < C$, $\mu_i = C - \lambda_i$ and $\xi_i = 0$ so $y_i(\omega^T x_i - b) - 1 = 0$
- If x_i is on the wrong side of the margin, $\lambda_i = C$, $\mu_i = 0$ and $\xi_i = 0$ so $y_i(\omega^T x_i - b) - 1 \leq 0$

$$E_i = g(x_i) - y_i = \omega^T x_i - b - y_i = \sum_{j=1}^{m} \lambda_j y_j x_j^T x_i - b - y_i \tag{3-57}$$

$$y_i(\omega^T x_i + \omega_0) - 1 = y_i(\omega^T x_i + \omega_0) - y_i^2 = y_i(\omega^T x_i + \omega_0 - y_i) = y_i E_i = R_i \tag{3-58}$$

Assuming E_i is the prediction error the SVM classifier makes on observation x_i, we can rewrite $y_i(\omega^T x_i + \omega_0) - 1$ as R_i. The SMO algorithm implements the KKT conditions given in Equation 3-59 [9]. The KKT conditions are violated either if $\lambda_i < C$ and $R_i < 0$ or if $\lambda_i > 0$ and $R_i > 0$ [9].

$$\begin{aligned} \lambda_i = 0 \;&\Rightarrow\; R_i \geq 0 \\ 0 < \lambda_i < C \;&\Rightarrow\; R_i \approx 0 \\ \lambda_i = C \;&\Rightarrow\; R_i \leq 0 \end{aligned} \tag{3-59}$$

3-2-3-1 Choosing two Lagrange Multipliers for Optimization

Wisely choosing which two Lagrange multipliers λ_1 and λ_2 to optimize is crucial for efficiently training the SVM classifier. Since there are $m(m-1)$ possibilities for λ_1 and λ_2 at each step, the larger the dataset, the more computationally expensive it becomes to try out all possible pairs before choosing the one that yields the largest increase in the objective function. Hence the importance of Platt's heuristic approach for choosing which two Lagrange multipliers to optimize per step. Platt's method is based on two loops: the outer-loop selects λ_2 and, for a given λ_2, the inner-loop selects λ_1 [8]. An observation must violate the KKT conditions to be eligible for optimization [8]. Each step of the SMO algorithm is guaranteed to increase the objective function provided one of the two Lagrange multipliers considered for optimization violates the KKT conditions [8]. The KKT conditions of Equation 3-59 are checked to be within ϵ of fulfillment. Platt recommends setting the KKT precision such that $10^{-4} \leq \epsilon \leq 10^{-3}$ [8].

Non-bound observations whose Lagrange multipliers verify $0 < \lambda_i < C$ are more likely to violate the KKT conditions than bound observations for whom $\lambda_i = 0$ or $\lambda_i = C$ [8]. Indeed, as the SMO algorithm progresses, the Lagrange multipliers of non-bound observations will change, unlike those of bound observations [8]. The SMO algorithm therefore dedicates most of its computational time to adjusting the Lagrange multipliers of the non-bound observations. Platt recommends maintaining and updating a cache for the error E_i of non-bound observations [9]. The error on observations whose Lagrange multipliers take boundary values are only computed when needed.

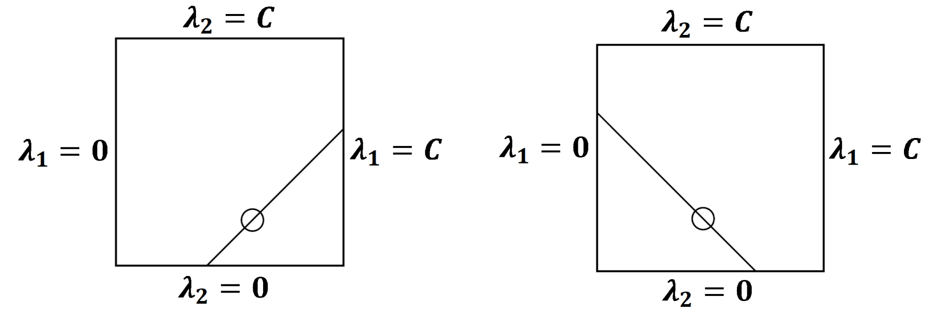
The outer-loop first passes through the entire dataset, then makes repeated passes over the non-bound observations until all the non-bound observations verify the KKT conditions, whereupon the algorithm terminates [8]. Given a λ_2 that violates the KKT conditions, the aim of the inner-loop is to choose a λ_1 that yields a large objective function increase during joint optimization[9]. The inner-loop starts by looking for a non-bound observation that maximizes $|E_2 - E_1|$ where E_1 and E_2 are computed as per Equation 3-57 [8]. If this first approach does not result in progress, SMO hierarchically resorts to the two following methods for choosing λ_1: SMO scans through the non-bound observations in search of a suitable Lagrange multiplier and, in case of failure, it scans through the entire dataset [9]. Both the iteration over non-bound observations and the iteration over all observations are randomly initialized to avoid biasing the SMO algorithm towards the beginning of the training dataset [8]. If none of these three methods for choosing λ_1 succeed, SMO skips the current λ_2 and continues with another choice of λ_2.

3-2-3-2 Optimizing the two chosen Lagrange Multipliers

The Lagrange multipliers λ_i, for $i = 1, 2...m$ are initialized at zero. Without loss of generality, suppose we are optimizing λ_1, λ_2 from a previous set of feasible solutions $\lambda_1^{\text{old}}, \lambda_2^{\text{old}}, \lambda_3...\lambda_m$. The bound constraint $0 \leq \lambda_i \leq C$ causes both λ_1 and λ_2 to lie within a box [8]. The linear equality constraint $\sum_{i=1}^{m} \lambda_i y_i = 0 \iff y_1\lambda_1 + y_2\lambda_2 = y_1\lambda_1^{\text{old}} + y_2\lambda_2^{\text{old}}$ causes λ_1 and λ_2 to lie on a line [8]. Hence the constrained minimum of the objective function must lie on a diagonal line segment as shown in Figure 3-7. Let $s = y_1 y_2$ and $\gamma = \lambda_1^{\text{old}} + s\lambda_2^{\text{old}}$. Note that the feasible line segment depends on whether the class labels of observations x_1 and x_2 are equal.

The objective function $\mathcal{L}(\lambda)$ of Equation 3-48 is rewritten as a function of λ_2 in Equation 3-60 [9]. We consider the following entries of the Gram matrix: $K_{11} = x_1^T x_1$, $K_{22} = x_2^T x_2$ and $K_{12} = x_1^T x_2$.

$$
\begin{aligned}
\mathcal{L}(\lambda) &= -\frac{1}{2}\left(y_1y_1x_1^Tx_1\lambda_1^2 + y_2y_2x_2^Tx_2\lambda_2^2 + 2y_1y_2x_1^Tx_2\lambda_1\lambda_2 + 2\left(\sum_{i=3}^{m}\lambda_iy_ix_i^T\right)(y_1x_1\lambda_1 + y_2x_2\lambda_2)\right) \\
&\quad +\lambda_1 + \lambda_2 \\
\mathcal{L}(\lambda) &= -\frac{1}{2}\left(K_{11}\lambda_1^2 + K_{22}\lambda_2^2 + 2sK_{12}\lambda_1\lambda_2 + 2y_1\lambda_1\sum_{i=3}^{m}\lambda_iy_ix_i^Tx_1 + 2y_2\lambda_2\sum_{i=3}^{m}\lambda_iy_ix_i^Tx_2\right) \\
&\quad +\lambda_1 + \lambda_2 \\
\mathcal{L}(\lambda) &= (1-s)\lambda_2 + sK_{11}\gamma\lambda_2 - \frac{1}{2}K_{11}\lambda_2^2 - \frac{1}{2}K_{22}\lambda_2^2 - sK_{12}\gamma\lambda_2 + K_{12}\lambda_2^2 \\
&\quad +y_2\lambda_2\left(\sum_{i=3}^{m}\lambda_iy_ix_i^Tx_1 - \sum_{i=3}^{m}\lambda_iy_ix_i^Tx_2\right) \\
\mathcal{L}(\lambda) &= \left(1 - s + sK_{11}\gamma - sK_{12}\gamma + y_2\sum_{i=3}^{m}\lambda_iy_ix_i^Tx_1 - y_2\sum_{i=3}^{m}\lambda_iy_ix_i^Tx_2\right)\lambda_2 \\
&\quad +\frac{1}{2}\left(2K_{12} - K_{11} - K_{22}\right)\lambda_2^2
\end{aligned}
\tag{3-60}
$$



(a) If $y_1 \neq y_2$, $\lambda_1 - \lambda_2 = \gamma$ **(b)** If $y_1 = y_2$, $\lambda_1 + \lambda_2 = \gamma$

Figure 3-7: Feasible region of Lagrange multipliers λ_1 and λ_2 during one SMO step. Figure adapted from [8].

Let $\eta = 2K_{12} - K_{11} - K_{12}$. The objective function, it's first and second order derivatives can be rewritten as per Equations 3-61, 3-62 and 3-63 respectively [9].

$$
\mathcal{L}(\lambda) = \frac{1}{2}\eta\lambda_2^2 + \left(y_2\left(E_1^{old} - E_2^{old}\right) - \eta\lambda_2^{old}\right)\lambda_2 + \text{const} \tag{3-61}
$$

$$
\frac{\partial\mathcal{L}}{\partial\lambda_2} = \eta\lambda_2^2 + \left(y_2\left(E_1^{old} - E_2^{old}\right) - \eta\lambda_2^{old}\right) \tag{3-62}
$$

$$
\frac{\partial^2\mathcal{L}}{\partial\lambda_2^2} = \eta \tag{3-63}
$$

The updated value of Lagrange multiplier λ_2 is obtained by setting the first order derivative to zero, hence λ_2^{new} defined in Equation 3-64 [9]:

$$\lambda_2^{new} = \lambda_2^{old} + \frac{y_2(E_2^{old} - E_1^{old})}{\eta} \tag{3-64}$$

If $\eta < 0$, the above expression is the unconstrained maximum of λ_2. It must be checked against the feasible range, which is determined as follows [9]. Figure 3-8 corresponds to the case $y_1 = y_2 \iff s = 1$, whereas Figure 3-9 corresponds to the case $y_1 \neq y_2 \iff s = -1$ [9]. Let the minimum feasible value of λ_2 be L, and its maximum be H.

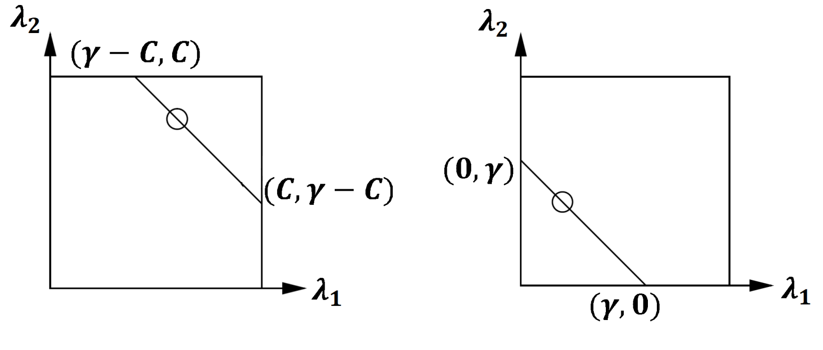
- If $s = 1$, then $\gamma = \lambda_1^{old} + \lambda_2^{old}$.
  - If $\gamma > C$, then $\min(\lambda_2^{new}) = \gamma - C$ and $\max(\lambda_2^{new}) = C$
  - If $\gamma < C$, then $\min(\lambda_2^{new}) = 0$ and $\max(\lambda_2^{new}) = \gamma$
- If $s = -1$, then $\gamma = \lambda_1^{old} - \lambda_2^{old}$.
  - If $\gamma > 0$, then $\min(\lambda_2^{new}) = 0$ and $\max(\lambda_2^{new}) = C - \gamma$
  - If $\gamma < 0$, then $\min(\lambda_2^{new}) = -\gamma$ and $\max(\lambda_2^{new}) = C$

Lagrange multiplier λ_2 is clipped if it exceeds its feasible bounds as per Equation 3-65 [9]. As a result, $\lambda_2^{new,clip}$ will belong to the allowed range $[L, H]$.

$$\lambda_2^{new,clip} = \begin{cases} H \text{ if } \lambda_2^{new} \geq H \\ \lambda_2^{new} \text{ if } L < \lambda_2^{new} < H \\ L \text{ if } \lambda_2^{new} \leq L \end{cases} \tag{3-65}$$

Now, Lagrange multiplier λ_1 is updated as per Equation 3-66 [8].

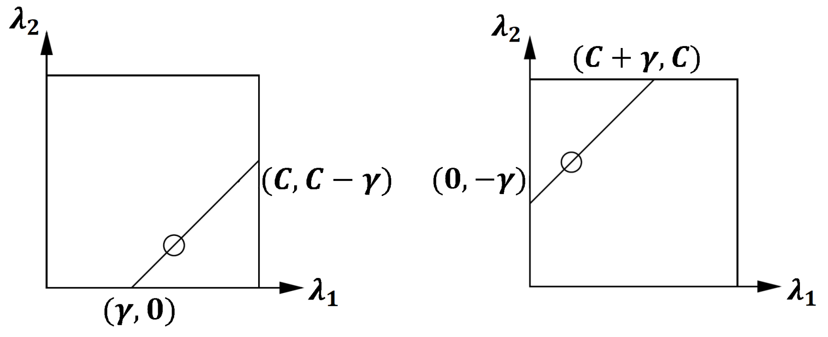
$$\lambda_1^{new} = \lambda_1 + s\left(\lambda_2 - \lambda_2^{new,clip}\right) \tag{3-66}$$



(a) Case $y_1 = y_2$: $\lambda_1^{old} + \lambda_2^{old} = \gamma$, $\gamma > C$ **(b)** Case $y_1 = y_2$: $\lambda_1^{old} + \lambda_2^{old}$, $\gamma < C$

Figure 3-8: Feasible range assuming equal class labels hence $s = 1$:
$L = \max\left(0, \lambda_2^{old} + \lambda_1^{old} - C\right)$ and $H = \min\left(C, \lambda_2^{old} + \lambda_1^{old}\right)$. Figure adapted from [9].

The objective function's second order derivative η may not be negative, in which case we need to evaluate the objective function Ψ at the two endpoints Ψ_L and Ψ_H defined in Equation



(a) Case $y_1 \neq y_2$: $\lambda_1^{old} - \lambda_2^{old} = \gamma$, $\gamma > 0$ **(b)** Case $y_1 \neq y_2$: $\lambda_1^{old} - \lambda_2^{old} = \gamma$, $\gamma < 0$

Figure 3-9: Feasible range assuming different class labels hence $s = -1$:
$L = \max\left(0, \lambda_2^{old} - \lambda_1^{old}\right)$ and $H = \min\left(C, C + \lambda_2^{old} - \lambda_1^{old}\right)$. Figure adapted from [9].

3-67, and set λ_2^{new} to be the one with the larger objective function value [9]. Therefore, if $\Psi_L < \Psi_H$, we have $\lambda_2^{new} = L$. However, if $\Psi_L > \Psi_H$, $\lambda_2^{new} = H$. Otherwise, λ_2 does not change and the optimization step is deemed unsuccessful.

$$f_1 = y_1(E_1 + b) - \lambda_1 K(x_1, x_1) - s\lambda_2 K(x_2, x_2)$$

$$f_2 = y_2(E_2 + b) - s\lambda_1 K(x_1, x_2) - \lambda_2 K(x_2, x_2)$$

$$L_1 = \lambda_1 + s(\lambda_2 - L)$$

$$H_1 = \lambda_1 + s(\lambda_2 - H)$$

$$\begin{aligned} \Psi_L &= L_1 f_1 + L f_2 + \frac{1}{2} L_1^2 K(x_1, x_1) + \frac{1}{2} L^2 K(x_2, x_2) + s L L_1 K(x_1, x_2) \\ \Psi_H &= H_1 f_1 + H f_2 + \frac{1}{2} H_1^2 K(x_1, x_1) + \frac{1}{2} H^2 K(x_2, x_2) + s H H_1 K(x_1, x_2) \end{aligned} \tag{3-67}$$

The separating hyperplane's bias b is updated after each successful optimization step to guarantee both observations x_1 and x_2 verify the KKT conditions [8]. Threshold b_1, defined in Equation 3-68, is valid when $0 < \lambda_1^{new} < C$ because it forces the output of the SVM to be y_1 when the input is x_1 [8]. Threshold b_2, defined in Equation 3-69, is valid when $0 < \lambda_2^{new} < C$ because it forces the output of the SVM to be y_2 when the input is x_2 [8]. If both $0 < \lambda_1^{new} < C$ and $0 < \lambda_2^{new} < C$, both b_1 and b_2 are valid and they will be equal. If both Lagrange multipliers are at the bounds (i.e. $\lambda_1 = 0$ or $\lambda_1 = C$ and $\lambda_2 = 0$ or $\lambda_2 = C$) then all the thresholds between b_1 and b_2 verify the KKT conditions [8].

Platt recommends setting the new threshold to the mean of b_1 and b_2 [8]. Hence the new bias b^{new} defined in Equation 3-70.

$$b_1 = E_1 + y_1(\lambda_1^{new} - \lambda)K(x_1, x_1) + y_2(\lambda_2^{new,clip} - \lambda_2)K(x_1, x_2) + b^{old} \tag{3-68}$$

$$b_2 = E_2 + y_1(\lambda_1^{new} - \lambda_1)K(x_1, x_2) + y_2(\lambda_2^{new,clip} - \lambda_2)K(x_2, x_2) + b^{old} \tag{3-69}$$

$$b^{new} = \begin{cases} b_1 \text{ if } 0 < \lambda_1 < C \\ b_2 \text{ if } 0 < \lambda_2 < C \\ \frac{b_1 + b_2}{2} \text{ otherwise} \end{cases} \tag{3-70}$$

The cached error on the non-bound observations is also updated after each successful optimization step: Equation 3-71 assumes the Lagrange multiplier of observation x_j verifies $\lambda_j \neq 0$ and $\lambda_j \neq C$. The errors E_1 and E_2 corresponding to the optimized Lagrange multipliers λ_1 and λ_2 are set to zero.

$$E_j = E_j + \left(\lambda_1^{new} - \lambda_1^{old}\right) y_1 x_1^T x_j + \left(\lambda_2^{new} - \lambda_2^{old}\right) y_2 x_2^T x_j - \left(b^{new} - b^{old}\right) \tag{3-71}$$

3-3 Random Forests

3-3-1 Decision Trees

Tree-based supervised machine learning methods owe their success to the fact that decision trees are nonlinear, non-parametric, easily interpretable models that intrinsically perform feature selection. The methodology for building classification and regression trees (CART) was first developed by Breiman in 1984 [69]. Decision trees recursively partition the feature space into hyper-rectangular regions. The splitting rules necessary to partition the feature space can be represented in the form of an upside-down tree-like directed graph whose initial node is called the root, and whose terminal nodes are called the leaves [70]. We focus on binary trees: the root and the internal nodes have two outgoing edges and two child nodes each.

A decision tree is defined by a model $\phi : \mathcal{X} \rightarrow \mathcal{Y}$ where $\mathcal{X}$ and $\mathcal{Y}$ are the input and output spaces respectively. Assuming a binary classification problem, decision trees partition the feature space into hyper-rectangles by applying a sequence of binary decisions to individual features [69]. Note that, in section 3.6, each feature is noted $x^j = X(:, j)$ where $j = 1, 2...n$. Recursive binary splitting is a top-down, greedy approach to partitioning the feature space [70]. It starts from the root of the decision tree and then successively splits the feature space into regions until a stopping criterion is met. Each internal node t represents a partition of the input space $\mathcal{X}_t \in \mathcal{X}$. Node t divides subspace $\mathcal{X}_t$ into two disjoint subspaces $\mathcal{X}_{t,L}$ and $\mathcal{X}_{t,R}$ depending on whether observations $x_i \in \mathcal{X}_t$ verify $x^j \leq s_t$ or $x^j > s_t$ respectively. Subspaces $\mathcal{X}_{tL}$ and $\mathcal{X}_{tR}$ verify $\mathcal{X}_{tL} \cap \mathcal{X}_{tR} = \emptyset$ and $\mathcal{X}_{tL} \cup \mathcal{X}_{tR} = \mathcal{X}_t$. Nodes are sometimes referred to as weak learners [71]. We choose to use axis-aligned weak learners, also called decision stumps, whose decision boundaries are aligned with one of the axes of the feature space [71]. Recursive binary splitting consists in selecting the feature x^j and threshold s_t such that partitioning $\mathcal{X}_t$ into the regions $\mathcal{X}_{tL}$ and $\mathcal{X}_{tR}$ leads to the maximum reduction of the cost function [70].

$$\mathcal{X}_{tL} = \{x_i | x^j \leq s_t\} \qquad \mathcal{X}_{tR} = \{x_i | x^j > s_t\} \tag{3-72}$$

The cost function $i(t)$ evaluated at each node t is a measure of node impurity. The most common node impurity metrics are based on the Gini index and the Shannon entropy [69]. The Gini-based impurity $i_G(t)$ and the entropy-based impurity $i_H(t)$ are defined in Equation 3-74 and 3-75 respectively. Estimated probability $p(c|t)$ is the proportion of observations in subset $\mathcal{X}_t$ that belong to class c. In Equation 3-73, $m_t = \operatorname{card}(\mathcal{X}_t)$ and m_t^c is the number of observations in $\mathcal{X}_t$ belonging to c. Node impurity is maximum if there are equally many observations belonging to different classes within $\mathcal{X}_t$. Conversely, node impurity is minimum if all observations $x_i \in \mathcal{X}_t$ belong to just one class. Hence the purer the node, the smaller the classification error and the better the split [69].

$$p(c|t) = p(y_i = c | x_i \in \mathcal{X}_t) = \frac{m_t^c}{m_t} \tag{3-73}$$

$$i_G(t) = \sum_c p(c|t)(1 - p(c|t)) \tag{3-74}$$

$$i_H(t) = -\sum_c p(c|t) \log_2(p(c|t)) \tag{3-75}$$

The impurity decrease $\Delta_i(s_t, t)$ of a binary split s_t dividing node t into a left node t_L and a right node t_R is therefore given by Equation 3-76 where $m_{t,L}$ and $m_{t,R}$ are the number of observations going from node t to t_L and t_R respectively. In other words, $m_{t,L} = \text{card}\,(\mathcal{X}_{tL})$ and $m_{t,R} = \text{card}\,(\mathcal{X}_{tR})$. Figure 3-10 illustrates how node t is split into two children node by performing a test on feature x^j of all observations x_i, for $i = 1, 2...m$.

$$\Delta_i(s_t, t) = i(t) - \frac{m_{t,L}}{m_t} i(t_L) - \frac{m_{t,R}}{m_t} i(t_R) \tag{3-76}$$

Decision trees are built by iteratively splitting nodes into purer nodes. Greedily growing decision trees implies solving a local optimization problem at each node. That is to say, the recursive binary splitting algorithm does not look ahead of the current node. For each node t, subset $\mathcal{X}_t$ is divided using the threshold s that locally maximizes the decrease in overall impurity $\Delta_i(s_t, t)$ of the child nodes with respect to the parent node [69]. The rationale behind the greedy assumption is twofold: to limit computational costs and to favor simple and small solutions that are easy to interpret and generalize well [69].

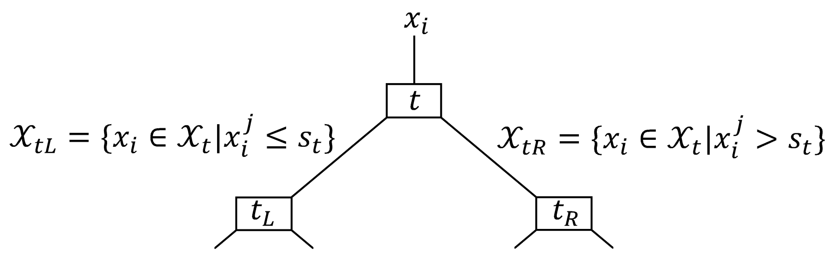

x_i
t
$\mathcal{X}_{tL} = \{x_i \in \mathcal{X}_t | x_i^j \leq s_t\}$
$\mathcal{X}_{tR} = \{x_i \in \mathcal{X}_t | x_i^j > s_t\}$
t_L
t_R

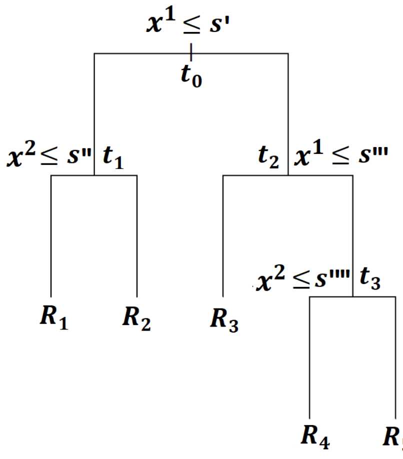
Figure 3-10: Diagram of a decision tree's node t being split into a left child node t_L and a right child node t_R by performing a binary split, with a threshold s_t, on feature x^j

The stopping criterion is a heuristic method to determine when to stop splitting a node and declare it as a leaf of the decision tree. Determining an adequate stopping criterion for the problem under study is important to avoid overfitting [70]. Not growing full trees has repeatedly been demonstrated to have positive effects in terms of generalization [71]. Different hyper-parameters may be used as stopping criteria: the most common approach consists of setting a maximum allowed tree depth D, in which case we make any node whose depth (i.e. number of edges from the root to the node) is D a leaf [70]. Another approach is to define a minimum splitsize m_{min} that stops the further splitting of any node whose number of observations (i.e. cardinality) is less than the minimum splitsize. Yet another approach consists in declaring a node a leaf if the decrease in impurity at that node is less than a user-defined threshold $\Delta_{i,\text{min}}$ [69].

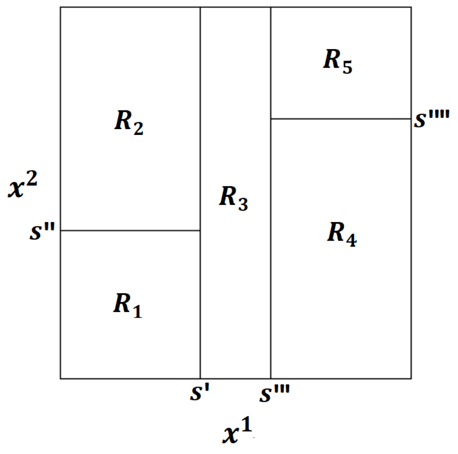
At the decision tree's terminal nodes $t^\star$, we consider the subset $\mathcal{X}_{t^\star}$. The estimated probability of class c is defined as the proportion of training observation in $\mathcal{X}_{t^\star}$ that belong to class c: $p(c|t^\star) = p(y_i = c|x_i \in \mathcal{X}_{t^\star}) = m^c_{t^\star}/m_{t^\star}$ where $m_{t^\star}$ is the size of terminal node $t^\star$. The decision tree's terminal nodes are labeled as per Equation 3-77. The class label assigned to a given terminal node $t^\star$ is obtained by majority vote of the training observations that fall into $\mathcal{X}_{t^\star}$: $\hat{y}_i(t^\star) = c^\star$ is the class to which most observations in $\mathcal{X}_t$ belong.

$$\hat{y}_i(t^\star) = \underset{c \in \mathcal{Y}}{\arg\max}\, p(c|t^\star) \tag{3-77}$$

Predictions are made by propagating a test observation x_i through the decision tree and assigning it the class label characterizing the terminal node $t^\star$ in which x_i falls: $\forall x_i \in \mathcal{X}_{t^\star}, \phi(x_i) = \hat{y}_i(t^\star)$. Classification of new observations is done by assignment to the most commonly occurring class in the feature space partition to which the observation belongs.



(a) Binary decision tree



(b) Partition of two dimensional feature space

Figure 3-11: Recursive binary decision tree and two-dimensional feature space partition

A simple example of recursive binary splitting is illustrated in Figure 3-11. The root t_0 partitions the two-dimensional feature space $\mathcal{X}$ into left and right subspaces $\mathcal{X}_{t_0,L}$ and $\mathcal{X}_{t_0,R}$ depending on whether $x^1 \leq s'$ or $x^1 > s'$. The process is repeated for internal nodes t_1, t_2, t_3: we look for the feature (either x^1 or x^2) and the threshold that minimize the cost function at each split of the previously obtained subspaces. For example, at node t_2, subspace $\mathcal{X}_{t_2} = \mathcal{X}_{t_0,R}$ is partitioned depending on whether $x^1 \leq s'''$ or $x^1 > s'''$, yielding subspaces $\mathcal{X}_{t_2,L} = R_3$ and $\mathcal{X}_{t_1,R} = \mathcal{X}_{t_3}$. The leaves of the decision tree are the regions R_1, R_2, R_3, R_4 and R_5. Once each of these five leaves are either assigned to the positive or negative class, the predicted response for a given test observation is determined by the majority vote of the training observations in the region to which the test observation is assigned.

The aim of supervised machine learning is not only to accurately model the response, but also to identify which of the input variables are most important in making the predictions [69]. The graphical representation of decision trees enables the user to easily visualize the process: the influence of a feature on the output directly corresponds to its position in the tree [72]. Intuitively, features that appear at the top of the tree or those that appear at multiple nodes will be more important than features appearing at the bottom of the tree or not at all.

Using decision trees for classification has two noteworthy weaknesses: average predictive performance and model instability [69]. Indeed, the performance of trees suffers when the output cannot be defined using hyper-rectangular partitions of the feature space - a situation which is frequent in practice [69]. Furthermore, decision trees are low bias, high variance classifiers whose structure, and performance, strongly depend on the training data they are built upon: a small change in the training set will produce a radically different tree [70]. The propensity of decision trees for overfitting implies they are robust to neither noise nor outliers, and do not generalize well. A simple and efficient solution to this problem is to use decision trees in the context of randomization-based ensemble methods [73]. The idea is to introduce random perturbations into the learning process in order to produce several different decision trees from a single training dataset and to use some aggregation technique to combine the predictions of all these trees [73].

3-3-2 From Bagging to Random Forests

Bagging is an ensemble method for improving the predictive performance of unstable machine learning algorithms. Originally proposed by Breiman in 1996 [72], bagging stands for bootstrap aggregation. Assume a training set $\mathcal{L}$ of m pairs of input observations and output class labels $(x_1, y_1), (x_2, y_2)...(x_m, y_m)$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. A bootstrapped dataset $\mathcal{L}^k$, where $k = 1, 2...B$, is one created by randomly drawing m observations from the training set with replacement [70]. Sampling with replacement means that when an observation is copied to the bootstrap set, it is not removed from the training set and is considered as a candidate in the next sampling. As a result, duplicates will appear within a bootstrapped set and some observations will be left out. The left-out observations $\mathcal{L} \backslash \mathcal{L}^k$ are called the out-of-bag (OOB) observations of the k^{th} bootstrapped set [70]. Note that although card $\left(\mathcal{L}^k\right) = \text{card}\left(\mathcal{L}\right) = m$, 37% of observations from $\mathcal{L}$ are on average missing in each of the bootstrap replicates [69]. Indeed, Equation 3-78 states that the probability of never been selected after m draws with replacement is 37%.

$$\left(1 - \frac{1}{m}\right)^m \approx \frac{1}{e} \approx 0.37 \tag{3-78}$$

Repeating this sampling process B times yields B bootstrap sets, which are treated as independent datasets. Bagging uses these B bootstrapped versions of the training dataset to build B different component classifiers ϕ_k where $k = 1, 2...B$. Assuming we chose the component classifiers to be decision trees, one tree is grown per bootstrap set and the ensemble of decision trees is called a forest. Each component classifier generates a prediction for a test observation and a majority vote determines the bagged classifier's prediction [72]. As demonstrated by Equation 3-79, the overall prediction $\Phi(x_i)$ is the most commonly predicted class for observation x_i among the B predictors. For component classifiers that suffer from high variance, such as decision trees, aggregating over multiple bootstrap sets reduces variance and stabilizes the prediction [72].

$$\hat{y}_i = \Phi(x_i) = \underset{c \in \mathcal{Y}}{\arg\max} \sum_{k=1}^{B} I(\phi_k(x_i) = c) \tag{3-79}$$

After Breiman introduced bagging, several authors, namely Dietterich (1998), Amit and Geman (1997), tweaked the algorithm by adding randomness to the learning process in order to decorrelate the decision trees and reduce variance [74]. The underlying rationale is to avoid the presence of a few dominant features in the data leading to several similarly structured, hence highly correlated, trees. The problem is that aggregating highly correlated component classifiers seriously limits the variance reduction achieved by the ensemble [70]. Based on the work of Dietterich, Amit and Geman, Breiman developed in 2001 the random forest (RF) algorithm [74]. The difference with bagging lies in the way the decision trees are grown: unlike CART, the feature to split in each node is selected as the best among a set of randomly chosen features. By implementing bagging in tandem with random feature selection, the RF algorithm operates a two-step randomization procedure. In addition to the randomization inherent to growing the decision trees on bootstrapped version of the training data, the node-splitting is also randomized [75]. Injecting randomness into the tree-growing process has been proven to decorrelate the decision trees making up the random forest, resulting in improved robustness, computational efficiency and predictive accuracy [74].

The RF algorithm requires the two following user-defined hyperparameters, in addition to the stopping criteria of its decision trees (either a maximum allowed tree depth D, or a minimum allowed splitsize m_{min}, or a minimum decrease in impurity $\Delta_{i,\text{min}}$).

- The number $n^\star$ of randomly selected features considered to split each tree's nodes. Random feature selection performs the following: at each node of each decision tree, a random subset of $n^\star < n$ features is selected and the one yielding the maximum decrease in impurity is chosen for the split [69]. When selecting the best split, the RF algorithm is therefore only allowed to consider these $n^\star$ features as splitting candidates. A new subset of $n^\star$ features is taken at each new split of the tree. Usually, $n^\star$ is chosen to be approximately equal to the square root of the total number of features: $n^\star \approx \sqrt{n}$ [76]. Building a random forest such that $n^\star = n$ would be equivalent to bagging.

- The number B of decision trees making up the forest. As in bagging, the RF algorithm grows one decision tree per bootstrapped version of the training dataset. Since random forests are not prone to overfitting, choosing a large B will not adversely affected the classifier's generalization performance [72]. A large B will however increase computational costs and memory requirements. It is worth mentioning that the RF algorithm is easy to parallelize since the trees corresponding to the different bootstrapped sets are independent from each other [72]: they can be grown separately and then aggregated to vote on the final prediction.

- The maximum tree depth D, or any other stopping criterion that limits the depth of decision trees (i.e. minimum splitsize m_{min} or minimum decrease in impurity $\Delta_{i,\text{min}}$). Although Breiman argued that in an ensemble each individual tree should be grown as deep as possible, Stobl recommends limiting the tree depth when dealing with datasets that have many more observations than features [77]. On the other hand, there are more features than observations, Strobl is in favor of growing deep trees [77].

A key advantage of the RF algorithm is that it provides an internal estimate of predictive performance. Unlike the supervised learning algorithms studied previously, the generalization error of a random forest can be estimated without resorting to validation. We consider the k^{th} decision tree ϕ_k and its corresponding bootstrapped set $\mathcal{L}^k$. As demonstrated by Equation 3-78, 37% of all observations are left out of $\mathcal{L}^k$. Since these OOB observations were not used to grow tree ϕ_k, they can be used to asses its generalization error [72]. Hence every tree in the forest generates an error estimate using its respective OOB observations. The average of all OOB metrics, referred to as the OOB error estimate $\hat{E}_{\text{OOB}}$, is used to gauge the predictive performance of the ensemble classifier [72]. In practice, $\hat{E}_{\text{OOB}}$ is a computationally efficient alternative to k-fold cross-validation [69].

$$\hat{E}_{\text{OOB}} = \frac{1}{m} \sum_{(x_i, y_i) \in \mathcal{L}} L \left(\underset{c \in \mathcal{Y}}{\arg\max} \sum_{l=1}^{B^{-i}} I(\phi_l(x_i) = c) \right) \quad \text{where} \quad L(\hat{y}_i) = \begin{cases} 0 & \text{if } \hat{y}_i = y_i \\ 1 & \text{if } \hat{y}_i \neq y_i \end{cases} \tag{3-80}$$

In Equation 3-80, the class label y_i of the i^{th} observation is predicted by majority vote of all B^{-i} trees grown from bootstrapped replicates that exclude (x_i, y_i). The trees for which x_i is OOB are denoted ϕ_l, where $l = 1, 2...B^{-i}$. The classification error is computed using the zero-one loss function L: L assigns a loss of zero for a correct classification and a loss of one for an incorrect classification. Finally, the OOB error estimate $\hat{E}_{\text{OOB}}$ is the average of the OOB errors of all m observations in $\mathcal{L}$.

3-3-3 Measuring Feature Importance

Improving classification accuracy by aggregating randomized decision trees comes at the expense of a loss in interpretability: unlike individual trees, random forests do not have a simple graphical representation. In order to understand which of the features have the most influence on the random forest's prediction, Breiman developed the mean decrease impurity (MDI) approach to measuring feature importance in RF classifiers [74]. The MDI importance of a feature x^j, for $j = 1, 2...n$, measures how effectively the feature reduces uncertainty. MDI(x^j) is defined as the total decrease in node impurity achieved by splitting on x^j, averaged over all decision trees making up the random forest. At each split of each tree, the improvement in node purity, weighted by the probability of an observation reaching that node, is the importance attributed to the splitting variable [78]. When using the Gini index $i_G(t)$ - as defined in Equation 3-74 - to quantify node impurity, MDI feature importance is referred to as the Gini importance [69].

The MDI importance of feature x^j is evaluated by summing the weighted impurity decreases $p(t)\Delta_i(s_t, t)$ for all nodes t that are split using x^j, averaged over all trees ϕ_k, where $k = 1, 2...B$, in the random forest [69]. In Equation 3-81, the probability $p(t)$ of reaching node t is approximated by the proportion m_t/m of observations reaching node t [69]. Index j_t denotes the variable used for splitting node t, s_t denotes the threshold of the split and I is the indicator function such that $I(j_t = j) = 1$ if $j_t = j$ and $I(j_t = j) = 0$ otherwise.

$$\text{MDI}\left(x^j\right) = \frac{1}{B} \sum_{k=1}^{B} \sum_{t \in \phi_k} I\left(j_t = j\right) \left(\frac{m_t}{m} \Delta_i(s_t, t) \right) \tag{3-81}$$

MDI tends to overestimate the importance of categorical features with high cardinality (i.e. features that take on a larger number of different values) [79]. The so-called variable selection bias stems from an unfair advantage given to high cardinality features that is inherent to the node impurity reduction algorithm: according to Breiman, "variable selection is biased in favor of those variables having more values and thus offering more splits" [77]. Low cardinality features are therefore less likely to be selected for splitting tree nodes because they are offer fewer potential cut-points [69]. When studying categorical variables, the variable selection bias may cause MDI to produce misleading feature rankings [77]. However, as long as only continuous predictor variables, or only categorical variables with the same cardinality are considered, Strobl claims that the MDI feature importance measure of random forests is not affected by variable selection bias [77]. Variable selection bias should therefore not be a problem when analyzing IMS data.

The advantage of MDI is that it covers the impact of each feature individually as well as in multivariate interactions with other predictor variables [77]. MDI therefore verifies our definition of feature importance. It is however important to note that MDI is severely affected by correlation between features [69] and that correlation bias is problematic for the analysis of IMS data. For example, MDI risks overestimating the importance of uninformative features that are highly correlated with informative features: Strobl reported that, when growing CART, a feature that is only weakly associated with the class label, but is highly correlated with another influential feature, may appear equally well suited for node splitting as the truly influential predictor variable [77]. MDI is also prone to underestimating the importance of equally informative features that are correlated among themselves because once one of these features has been selected for node splitting, the others become redundant and loose importance.

3-4 Post-hoc Model-agnostic Interpretability Methods

3-4-1 Permutation Importance

Permutation importance is a global post-hoc model-agnostic interpretability method. The permutation importance (PI) of a feature is the average decrease in model accuracy when its values are randomly permuted [80]. Permutation importance was originally developed by Breiman, under the name of mean decrease accuracy, for measuring feature importance in random forests [74]. PI measures the degree to which a trained supervised machine learning model relies on a specific feature [81]. Randomly permuting the values of a predictor variable across all data instances is supposed to mimic the absence of the variable from the model [82]. The underlying rationale is the following: randomly permuting x^j will break its association with the model prediction y and effectively cancel its predictive power. Therefore, if feature x^j is strongly associated to y, permuting its values should result in a large drop in predictive performance [69]. Conversely, if x^j is weakly associated to y, permuting it should have little to no impact on performance. Two alternative definitions of PI are given by Equation 3-82 and 3-83, where $\hat{E}_{ref}$ and $\hat{E}_j$ are the estimated classification error, before and after random permutation of variable x^j. The more x^j is important, the larger the drop in classification performance due to permuting x^j, hence $\hat{E}_{ref} < \hat{E}_j$ [79]. Although using the ratio, rather than the difference, is considered better for comparing different models [81], it risks yielding numerically unstable estimations when $\hat{E}_{ref} \approx 0$ [83]. We therefore prefer implementing PI as defined in Equation 3-82. Since PI quantifies each feature's influence on the classifier's prediction by considering both its individual impact and its interrelations with other features, PI verifies our definition of feature importance, as per subsection 1-2-2.

$$\mathrm{PI}\left(x^j\right) = \hat{E}_j - \hat{E}_{ref} \tag{3-82}$$

$$\mathrm{PI}\left(x^j\right) = \frac{\hat{E}_j}{\hat{E}_{ref}} \tag{3-83}$$

The problem with permutation importance is that it severely overestimates the importance of correlated features [77]. Ranking features with respect to their PI score may therefore not be reliable when analyzing IMS data. When investigating how correlation bias affects PI of random forests, Strobl observed that features that appear important may actually be independent of the prediction when considered conditional on another variable [77]. According to Strobl, correlation bias is due to the fact that the null hypothesis of PI assumes the predictor variables to be independent from each other. The original permutation importance, where one predictor variable x^j is permuted against both the response y and the remaining predictor variables $X_{\setminus\{j\}} = x^1, x^2 ... x^{j-1}, x^{j+1}, ..., x^n$, corresponds to a null hypothesis of independence between x^j and both y and $X_{\setminus\{j\}}$ [82]:

$$H_0 : x^j \perp y, X_{\setminus\{j\}} \iff x^j \perp y \wedge x^j \perp X_{\setminus\{j\}} \tag{3-84}$$

The reason why correlated features are preferred by PI is that a positive PI importance score corresponds to a deviation from the null hypothesis of Equation 3-84 that can be caused by a violation of either the independence of x^j and y, or of the independence of x^j and $X_{\setminus\{j\}}$ [82]. The latter violation of H_0 is not of interest: whether x^j is independent from the other features should not affect the PI importance score of x^j, and yet it does [82]. Since our aim is to assess the impact of x^j on the model prediction y, rather than its correlations with other features, the only question of interest is whether x^j and y are independent.

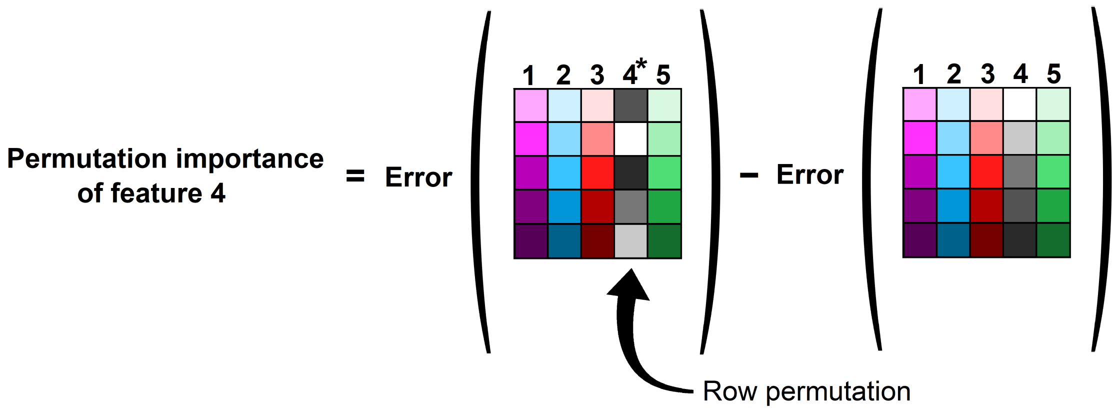


Figure 3-12: Diagram explaining how to compute the permutation importance PI $\left(x^4\right)$ of feature x^4, as per Equation 3-82, on a data whose 5 features are color-coded. We compare the error made by the classifier on a data matrix whose 4^{th} column has been randomly permuted to the error made by the classifier on an unperturbed data matrix.

3-4-2 Shapley Importance

The Shapley approach to measuring feature importance is based on the assumption that each feature is a player in a coalitional game where the model prediction is the payout [34]. We start by introducing Shapley values as defined in cooperative game theory and then present Shapley importance (SI). SI is a global post-hoc model-agnostic interpretability method that ranks features by estimating their respective contributions to a black-box model's prediction. SI is a novel method that was developed as part of this thesis. Note the difference between our SI and the Shapley additive explanations (SHAP) approach to providing local explanations for black-box machine learning models. SHAP was developed by Lundberg and Lee, in 2017 based on the work of Strumbelj and Kononenko [84].

3-4-2-1 Shapley Values from Cooperative Game Theory

According to cooperative game theory, a coalitional game is a tuple $\langle F, v\rangle$, where $F = \{1, 2, ..., n\}$ is a finite set of n players, and function $v : 2^n \rightarrow \mathbb{R}$ meets $v(\emptyset) = 0$ [85]. Subsets of players $S \subseteq F$ form coalitions and the set of all players F is called the grand coalition. Function v describes the payoff, or worth, of each coalition. The objective of cooperative game theory is to calculate the contribution of each player to game $\langle F, v\rangle$ by fairly splitting the grand coalition's worth among the $|F| = n$ players [86]. The payout assigned to the j^{th} feature, for $j = 1, 2...n$, is written ϕ_j and is termed the Shapley value of x^j. The method for calculating the contribution of each player was developed by Shapley in 1953 [86]. The Shapley value is the unique solution of game $\langle F, v\rangle$ that fairly distributes the payout among players and consequently verifies all four axioms. The four following axioms formally define the notion of fairness in game theory [85]:

- Axiom 1: Efficiency property
  For any game $\langle F, v\rangle$ it holds that $\sum_{j\in F} \phi_j(v) = v(F)$.
- Axiom 2: Symmetry property
  For any game $\langle F, v\rangle$ with two players i and j, if $v(S \in \{i\}) = v(S \in \{j\})$ holds for every subset of players S where $S \in F$ and $i, j \notin S$, it holds that $\phi_i(v) = \phi_j(v)$.
- Axiom 3: Dummy property
  For any game $\langle F, v\rangle$ such that $v(S\cup\{j\}) = v(S)$ for every $S \subseteq N$, it holds that $\phi_j(v) = 0$.
- Axiom 4: Additivity property
  For any two games $\langle F, v\rangle$ and $\langle F, w\rangle$, it holds that $\phi_j(v + w) = \phi_j(v) + \phi_j(w)$ where, for all coalitions S, $(v + w)(S) = v(S) + w(S)$.

The importance of player j to a coalition S, with $j \notin S$, is written $\Delta_j(S)$ [86]. In Equation 3-85, $v(S)$ is the payoff when all players upto j are known, excluding player j, and $v(S \cup \{j\})$ is the payoff when all players upto j are known, including j. So, $\Delta_j(S)$ represents the change in payoff resulting from adding player j to coalition S.

$$\Delta_j(S) = v(S \cup \{j\}) - v(S) \tag{3-85}$$

The Shapley value of player j, joining coalition S, is defined by the payoff ϕ_j in Equation 3-86 where Π is the set of permutations over F and $S_j(\pi)$ is the set of players appearing before the j^{th} player in permutation π [86]. Hence, **the Shapley value of player j is its average contribution to the payoff across all possible subsets of players** [86].

$$\phi_j = \frac{1}{n!} \sum_{\pi \in \Pi} \Delta_j(S_j(\pi)) \tag{3-86}$$

An equivalent expression of the Shapley value is given by Equation 3-87 [85]. $|F| = n$ is the total number of players in the grand coalition and S is a coalition that does not include player j and whose cardinality is $|S|$. Therefore $|F| - |S| - 1$ is the number of players left to be added after player j. Hence $|F|!$, $|S|!$ and $(|F| - |S| - 1)!$ are the number of possible permutations of players belonging to these respective sets.

$$\phi_j = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \underbrace{(v(S \cup \{j\}) - v(S))}_{\Delta_j(S)} \tag{3-87}$$

The Shapley approach to model interpretability regards the features as players that form coalitions S (i.e. ordered subsets) to achieve the classifier's prediction $f(x_i)$ given a specific observation $x_i = X_{(i,:)}$. The idea is to explain the behavior of a black-box supervised learning model near x_i by assigning a Shapley value ϕ_j to each feature $x_j = X_{(:,j)}$, for $j \in \{1, 2...n\}$: ϕ_j is the contribution of x^j to the difference between the classifier's actual prediction $f(x_i)$ and its mean prediction [34].

Equation 3-88 defines the classifier's prediction conditional to only a subset S of features being known [87]. The payoff $v(S)$ is the contribution of the subset of feature values S for a particular instance x_i, such that $i \in \{1, 2...m\}$. This contribution is defined in Equation 3-89 as being equal to the change in the classifier's output expectation caused by observing the feature values belonging to set S [87]: $f_S(x_S)$ is the expected prediction when we know only those values of features in S, and $f_\emptyset(x_S)$ is the expected prediction when no feature values are known. In other words, the worth of each coalition is the change in the model's prediction $v(S)$ and our aim is to fairly distribute $v(S)$ among its features [85].

$$f_S(x_S) = \mathbb{E}[f | x^j = x_i^j, \forall j \in S] \tag{3-88}$$

$$v(S) = f_S(x_S) - f_\emptyset(x_S) = \mathbb{E}[f | x^j = x_i^j, \forall j \in S] - \mathbb{E}[f] \tag{3-89}$$

Computing Shapley values requires testing the classifier f on all feature subsets $S \subseteq F$. The Shapley value ϕ_j represents the effect of including feature x^j in the model prediction. It is the difference between the model's prediction and the expected prediction if the value of x^j is unknown [87]. In order to compute ϕ_j, a model $f_{S \cup \{j\}}$ is trained with x^j, and another model $f_{S \setminus j}$ is trained without x^j [84]. The difference between the predictions of $f_{S \cup \{j\}}$ and $f_{S \setminus j}$ represents the contribution $\Delta_j(S)$ of feature x^j. The effect of withholding a feature depends on its interactions with other features in the model. Reviewing all possible feature permutations $S \subseteq F \backslash \{j\}$ is key to taking all potential dependencies and interactions into account when measuring feature importance [84]. Finally, the Shapley value ϕ_j is the weighted average of the j^{th} feature's contributions over all possible feature permutations. Equation 3-90 is therefore equivalent to Equation 3-87 [84].

$$\phi_j = \sum_{S \subseteq F \backslash \{j\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \underbrace{\Big(f_{S \cup \{j\}}\left(x_{S \cup \{j\}}\right) - f_S\left(x_S\right)\Big)}_{\Delta_j(S)} \tag{3-90}$$

The four Shapley axioms have important implications for black-box classifier explainability [87]. The efficiency property guarantees that the payout is fully divided among the features and that the contributions of each feature are implicitly normalized, which makes them easier to interpret and compare [87]. The symmetry property ensures that features that have a symmetrical impact across all subsets, that is equal contribution to all possible coalitions, will be assigned equal contributions [87]. It also implies that feature scores are not altered by arbitrarily reordering the features [86]. As for the dummy property, it states that a feature that does not influence the classifier's performance is assigned zero contribution [87]. The additivity property states that the Shapley value of a combination of different payoffs (i.e. different predictions) based on the same set of features is the sum of the corresponding Shapley values [86].

In practice, the main challenge posed by Shapley values is computational cost [34]. Computing the Shapley value of one feature x^j requires evaluating the performance of classifier f for all possible coalitions of features with and without x^j. The number of possible coalitions, and hence the computational time complexity increases exponentially with the number of features. As a result, SI is computationally unfeasible for some high-dimensional classification problems. Strumbelj and Kononenko therefore proposed an approximation with Monte-Carlo sampling that efficiently reduces the computational time complexity [87]. An approximation $\hat{\phi}_j$ of the Shapley value ϕ_j of feature x^j is presented in Equation 3-91 [34].

$$\hat{\phi}_j = \frac{1}{K} \sum_{k=1}^{K} \underbrace{f(x_k^{+j}) - f(x_k^{-j})}_{\phi_k^j} \tag{3-91}$$

$$\begin{aligned} x_k^{+j} &= \left(x_k^1, x_k^2, ..., x_k^{j-1}, x_k^j, x_i^{j+1}, x_i^{j+2}, ..., x_i^n\right) \\ x_k^{-j} &= \left(x_k^1, x_k^2, ..., x_k^{j-1}, x_i^j, x_i^{j+1}, x_i^{j+2}, ..., x_i^n\right) \end{aligned} \tag{3-92}$$

For each iteration, a random instance x_k, where $k \neq i$, is selected from the data and a random feature ordering is generated [34]. Two new instances, x_k^{+j} and x_k^{-j}, are created by combining values from the instance of interest x_i and the sample x_k. As shown in Equation 3-92, both randomized instances x_k^{+j} and x_k^{-j} have a number of feature values x_i^j replaced by feature values x_k^j taken from sample x_k [34]. The difference between x_k^{+j} and x_k^{-j} is the following: x_k^{+j} has all feature values before and including j replaced by the respective feature values of x_k, whereas x_k^{-j} has all feature values before but excluding j replaced by the respective feature values of x_k [34]. The difference in prediction from the black-box classifier ϕ_k^j is averaged over all K iterations to obtain the estimated Shapley value of feature x^j.

$$\text{Var}\left(\hat{\phi}_j\right) = \frac{\sigma_j^2}{m} \tag{3-93}$$

Approximation $\hat{\phi}_j$ is an unbiased and consistent estimate of ϕ_j whose variance is defined in Equation 3-93 where σ_j^2 is the population variance [87]. Since K is the number of iterations, reducing K will reduce computational time. However, since $\text{Var}(\hat{\phi}_j)$ is inversely proportional to K, reducing K will also increase the Shapley value estimate's variance. Hyperparameter K should be chosen large enough to accurately estimate the feature contributions, but small enough to complete the computation in a reasonable time [34]. Since the population variance σ_j^2 varies from one feature to the next, it has been suggested to adapt the number K of samples drawn to the feature x^j whose Shapley value needs to be estimated [87].

3-4-2-2 A Novel Global Measure of Feature Importance

Shapley importance (SI) is a global approach to machine learning model interpretability based on the computation of Shapley values. Rather than focus on one specific observation x_i, for $i \in \{1, 2...m\}$, we consider all m observations of data matrix X. In order to limit computational cost, we use the approximation proposed by Strumbelj and Kononenko in Equation 3-91. Our approach to computing a global estimate $\text{SI}(x^j)$ of the importance of a feature x^j consists in averaging the approximate Shapley values $\hat{\phi}_j$ across all observations x_i, as in Equation 3-94. Unlike PI, the importance score $\text{SI}(x^j)$ of feature x^j does not quantify the difference in prediction due to removing feature x^j, but rather the contribution of x^j to the classifier's prediction, averaged over all possible feature permutations and all data instances.

$$\text{SI}(x^j) = \hat{\Phi}_j = \frac{\sum_{i=1}^{m} \hat{\phi}_j}{m} \tag{3-94}$$

Figure 3-13 illustrates our global implementation of Shapley feature importance, considering a simple 5-by-5 data matrix and $K = 3$ iterations. Assuming our aim is to compute $\text{SI}(x^j)$ where $x^j = X(:, j)$ and $j \in \{1 : n\}$, one iteration produces Δ_k^j: Δ_k^j measures the global contribution of feature x^j to the classifier's prediction $\hat{y} = f(X)$ for one random permutation of the features (i.e. columns) and one random permutation of the observations (i.e. rows). In Equation 3-95, K is the number of iterations, $X^*_{\cup j}$ is obtained by randomly permuting the rows of each column after x^j, including the column of x^j, and $X^*_{\setminus j}$ is obtained by randomly permuting the rows of each column after x^j, excluding the column of x^j. Assuming knowledge of all features before x^j (i.e. the coalition), Δ_k^j measures by how much the classifier's error increases due to canceling the predictive power of x^j and effectively removing it from the coalition.

$$\text{SI}(x^j) = \hat{\phi}_j = \frac{\sum_{k=1}^{K} \Delta_k^j}{K} = \frac{\sum_{k=1}^{K} \text{E}\left(X^*_{\cup j}\right) - \text{E}\left(X^*_{\setminus j}\right)}{K} \tag{3-95}$$

To conclude, Shapley values were originally developed to assign payouts to players of a co-operative game depending on their contribution towards the total payout. SI is a post-hoc model-agnostic interpretability method that considers the features as players and the model prediction as its total payout. Note that our implementation of SI as a global measure of feature importance is new, and differs from the popular local explanation scheme, called SHAP, that was proposed by Lundberg and Lee [84]. We argue that SI is the only model-agnostic explanation method with a strong theoretical foundation, which is based on the efficiency, symmetry, dummy, and additivity axioms of Shapley values [88]. Shapley values provide a fair and efficient way to estimate the importance of features, by quantifying their respective contributions to the classifier's prediction [30]. By accounting for the interrelations among features, SI verifies our definition of feature importance, as per subsection 1-2-2. According to Molnar [34], in situations where machine learning explainability is legally required - like the European Union's General Data Protection Regulation (GDPR) - Shapley values might be the only legally compliant method. The reason why Shapley importance is more likely than permutation importance to fulfill legal requirements is that the efficiency, symmetry, dummy, and additivity axioms of Shapley values guarantee that the model's prediction is fairly distributed among the features [34]. Note however that, similarly to the interpretability methods studied previously, SI may provide misleading importance estimates if there is a high degree of dependence and correlation among some or all the features [88].

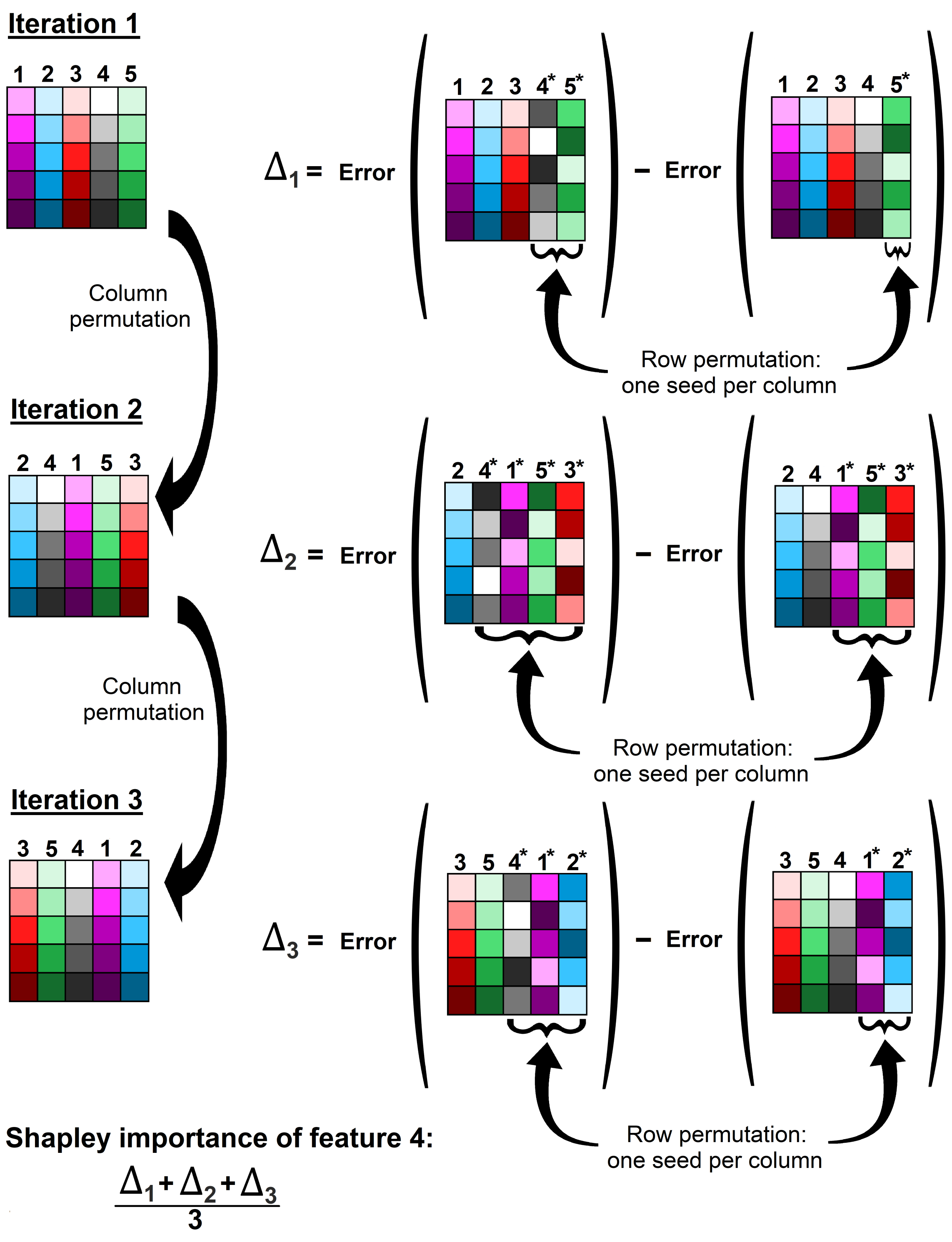


Figure 3-13: Diagram explaining how to compute the (global) Shapley importance SI $\left(x^4\right)$ of feature x^4, as per Equation 3-94, on data whose 5 features are color-coded. The feature of interest is black-and-white. We arbitrarily choose $K = 3$ iterations. At each iteration $k = 1, 2, 3$, the columns of the data matrix are randomly reordered and we compute the difference Δ_k between the error made by the classifier with and without knowledge of x^4, assuming knowledge of the features to its left. Δ_k effectively measures the contribution of feature x^4 to the predictive power of the coalition of features to its left. SI $\left(x^4\right)$ is the average of Δ_1, Δ_2 and Δ_3.

Chapter 4

Experiments

4-1 Two Imaging Mass Spectrometry Datasets

	Sample	IMS method	Classification task
Dataset n^{o}1	Coronal section of a rat brain	MALDI-FTICR	Binary classification of the diseased left brain hemisphere versus the healthy right brain hemisphere
Dataset n^{o}2	Sagittal whole-body section of a mouse	MALDI-TOF	Classification of different organs: brain, heart, lungs, liver, adipose tissue

Table 4-1: Differences between our two imaging mass spectrometry datasets.

Table 4-1 presents the two imaging mass spectrometry (IMS) datasets that we worked on in the course of this thesis. Please refer to subsection 1-1-1 for an explanation of matrix assisted laser desorption ionization (MALDI), and for an explanation of the mode of operation of the Fourier transform ion cyclotron resonance (FTICR) and time-of-flight (TOF) mass analyzers. What follows is an overview of the main differences between these two datasets.

Dataset n^{o}1, obtained by MALDI-FTICR IMS from the coronal section of a rat brain, was acquired for the purpose of a study on Parkinson's disease jointly conducted by the Delft University of Technology and Vanderbilt University [89]. Parkinson's disease is a slowly progressing neurodegenerative disorder whose symptoms include tremors, muscle stiffness, slowness of movement and imbalance [90]. It is characterized by a degeneration of dopaminergic neurons (i.e. neurons that synthesize dopamine) and the formation of Lewy bodies (i.e. protein aggregates) in the substantia nigra [90]. The substantia nigra is a region of the basal ganglia. The basal ganglia is a group of nuclei (i.e. clusters of neurons) located beneath the cerebral cortex that is in charge of regulating movement [91]. Dopamine is the primary neurotransmitter (i.e. chemical messenger) of the basal ganglia [91]. The loss of dopaminergic neurons severely impairs the brain's motor function, leading to movement disorders [91]. The dopaminergic neurons of a rat's left brain hemisphere were destroyed to make it resemble a diseased brain [89]. The right hemisphere was used as a control [89]. We therefore have two classes: the dopamine depleted left brain hemisphere is labeled as positive, and the healthy right brain hemisphere is labeled as negative. Our aim is to study the biomolecular differences between the two hemispheres. Hence a binary classification problem with balanced classes. Dataset n^{o}1 is annotated as illustrated in Figure 4-1. Since card($\mathcal{C}_+$) $\approx$ card($\mathcal{C}_-$), we use accuracy to measure classification performance. Dataset n^{o}1 is divided into a training set (70%) on which we train the classifier, a validation set (10%) on which we tune the classifier's hyper-parameters, and a testing set (20%) on which we evaluate the classifier's generalization performance. Dataset n^{o}1 consists of 17964 pixels (i.e. observations) and 2003 m/z bins (i.e. features).

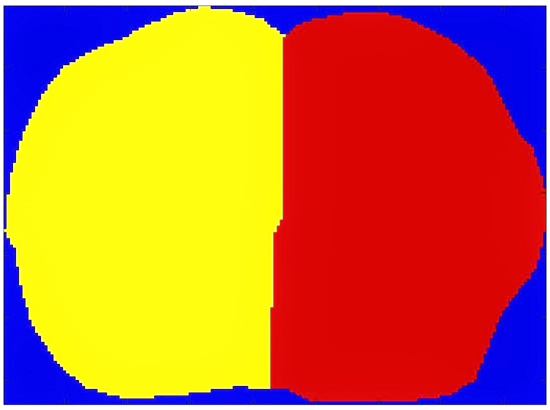


Figure 4-1: Diagram of the positive class in yellow (i.e. left diseased brain hemisphere) and negative class in red (i.e. right healthy brain hemisphere) in dataset n^{o}1.

Dataset n^{o}2, obtained by MALDI-TOF IMS from the sagittal whole-body section of a mouse-pup, was acquired for the purpose of a study on ion mobility IMS jointly conducted by the Delft University of Technology and Vanderbilt University [19]. Our aim is to classify five organs, namely the brain, heart, liver, lungs, and adipose tissue. In section 4-2, we explain how to manually define the organ masks using the parts-based representation provided by non-negative matrix factorization (NNMF). Unlike for dataset n^{o}1, which is a binary classification problem, analyzing dataset n^{o}2 is a matter of solving a classification problem with multiple classes. We use the one-versus-all (OVA) approach to decompose the problem into five binary classification problems, each of which consists in recognizing one of the five classes. The OVA approach therefore requires training five binary classifiers. Before training each classifier, dataset n^{o}2 is labeled such that the target organ is the positive class $\mathcal{C}_+$ and all other organs make up the negative class $\mathcal{C}_-$. Each one of these binary classification problems is severely imbalanced since card($\mathcal{C}_+$) $<<$ card($\mathcal{C}_-$). The brain represents approximately 8% of all acquired pixels, the liver represents 4% of all acquired pixels, the dorsal adipose tissue represents 3% of all acquired pixels, and the heart and lungs only amount to approximately 1% of all acquired pixels. In order to treat each organ equally, we choose to downsample the negative class so that the positive class, for every one of the five binary classification problems, represents 40%

of the total dataset. Downsampling also speeds up classifier training, limits overfitting, and reduces memory requirements. We choose not to further downsample the negative class to ensure that the learning algorithm considers the wide range of anatomical structures that our classifier must distinguish from the organ of interest. Further downsampling the negative class was observed to worsen classifier performance. We now have five binary classification problems where the ratio of $\mathcal{C}_+$ to $\mathcal{C}_-$ is 2 : 3. We use precision, recall, and the F-score, as well as accuracy, to measure classification performance. These measures are, as discussed in subsection 1-2-1, recommended for evaluating classifier predictive performance when dealing with imbalanced classes. Since the organ masks we define in section 4-2 do not exhaustively account for all the pixels belonging to each organ, we evaluate the generalization performance of our trained classifiers primarily by visual inspection of the anatomical distribution. Dataset n^{o}2 consists of 164808 pixels (i.e. observations) and 321 m/z bins (i.e. features).

4-2 Definition of Organ Masks for Dataset n^{o}2

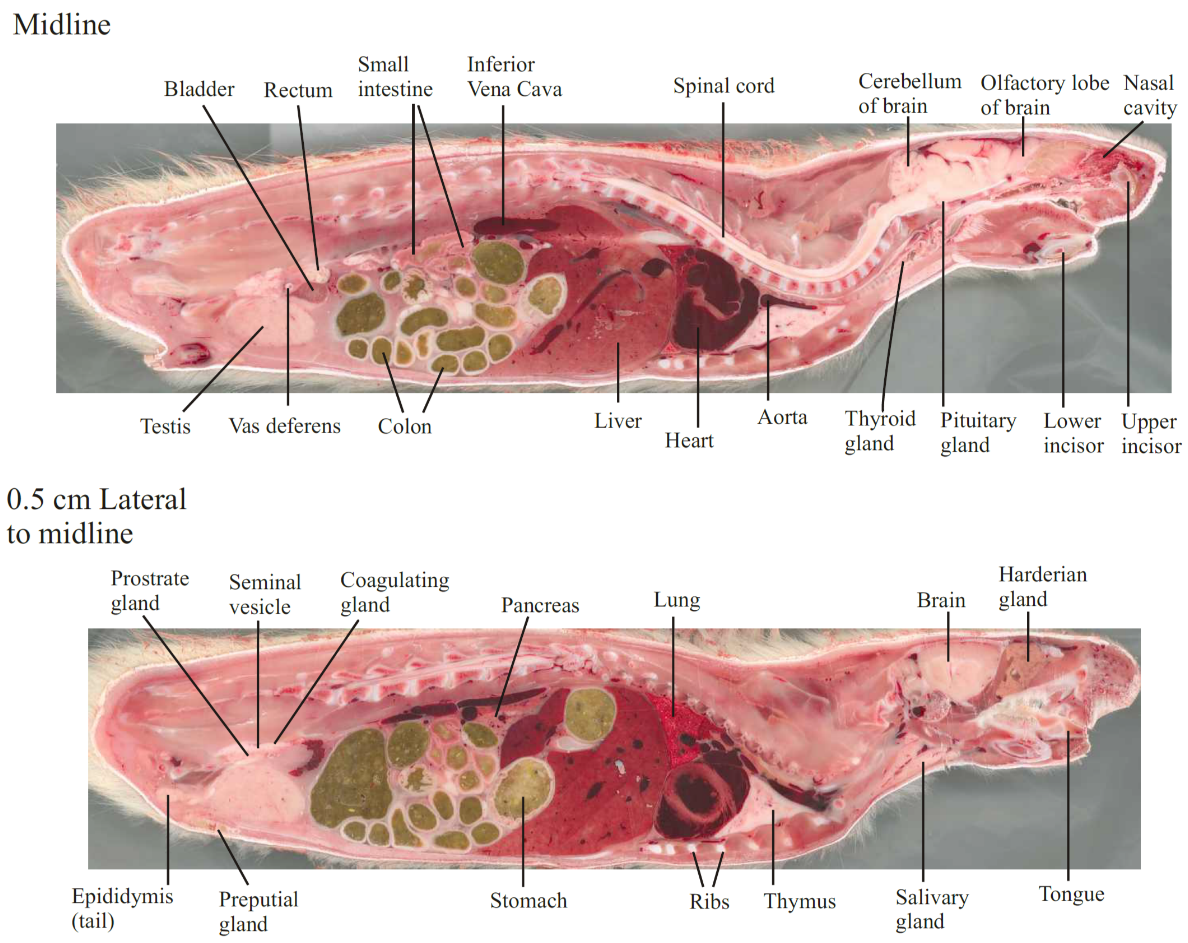
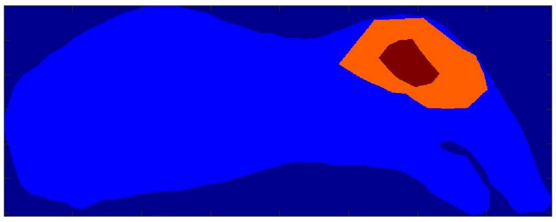


Figure 4-2: Sagittal whole-body section of a rat useful for annotating the brain, liver, heart, lungs, and dorsal adipose tissue in dataset n^{o}2. Figure copied from [10]. Copyright © 2008, 2009 InvivoPharm Inc.

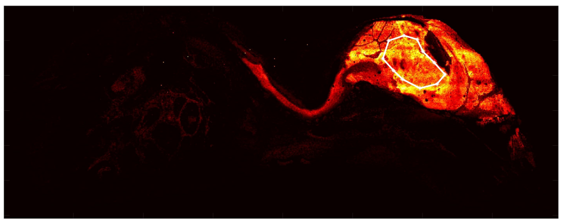
We manually annotate dataset n^{o}2 by means of the latent features provided by NNMF. As discussed in section 2-2, non-negative matrix factorization (NNMF) yields a parts-based representation of dataset n^{o}2. Annotating the five organs under study, namely the brain, heart, liver, lungs, and dorsal adipose tissue, is required for generating the anatomical class labels needed by supervised learning algorithms.

We use the anatomical atlas of a rat presented in Figure 4-2 to recognize our five target organs in the images obtained by plotting the NNMF latent features' degree of expression across the sample surface. We assume that the mouse-pup from whom dataset n^{o}2 was acquired has a similar anatomy as the rat in Figure 4-2. In Figures 4-3, 4-4, 4-5, 4-6, and 4-7, we define the masks corresponding to the brain, liver, heart, lungs, and dorsal adipose tissue respectively.

In Figures 4-3a, 4-4a, 4-5a, 4-6a, and 4-7a the red mask indicates which pixels were annotated as belonging to the brain, heart, liver, lungs, and dorsal adipose tissue respectively. The red mask corresponds to the region circled in white in the NNMF latent features of Figures 4-3b, 4-4b, 4-5b, 4-6b, and 4-7b. The orange mask indicates which pixels might belong to the organ under study but could not be annotated with absolute certainty. The blue region was annotated as mouse-pup tissue that does not belong to the organ under study. The dark blue region is the background. For each binary classification problem supposed to recognize one organ, the training dataset is made up of the red region (i.e. positive class) and the blue region (i.e. negative class). The orange region is excluded from the training set in order to avoid providing wrongly labeled pixels to the supervised learning algorithm. The testing set is made up of the red, orange, and blue regions, effectively evaluating the classifier's generalization performance across the entire tissue section. The dark blue background is excluded from both the training and testing sets since it does not correspond to mouse-pup tissue. Reliably and exhaustively annotating pixels is difficult to do manually, so rather than quantify the trained classifier's performance on the testing set, its performance is qualitatively evaluated by visual inspection.

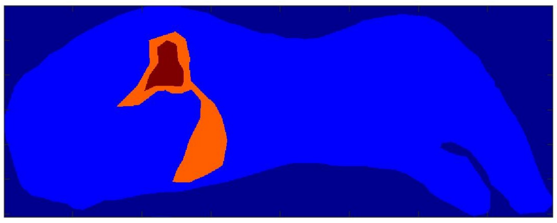


(a) Diagram of the mask used for training the binary classifier used to recognize the brain: the red pixels belong to the brain and are therefore labeled positive, the blue pixels do not belong to the brain and are therefore labeled negative, the orange pixels cannot be reliably annotated so they are excluded from the training set. The dark blue pixels are non-tissue background.

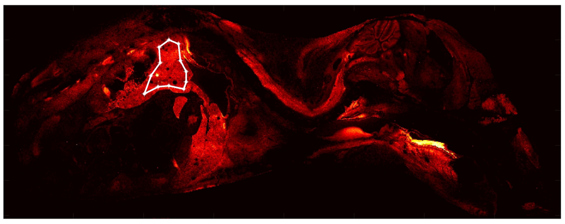


(b) Spatial distribution and relative degree of expression of the latent feature obtained by non-negative matrix factorization that highlights the brain. The region circled in white is the region manually annotated as belonging to the brain.

Figure 4-3: Manual annotation of the brain in dataset n^{o}2

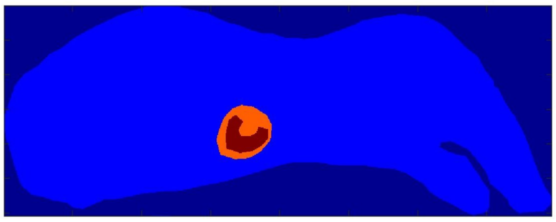


(a) Diagram of the mask used for training the binary classifier used to recognize the liver: the red pixels belong to the liver and are therefore labeled positive, the blue pixels do not belong to the liver and are therefore labeled negative, the orange pixels cannot be reliably annotated so they are excluded from the training set. The dark blue pixels are non-tissue background.

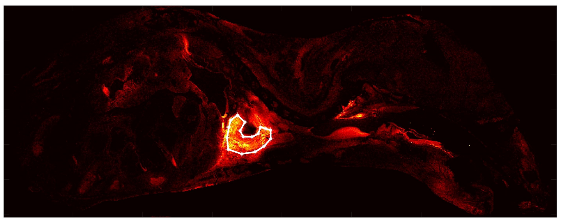


(b) Spatial distribution and relative degree of expression of the latent feature obtained by non-negative matrix factorization that highlights the liver. The region circled in white is the region manually annotated as belonging to the liver.

Figure 4-4: Manual annotation of the liver in dataset n^{o}2

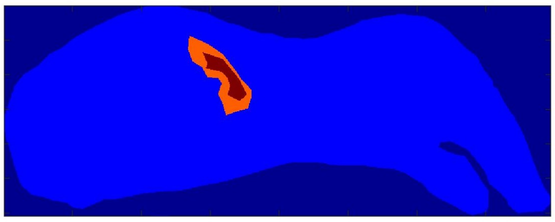


(a) Diagram of the mask used for training the binary classifier used to recognize the heart: the red pixels belong to the heart and are therefore labeled positive, the blue pixels do not belong to the heart and are therefore labeled negative, the orange pixels cannot be reliably annotated so they are excluded from the training set. The dark blue pixels are non-tissue background.

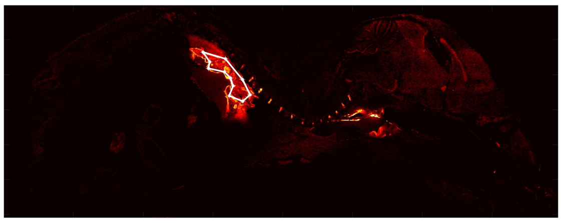


(b) Spatial distribution and relative degree of expression of the latent feature obtained by non-negative matrix factorization that highlights the heart. The region circled in white is the region manually annotated as belonging to the heart.

Figure 4-5: Manual annotation of the heart in dataset n^{o}2

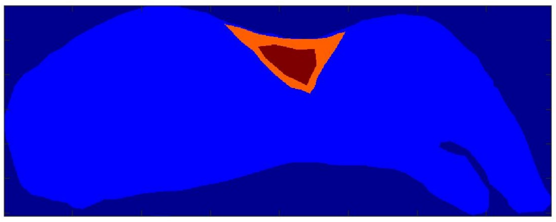


(a) Diagram of the mask used for training the binary classifier used to recognize the lungs: the red pixels belong to the lungs and are therefore labeled positive, the blue pixels do not belong to the lungs and are therefore labeled negative, the orange pixels cannot be reliably annotated so they are excluded from the training set. The dark blue pixels are non-tissue background.

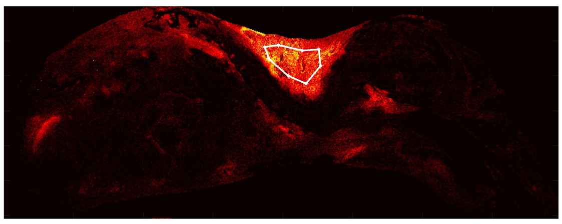


(b) Spatial distribution and relative degree of expression of the latent feature obtained by non-negative matrix factorization that highlights the lungs. The region circled in white is the region manually annotated as belonging to the lungs.

Figure 4-6: Manual annotation of the lungs in dataset n^{o}2



(a) Diagram of the mask used for training the binary classifier used to recognize dorsal adipose tissue: the red pixels correspond to dorsal adipose tissue and are therefore labeled positive, the blue pixels do not correspond to dorsal adipose tissue and are therefore labeled negative, the orange pixels cannot be reliably annotated so they are excluded from the training set. The dark blue pixels are non-tissue background.



(b) Spatial distribution and relative degree of expression of the latent feature obtained by non-negative matrix factorization that highlights the dorsal adipose tissue. The region circled in white is the region manually annotated as belonging to the dorsal adipose tissue.

Figure 4-7: Manual annotation of the dorsal adipose tissue in dataset n^{o}2

4-3 Workflow for Biomarker Discovery

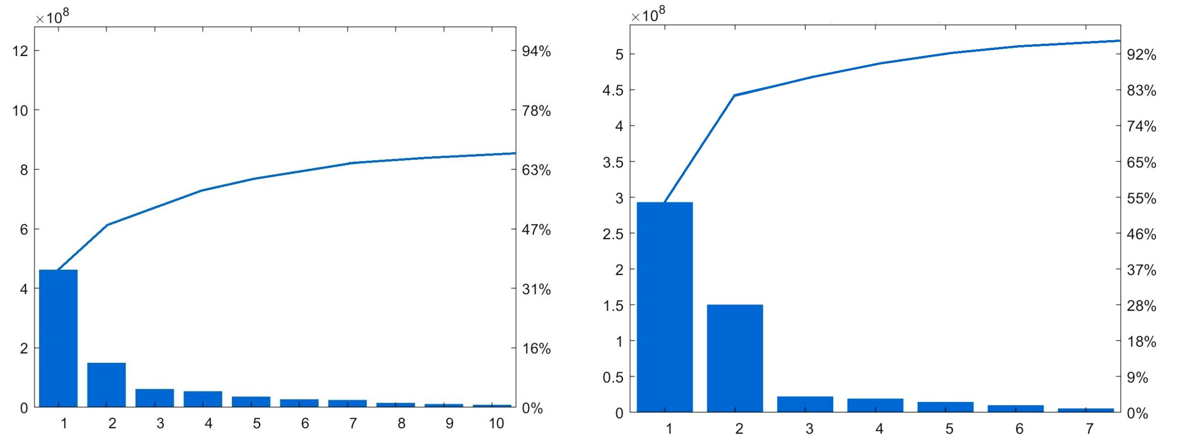
As discussed in subsection 1-2-1 of Chapter 1 and in Chapter 3, our aim is biomarker discovery in high-dimensional, large-scale imaging mass spectrometry (IMS) data. A biomarker is an objectively measurable molecular indicator of a specific biological state. We use interpretable supervised machine learning algorithms to identify the predictor variables, or features, that are most important for classifying data instances, or observations. When dealing with IMS data, each feature is the m/z bin corresponding to an ionized molecular species and each observation is the mass spectrum of one of the pixels making up the sample's surface. Each dataset is represented by an m-by-n matrix X whose m rows are observations and whose n columns are features. We use the superscript j, ranging from 1 to n, to denote a particular feature $x^j = X_{(:,j)}$ and we use subscript i, ranging from 1 to m, to denote a particular observation $x_i = X_{(i,:)}$. In our work, biomarker discovery is essentially a matter of identifying a list of highly discriminative features, and thus molecular species, that may be used to differentiate between two or more classes of data instances.

In the case of dataset n^{o}1, we have a balanced binary classification problem whose positive class is diseased rat brain tissue and whose negative class is healthy rat brain tissue. The masks used to label the positive and negative classes of dataset n^{o}1 are given by Figure 4-1. In the case of dataset n^{o}2, we have an imbalanced classification problem with five classes corresponding to five organs of a mouse-pup: the brain, the liver, the heart, the lungs, and the dorsal adipose tissue. According to the OVA method, we decompose the task of analyzing dataset n^{o}2 into five imbalanced binary classification problems, each of which involves recognizing one of the five organs. The masks used to label each organ were manually defined in Figures 4-3a, 4-4a, 4-5a, 4-6a, and 4-7a. The main challenge posed by dataset n^{o}1 was the large number of features, whereas the main challenge posed by dataset n^{o}2 was the large number of observations.

Our preprocessing pipeline involves noise removal by principal component analysis (PCA) and feature centering. Referring back to Chapter 2, we use Equation 2-26 to remove low-variance instrumental and chemical noise from our two IMS datasets. Note that PCA cannot remove the high-variance instrumental noise that affects dataset n^{o}1. Figures 4-8a and 4-8b illustrate which proportion of the original variance of dataset n^{o}1 and n^{o}2 is explained by each principal component (PC). According to Jolliffe, the required number of PCs should be chosen according to the cumulative percentage of total variance we wish the selected PCs to retain [42]. Since dataset n^{o}1 is more noisy than dataset n^{o}2, we choose to retain 90% of its original variance: we therefore select the first 361 PCs, out of a total of 2003 PCs and omit the remaining low-variance, noisy PCs. As for dataset n^{o}2, we choose to retain 99% of the original variance and therefore keep 21 out of 321 PCs. The score images and loading pseudo-scores of the two first PCs of datasets n^{o}1 and n^{o}2 are given by Figures 2-3 and 2-4 respectively. Noise removal is followed by feature zero-centering, which consists in subtracting the column-wise mean from the data, as per Equation 2-1, to set the mean of each feature to zero. Centering is recommended for speeding up the training of our supervised learning algorithms without altering the relationship among features and observations [21]. Zero-centering is followed by unit-variance scaling, which refers to the division of each column by its standard deviation so that each variable has unit variance. Zero-centering followed by unit-variance scaling is termed standardization because standardized features have a mean of zero and a standard deviation of one. Since IMS features are all ionized molecules measured by their respective mass-to-charge ratios and intensities (i.e. same unit), scaling should not be necessary. It may even have undesirable consequences, such as artificially boosting the predictive power of low-variance extraneous noise. Yet, in practice, we observed that logistic regression (LR) and especially support vector machine (SVM) are sensitive to feature scaling and that their performance benefits considerably from unit-variance scaling the features prior to classifier training. As discussed in 3-1-1, we use gradient descent to estimate the feature weights of LR models: without standardization, the weights of features with a greater numeric range will update faster than those of features with a smaller numeric range, resulting in the weights of features with a smaller numeric range not properly converging. Standardization is recommended to avoid penalizing features with a smaller numeric range when performing logistic regression with regularization [92]. As discussed in subsection 3-2-1, training and testing linear SVM models requires the computation of a kernel, which relies on taking the inner-product of observations in feature-space. Standardization is recommended to avoid numerical difficulties during the calculation of SVM kernels [93]. Since the training of random forests does not rely on measuring distances between data instances in feature space, random forest (RF) is insensitive to feature scaling. Nevertheless, in order to guarantee a fair comparison of the different classifiers, we standardize the IMS features before training LR, SVM, and RF models. To conclude on the question of pre-processing IMS data, we perform noise removal by PCA followed by standardization.

As discussed in Chapter 3, we use three types of supervised learning models for the binary classification of IMS data: logistic regression (LR), support vector machines (SVM) and random forests (RF). After having tuned each model's hyper-parameters, we compare their performance on a total of six binary classification problems: differentiating the left brain hemisphere from the right brain hemisphere of dataset n^{o}1, and recognizing five organs (one by one), namely the brain, liver, heart, lungs, dorsal adipose tissue, in dataset n^{o}2. Classification performance is measured in terms of accuracy, precision and recall, as per subsection 1-2-1. Since dataset n^{o}2 is imbalanced, we also use the F-score. As discussed in subsection 1-2-2, identifying the most important features is key to machine learning model interpretability and biomarker discovery. Feature ranking refers to the task of ordering the features by descending order of discriminative importance. In order to guarantee that our classifiers rely primarily on biologically-relevant features, rather than noise, we verify that the ion images of top-ranking features represent histological patterns. We rank molecular features using the five following interpretability methods, which are discussed in Chapter 3. Note that the model-specific interpretability methods of LR and linear SVM models do not account for feature inter-dependencies, whereas the model-specific interpretability method of RF does. We argue that the model-specific interpretability method of RF is therefore more reliable.

- A model-specific method intrinsic to LR defined by Equation 3-16 of subsection 3-1-2. Relative weights analysis is an interesting alternative to the method of subsection 3-1-2. Yet, since relative weights analysis involves decorrelating the data before training the classifier, it cannot easily be compared to the other interpretability methods.
- A model-specific method intrinsic to linear SVM defined by Equation 3-50 of subsubsection 3-2-1-3.
- A model-specific method intrinsic to RF, called mean decrease impurity (MDI), defined by Equation 3-81 of subsection 3-3-3.
- A model-agnostic post-hoc method, called permutation importance (PI), defined by Equation 3-82 of subsection 3-4-1.
- A model-agnostic post-hoc method, called Shapley importance (SI), defined by Equation 3-95 of subsection 3-4-2.



(a) Dataset n^{o}1: Pareto chart of the variance explained for by the first 10 principal components

(b) Dataset n^{o}2: Pareto chart of the variance explained for by the first 7 principal components

Figure 4-8: Pareto charts indicating the percentage of variance explained by the first principal components (PCs) in the case of dataset n^{o}1 and dataset n^{o}2.

The measure we propose to compare the feature rankings obtained by these four interpretability methods on IMS data is termed the percentage of overlapping molecules (POM). The POM is adapted from the percentage of overlapping genes (POG) frequently used in genomics to evaluate the consistency between two lists of differentially expressed genes [94]. We define the POM as the rate of agreement between the top 10% of two feature rankings: the POM is obtained by counting how many features both rankings have in common. It essentially defines the consistency between two lists of highly discriminative features as their intersection. In Equation 4-1, we consider one of two IMS datasets under study and compute the POM between feature rankings R_1 and R_2 obtained by two of the five interpretability methods listed above. Depending on its predictive importance score, feature $x^j = X_{(:,j)}$ with $j \in \{1, 2...n\}$, is assigned a rank $r_1^j = R_1(x^j)$ in list R_1 and rank $r_2^j = R_2(x^j)$ in list R_2. I denotes the indicator function such that $I(r_1^j \leq n_{\text{top}}) = 1$ if and only if $r_1^j \leq n_{\text{top}}$, otherwise I returns 0. The number of important features n_{top} is set to 200 for dataset n^{o}1 (i.e. 10% of n=2003) and 32 for dataset n^{o}2 (i.e. 10% of n=321).

$$\text{POM}\left(R_1, R_2, n_{\text{top}}\right) = \sum_{j=1}^{n} I\left(r_1^j \leq n_{\text{top}} \land r_2^j \leq n_{\text{top}}\right) \tag{4-1}$$

Biomarker discovery is done for all six following classification problems. We provide a list of highly discriminative features for each task, as agreed upon by our four interpretability methods.

1. Recognizing the left brain hemisphere from the right brain hemisphere in dataset n^{o}1.
2. Recognizing the brain, liver, heart, lungs and adipose tissue from the other mouse organs in dataset n^{o}2.

Note that dataset n^{o}2 has more than one hundred thousand pixels, which makes the training of support vector machines (SVMs) computationally very costly. Storing the kernel of IMS dataset n^{o}2, which is a 164808-by-164808 matrix of data instances, in double precision format requires 202 gigabytes of memory. Although sequential minimal optimization (SMO) reduces memory requirements to $O(m)$, its required computational time scales along $O\left(m^{2.3}\right)$ [6]. This poses a problem for training SVM classifiers on very large datasets like IMS dataset n^{o}2. It has been pointed out that the use of a single threshold to verify optimality is an important source of inefficiency in Platt's SMO algorithm [95]. Unfortunately, our approach to training SVM classifiers for IMS data uses Platt's SMO algorithm, as explained in subsection 3-2-3. Furthermore, we observe that the time complexity of our implementation of SMO is very sensitive to the box constraint tolerance that enforces $0 < \lambda < C$, where C is the regularization parameter, and to the Lagrange multiplier convergence precision. We therefore recommend thoroughly tuning these two hyper-parameters. Our results were obtained with a box constraint tolerance of 0.01 and a Lagrange multiplier convergence precision of $\epsilon = 0.001$. Although we successfully trained a linear SVM classifier on dataset n^{o}1 in subsubsection 5-1-1-3, training SVMs on dataset n^{o}2 turned out to be too computationally expensive to be practical for all five organs. We nevertheless present the result obtained by a linear SVM classifier that is trained to recognize the mouse-pup's liver from its other organs in subsubsection 5-1-2-3.

4-4 Code contributions

Every unsupervised and supervised machine learning algorithm, and every model-specific and model-agnostic interpretability method used in the course of our work on IMS data was implemented from scratch in MATLAB. What follows is a brief description of our MATLAB functions:

- Non-negative matrix factorization (NNMF) using either the multiplicative update algorithm, the alternating least squares (ALS) algorithm, or the hierarchical alternating least squares (HALS) algorithm: `NNMF.m`
- Choice of the reduced dimension k for NNMF using either the dispersion coefficient, the average silhouette index, or the proportion of ambiguous clustering: `chooseK_NNMF.m`
- Computation of the NNMF consensus matrix required by `chooseK_NNMF.m` for estimating the dispersion coefficient and the proportion of ambiguous clustering given a target rank k: `consensus_NNMF.m`
- Computation of the silhouette score for a given target rank k of NNMF, required by `chooseK_NNMF.m` to choose k: `silhouette_NNMF.m`
- Principal component analysis (PCA) using the singular value decomposition (SVD) approach: `PrincipalComponentAnalysis.m`
- Perform decorrelation by whitening, also termed sphering, using either the zero-phase component analysis (ZCA), ZCA-cor, PCA, PCA-cor or the Cholesky decomposition methods: `Sphering.m`
- Training of the logistic regression model with a choice of regularization parameter: `LogisticRegressionTrain.m`
- Testing of the logistic regression model: `LogisticRegressionTest.m`
- Relative weights analysis of logistic regression: `ComputeRelativeWeights.m`
- Training of the support vector machine model by SMO with a choice between a linear, polynomial, sigmoid, or Gaussian radial basis function kernel: `SupportVectorMachineTrain.m`
- Computation of the linear, polynomial, sigmoid, or Gaussian radial basis function kernel matrix: `KernelSVM.m`
- Search for data instances whose Lagrange multipliers violate the Karush-Kuhn-Tucker (KKT) conditions, as required by SMO: `ExamineExample.m`
- Perform one step of SMO by solving a quadratic programming (QP) problem involving two Lagrange multipliers: `TakeStep.m`
- Testing of the support vector machine model: `SupportVectorMachineTest.m`
- Training of a decision stump by choosing which feature to split on, and which threshold to use for a given internal node: `DecisionStumpTrain.m`

- Testing of the decision stump: `DecisionStumpTest.m`
- Training of the decision tree according to the recursive binary splitting approach, with a choice of node impurity measure (either the Shannon entropy of the Gini index): `DecisionTreeTrain.m`
- Testing of the decision tree: `DecisionTreeTest.m`
- Training of the random forest model: `RandomForestTrain.m`
- Testing of the random forest model: `RandomForestTest.m`
- Computation of permutation importance for all features of a data matrix given a trained classifier, either a logistic regression, support vector machine or random forest model: `PermutationImportance.m`
- Computation of the Shapley importance of all features of a data matrix given a trained classifier, either a logistic regression, support vector machine or random forest model: `ShapleyImportance.m`

Chapter 5

Results & discussion

5-1 Classification & feature ranking

5-1-1 Dataset n^{o}1

5-1-1-1 Logistic regression

Logistic regression performs well on dataset n^{o}1, both for the purposes of classification and feature ranking. Figure 5-1 shows the results of binary classification with an logistic regression (LR) classifier whose regularization parameter is set to $\lambda = 8000$. A large regularization parameter was necessary to account for LR's tendency to overfit the training data. In Figure 5-1, the white pixels are labeled positive, whereas the black pixels are labeled negative. Note that 70% of the pixels represented in Figure 5-1 made up the LR classifier's training set, and that 30% made up its testing set. Figure 4-1 is the benchmark by which we evaluate the LR classifier's performance. The training accuracy achieved by our LR classifier is 93.24% and its testing accuracy is 92.90%. The precision and recall obtained on the testing set are 92.85% and 92.98%. We rank the features of LR according to their decreasing order of predictive importance using the interpretability method presented in subsection 3-1-2. The importance score attributed to each feature is equal to the magnitude of the feature's weight in the LR classifier's decision boundary. This feature importance measure does not account for correlated features. The ion images of the twenty top-ranking features are presented in Figures 5-2, 5-3 and 5-4. These ion images are displayed using a pseudo-color scale whose brightness is indicative of the signal intensity measured at a given pixel: signal intensity correlates with the relative molecular concentration at that location. We observe that logistic regression is sensitive to noise: the top first four features, whose ions images are displayed in Figure 5-2, are high-variance instrumental noise. In Figure 5-3, we observe that features n^{o}6, n^{o}7, n^{o}8, n^{o}16 and n^{o}17 are instrumental noise. All other features in Figures 5-3 and 5-4 encode biological information. We also observe that many of the top-ranking features correspond to ionized molecules with very similar mass-to-charge ratios (i.e. m/z bins) and very similar spatial distributions: for example, features n^{o}5 and n^{o}9, or features n^{o}12 and n^{o}15, or n^{o}11, n^{o}19 and n^{o}20. We assume these features to be isotopes. Isotopes are chemical species with the same number of protons but different numbers of neutrons.

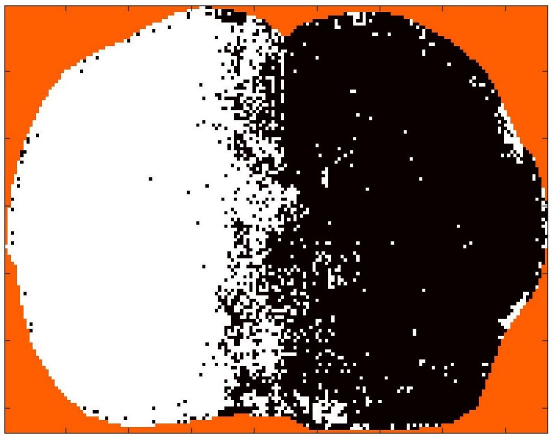
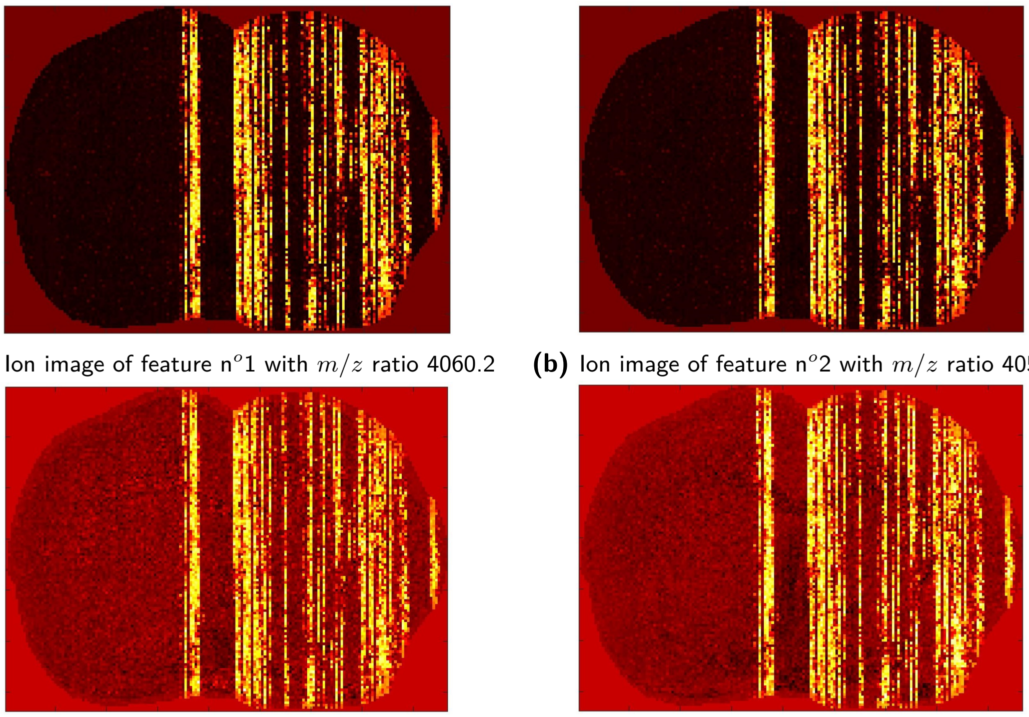


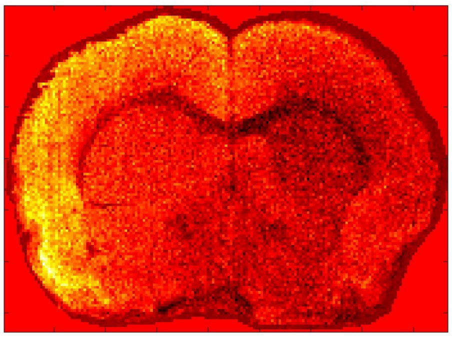
Figure 5-1: Classification of dataset n^{o}1 achieved by a logistic regression (LR) classifier whose regularization parameter is $\lambda = 8000$: the positive class (white) is the diseased left brain hemisphere, whereas the negative class (black) is the healthy right brain hemisphere. The accuracy, precision and recall measured on the test set are 92.90%, 92.85% and 92.98% respectively.

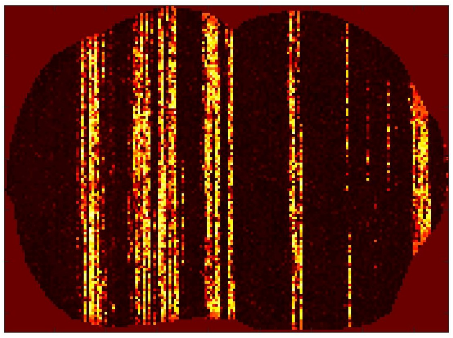


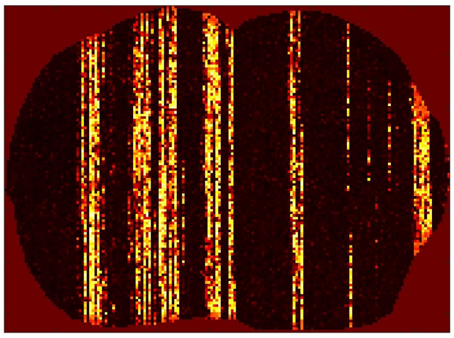
(a) Ion image of feature n^{o}1 with m/z ratio 4060.2 **(b)** Ion image of feature n^{o}2 with m/z ratio 4051.6

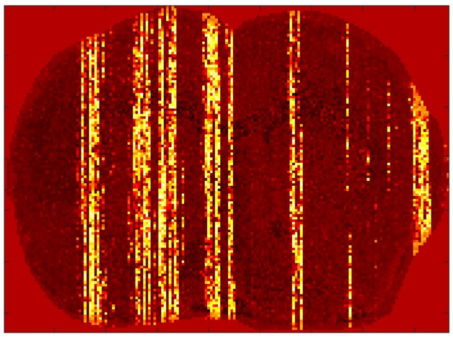
(c) Ion image of feature n^{o}3 with m/z ratio 4043.1 **(d)** Ion image of feature n^{o}4 with m/z ratio 4068.8

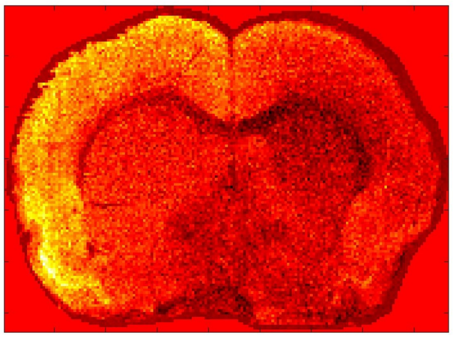
Figure 5-2: Ion images of the top-ranking features, from n^{o}1 to n^{o}4, according to the logistic regression classifier trained on imaging mass spectrometry dataset n^{o}1.

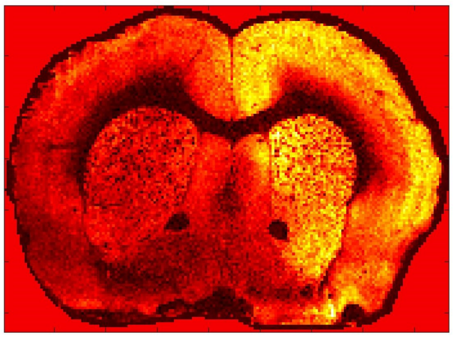

(a) Ion image of feature n^{o}5 with m/z ratio 1824.0

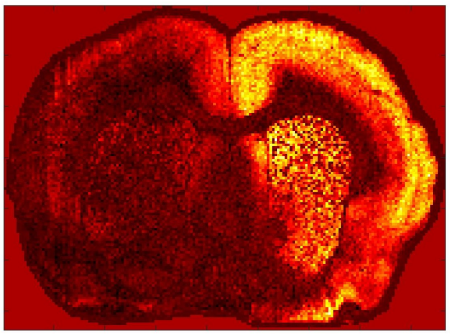

(b) Ion image of feature n^{o}6 with m/z ratio 4060.7


(c) Ion image of feature n^{o}7 with m/z ratio 4052.1


(d) Ion image of feature n^{o}8 with m/z ratio 4069.2


(e) Ion image of feature n^{o}9 with m/z ratio 1823.0


(f) Ion image of feature n^{o}10 with m/z ratio 5610.9


(g) Ion image of feature n^{o}11 with m/z ratio 1828.7

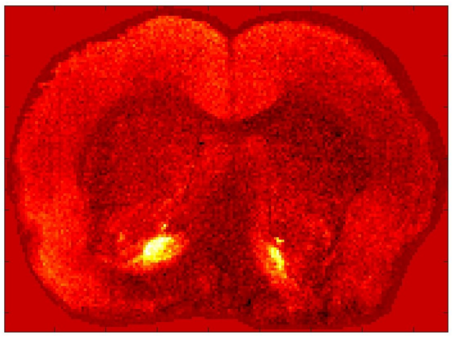
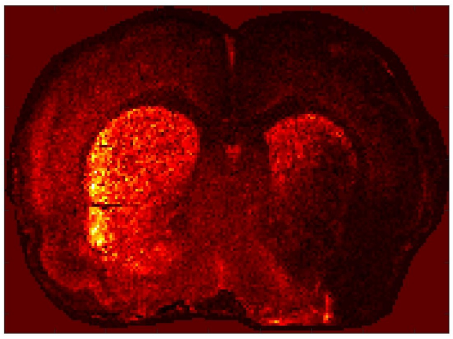
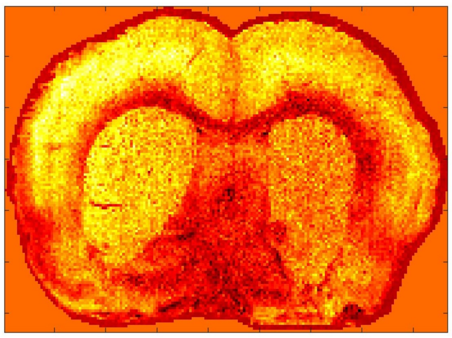
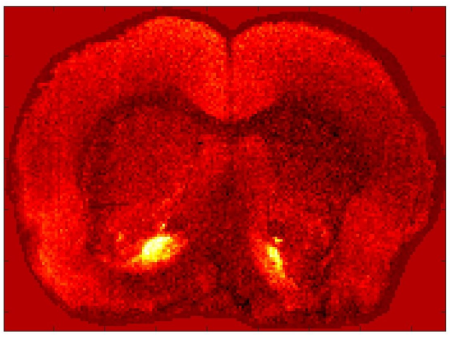
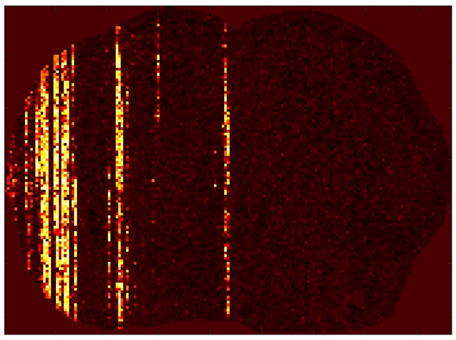

(h) Ion image of feature n^{o}12 with m/z ratio 2441.0

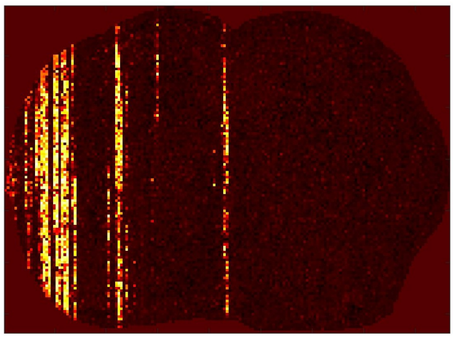
Figure 5-3: Ion images of the top-ranking features, from n^{o}5 to n^{o}12, according to the logistic regression classifier trained on imaging mass spectrometry dataset n^{o}1.

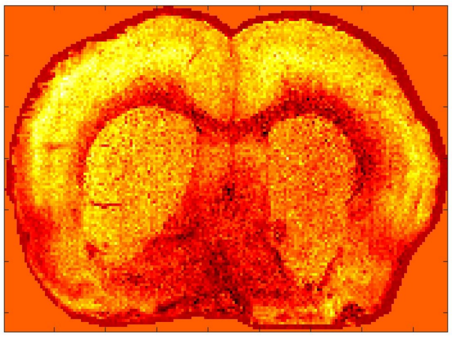

(a) Ion image of feature n^{o}13 with m/z ratio 1756.0

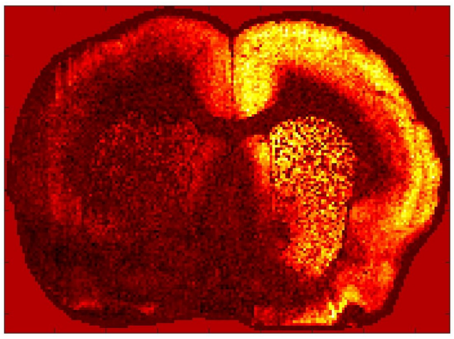

(b) Ion image of feature n^{o}14 with m/z ratio 1927.9


(c) Ion image of feature n^{o}15 with m/z ratio 2440.0


(d) Ion image of feature n^{o}16 with m/z ratio 4052.6


(e) Ion image of feature n^{o}17 with m/z ratio 4061.2


(f) Ion image of feature n^{o}18 with m/z ratio 1929.9


(g) Ion image of feature n^{o}19 with m/z ratio 1829.3

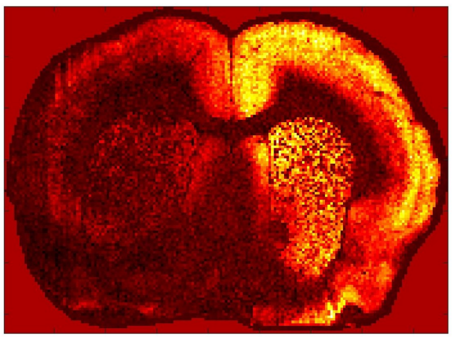

(h) Ion image of feature n^{o}20 with m/z ratio 1829.0

Figure 5-4: Ion images of the top-ranking features, from n^{o}13 to n^{o}20, according to the logistic regression classifier trained on imaging mass spectrometry dataset n^{o}1.

5-1-1-2 Random forest

Hyper-parameter tuning on dataset n^{o}1 yielded the following settings for our random forest (RF) classifier: the number of trees making up the random forest was set to 500, the maximum tree depth was set to 20, the number of features to consider as splitting candidates at each node was set to 500, the maximum terminal node cardinality was set to 10 data instances, and the node impurity measure was defined as the Gini index. The classification results are presented in Figure 5-5: the white pixels are labeled positive, whereas the black pixels are labeled negative. The partition between positive and negative classes achieved in Figure 5-5 is very similar to that of Figure 4-1. The training accuracy of the RF classifier is 99.33% and its testing accuracy is 93.57%. The recall and precision measured on the testing set are 92.45% and 94.74% respectively. The nonlinear decision boundary gives RF the ability to model complex high-dimensional biochemical processes. The classification boundary between the left and right brain hemispheres that is displayed in Figure 5-5 is visibly less noisy than that of Figure 5-1, which was obtained using LR. The features are ranked in decreasing order of mean decrease impurity (MDI), which is a feature importance measure specific to random forests. Since we choose to measure node impurity by the Gini index, MDI may also be termed Gini importance. As discussed in subsection 3-3-3, MDI covers the impact of each feature on the RF classifier's prediction individually as well as in interaction with other features. We believe that, since MDI accounts for correlation between features, it produces a more reliable ranking than LR. The ion images that map the spatial distribution and relative abundance of the sixteen top-ranking features are presented in Figures 5-6 and 5-7. These ion images are displayed using a pseudo-color scale whose brightness is indicative of the signal intensity measured at a given pixel: signal intensity correlates with the relative molecular concentration at that location. We observe that the two top-ranking features are instrumental noise and that the following fourteen top-ranking features all encode relevant biochemical information. We observe that features n^{o}3, n^{o}4, and n^{o}8, as well as features n^{o}10 and n^{o}15, have very similar mass-to-charge ratios and spatial distributions, indicating they may be isotopes.

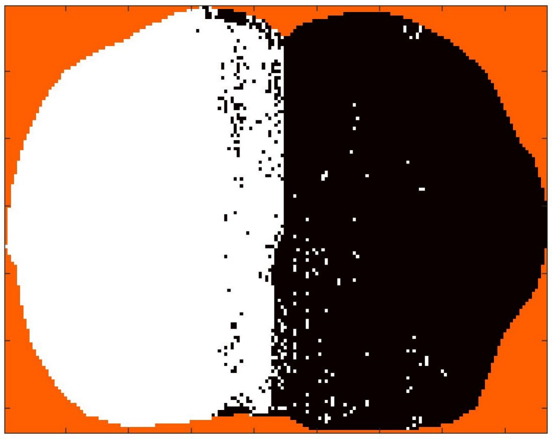
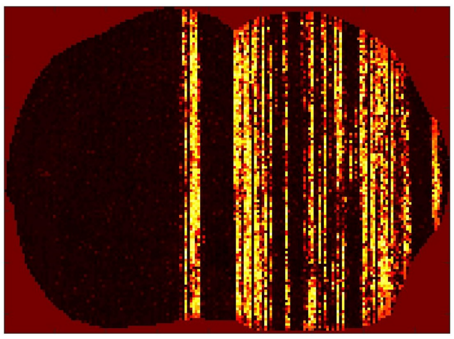
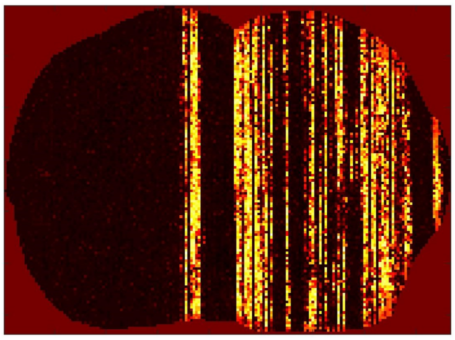
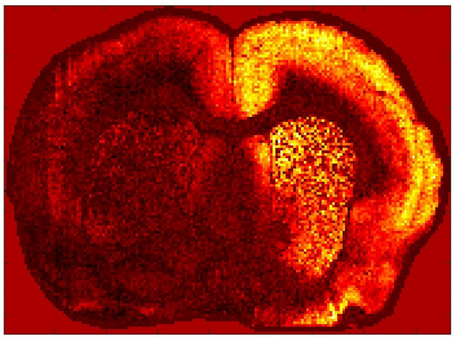
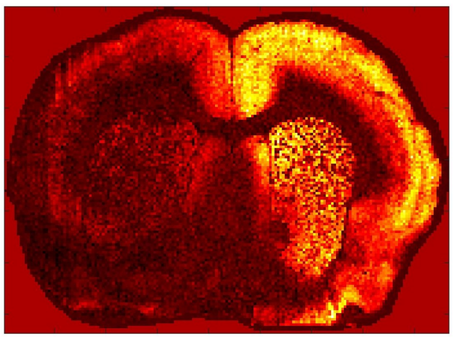


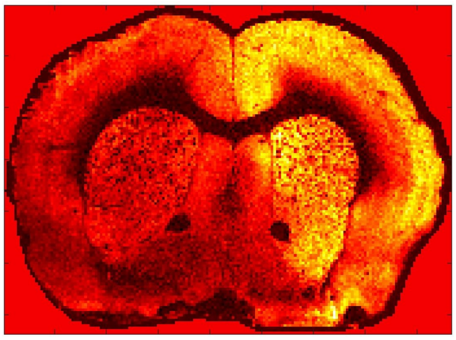
Figure 5-5: Classification of dataset n^{o}1 achieved by a random forest (RF) classifier whose number of trees is 500, whose maximum tree depth is 20, whose number of splitting candidates per node is 500, whose maximum terminal node cardinality is 10, and whose measure of node impurity is the Gini index: the positive class (white) is the diseased left brain hemisphere, whereas the negative class (black) is the healthy right brain hemisphere. The accuracy, precision and recall measured on the test set are 93.57%, 92.45% and 94.74%.

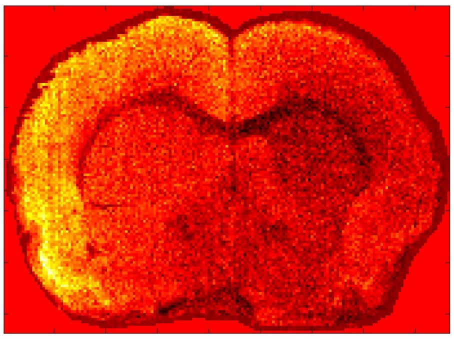

(a) Ion image of feature n^{o}1 with m/z ratio 4060.2

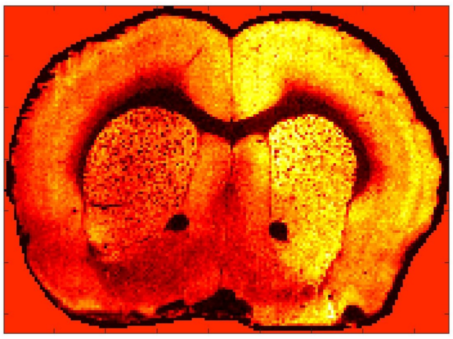

(b) Ion image of feature n^{o}2 with m/z ratio 4051.6


(c) Ion image of feature n^{o}3 with m/z ratio 1828.7


(d) Ion image of feature n^{o}4 with m/z ratio 1829.0


(e) Ion image of feature n^{o}5 with m/z ratio 5610.9


(f) Ion image of feature n^{o}6 with m/z ratio 1824.0


(g) Ion image of feature n^{o}7 with m/z ratio 5487.0

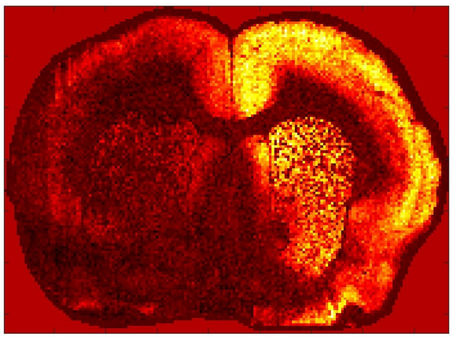
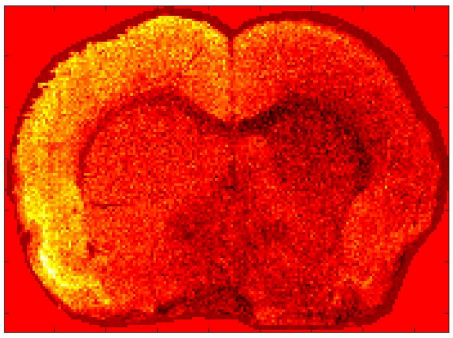
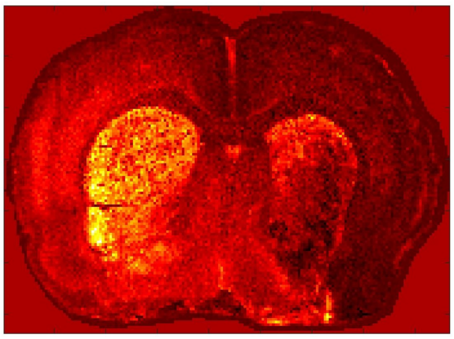
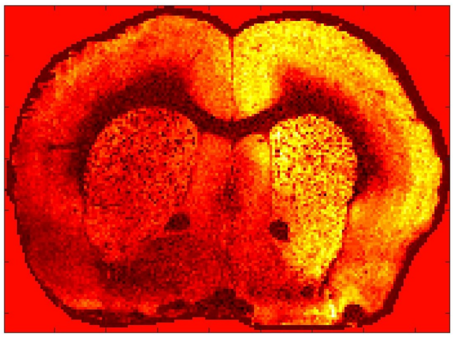
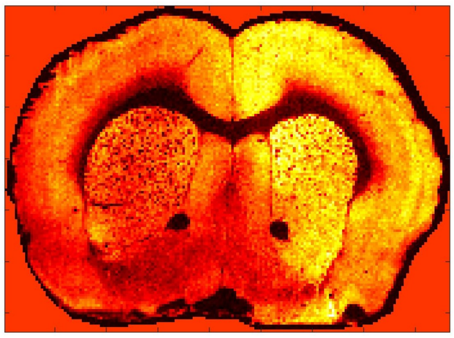

(h) Ion image of feature n^{o}8 with m/z ratio 1829.3

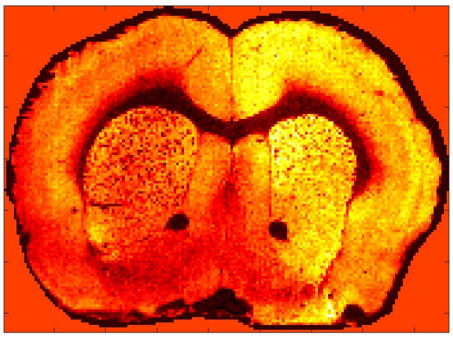
Figure 5-6: Ion images of the top-ranking features, from n^{o}1 to n^{o}8, according to the random forest classifier trained on imaging mass spectrometry dataset n^{o}1.

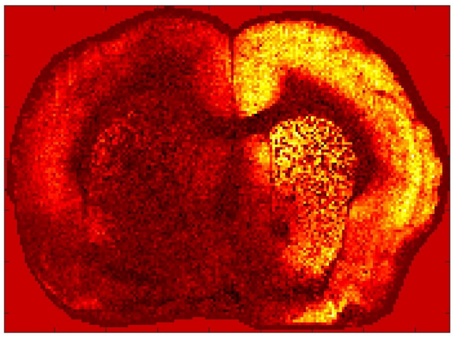

(a) Ion image of feature n^{o}9 with m/z ratio 1823.0

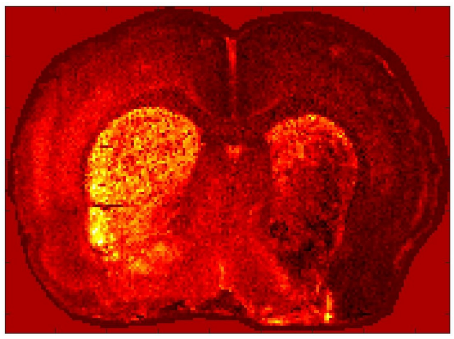

(b) Ion image of feature n^{o}10 with m/z ratio 1756.0


(c) Ion image of feature n^{o}11 with m/z ratio 5612.9


(d) Ion image of feature n^{o}12 with m/z ratio 5485.0


(e) Ion image of feature n^{o}13 with m/z ratio 5486.0


(f) Ion image of feature n^{o}14 with m/z ratio 6711.5


(g) Ion image of feature n^{o}15 with m/z ratio 1755.0

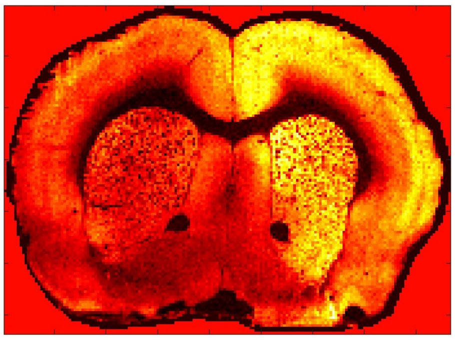

(h) Ion image of feature n^{o}16 with m/z ratio 5547.8

Figure 5-7: Ion images of the top-ranking features, from n^{o}9 to n^{o}16, according to the random forest classifier trained on imaging mass spectrometry dataset n^{o}1.

5-1-1-3 Support Vector Machines

Figure 5-8 presents the results of classifying dataset n^{o}1 with a linear support vector machine whose regularization parameter is $C = 0.01$, box constraint tolerance is 0.01 and whose Lagrange multiplier convergence precision is $\epsilon = 0.001$. The positively labeled pixels are shown in white, whereas the negatively labeled pixels are shown in black. As mentioned in subsection 3-2-3, our approach to training support vector machine (SVM) classifiers uses sequential minimal optimization (SMO). The training accuracy of the SVM classifier corresponding to Figure 5-8 is 94.92% and its testing accuracy is 92.68%. The precision and recall achieved on the testing set are 93.69% and 91.85% respectively. The ion maps of the twenty top-ranking features, as per the predictive importance measure of subsubsection 3-2-1-3, are given in Figures 5-9 and 5-10. These ion images are displayed using a pseudo-color scale whose brightness is indicative of the signal intensity measured at a given pixel: signal intensity correlates with the relative molecular concentration at that location. Unlike the two top-ranking features provided by LR and RF classifiers, the two top-ranking features according to our linear SVM classifier encode biological information. Ions n^{o}5 and n^{o}7 correspond to instrumental noise. We observe several isotopes in Figures 5-9 and 5-10: ions n^{o}1, n^{o}4 and n^{o}16; ions n^{o}3 and n^{o}9; ions n^{o}6, n^{o}11 and n^{o}14; ions n^{o}10 and n^{o}13. Note that, although we expected non-linear SVM classifiers to perform far better on complex high-dimensional imaging mass spectrometry (IMS) data than linear SVM classifiers, we observed that linear SVMs performed very well, both in terms of classification and biomarker discovery. Linear SVMs do not require extensive tuning of the kernel and its hyper-parameters: tuning non-linear SVM models on IMS data would be computationally expensive, especially since our implementation of SMO has a long computational time.

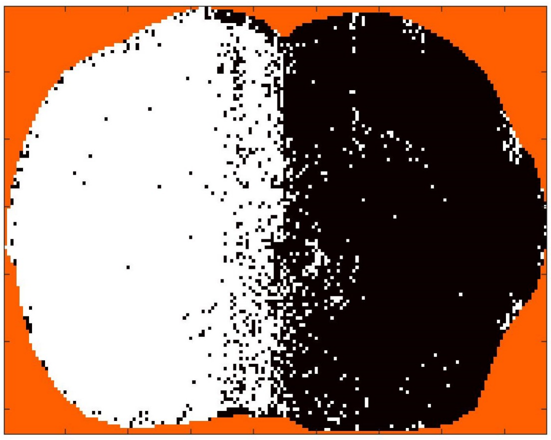
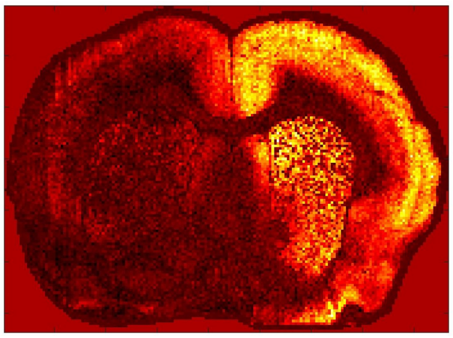
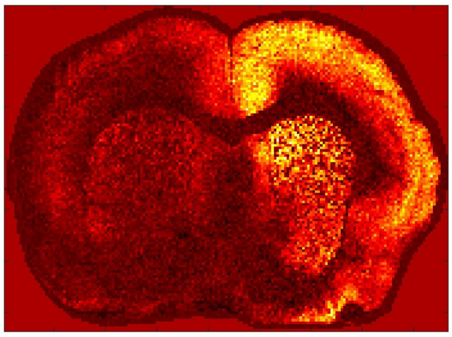
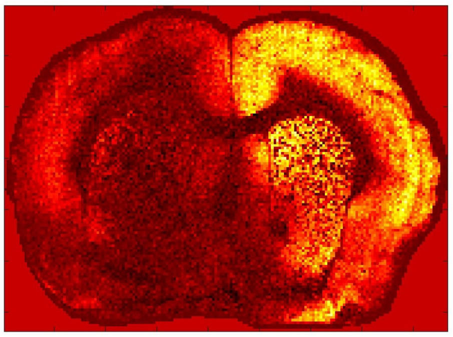
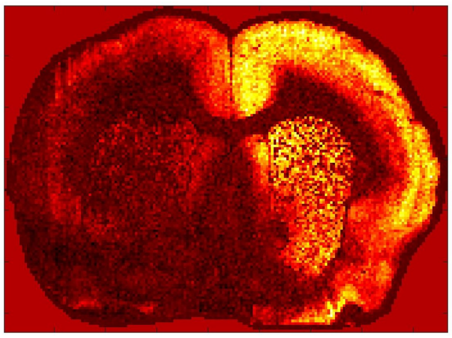


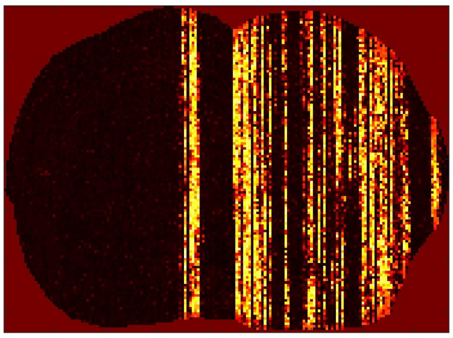
Figure 5-8: Classification of dataset n^{o}1 achieved by a linear support vector machine (SVM) classifier whose regularization parameter is $C = 0.01$: the positive class (white) is the diseased left brain hemisphere, whereas the negative class (black) is the healthy right brain hemisphere. The accuracy, precision and recall measured on the test set are 92.68%, 93.69% and 91.85%.

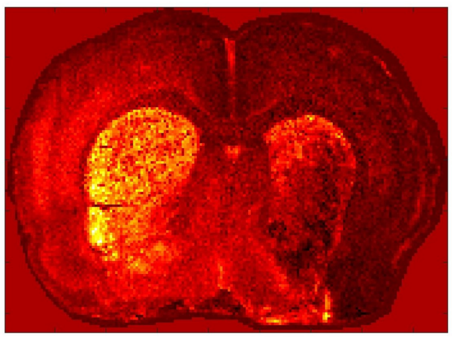

(a) Ion image of feature n^{o}1 with m/z ratio 1828.7

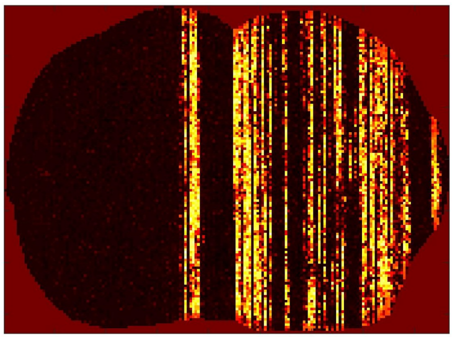

(b) Ion image of feature n^{o}2 with m/z ratio 7650.5


(c) Ion image of feature n^{o}3 with m/z ratio 6711.5


(d) Ion image of feature n^{o}4 with m/z ratio 1829.3


(e) Ion image of feature n^{o}5 with m/z ratio 4060.2


(f) Ion image of feature n^{o}6 with m/z ratio 1756.0


(g) Ion image of feature n^{o}7 with m/z ratio 4051.6

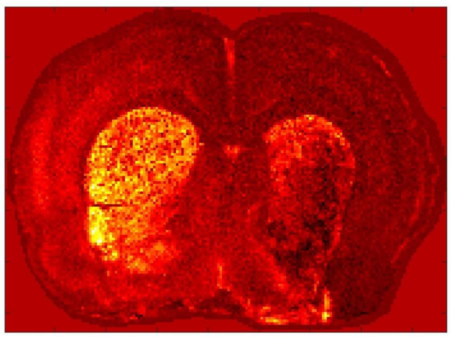
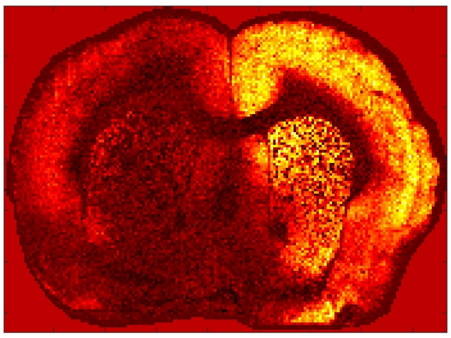
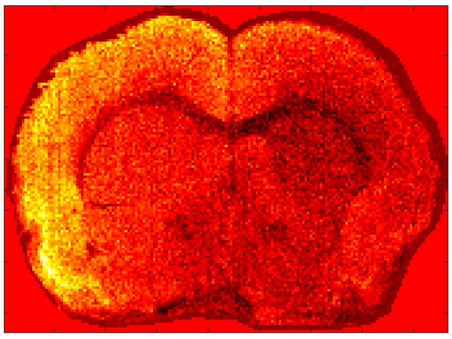
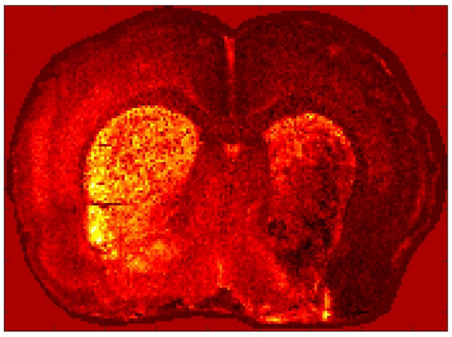
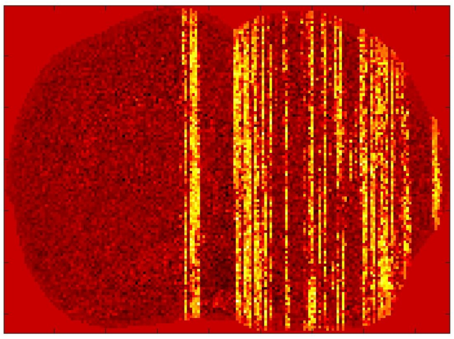

(h) Ion image of feature n^{o}8 with m/z ratio 1816.9

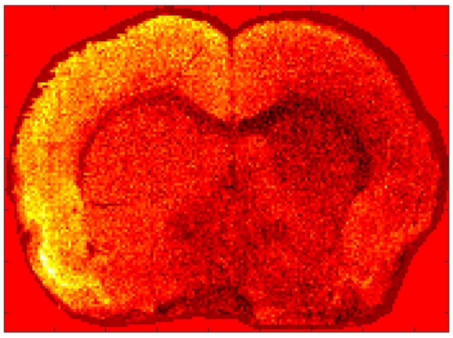
Figure 5-9: Ion images of the top-ranking features, from n^{o}1 to n^{o}8, according to the linear support vector machine classifier trained on imaging mass spectrometry dataset n^{o}1.

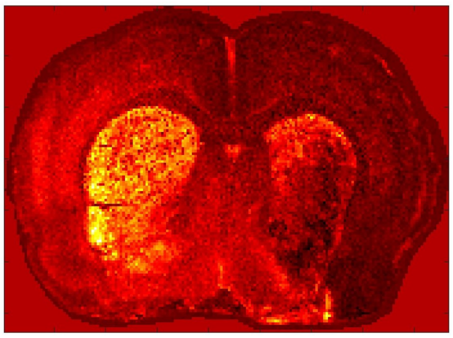

(a) Ion image of feature n^{o}9 with m/z ratio 6712.5

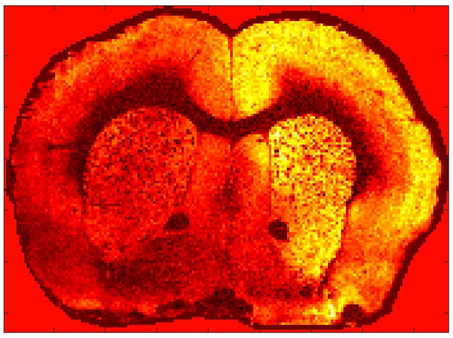

(b) Ion image of feature n^{o}10 with m/z ratio 1824.0


(c) Ion image of feature n^{o}11 with m/z ratio 1755.0


(d) Ion image of feature n^{o}12 with m/z ratio 4043.1


(e) Ion image of feature n^{o}13 with m/z ratio 1823.0


(f) Ion image of feature n^{o}14 with m/z ratio 1757.0


(g) Ion image of feature n^{o}15 with m/z ratio 5612.9

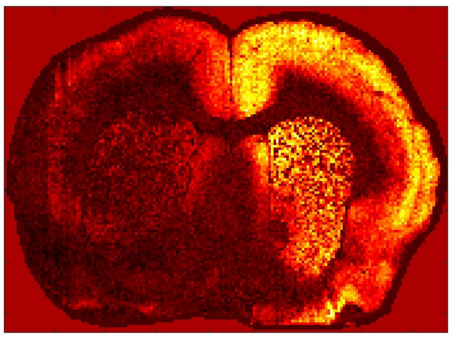

(h) Ion image of feature n^{o}16 with m/z ratio 1829.0

Figure 5-10: Ion images of the top-ranking features, from n^{o}9 to n^{o}16, according to the linear support vector machine classifier trained on imaging mass spectrometry dataset n^{o}1.

5-1-1-4 Feature Ranking for Dataset n^{o}1

To summarize, Table 5-1 lists the mass-to-charge ratios of the sixteen top-ranking features, according to the model-specific interpretability methods of the LR, linear SVM, and RF classifiers presented in subsubsections 5-1-1-1, 5-1-1-3, and 5-1-1-2.

	Mass-to-charge ratios of the sixteen top-ranking features
Logistic regression	4060.2 ; 4051.6 ; 4043.1 ; 4068.8 ; 1824.0 ; 4060.7 ; 4052.1 ; 4069.2 ; 1823.0 ; 5610.9 ; 1828.7; 2441.0 ; 1756.0 ; 1927.9 ; 2440.0 ; 4052.6
Linear support vector machine	1828.7 ; 7650.5 ; 6711.5 ; 1829.3 ; 4060.2 ; 1756.0 ; 4051.6 ; 1816.9 ; 6712.5 ; 1824.0 ; 1755.0 ; 4043.1 ; 1823.0 ; 1757.0 ; 5612.9 ; 1829.0
Random forest	4060.2 ; 4051.6 ; 1828.7 ; 1829.0 ; 5610.9 ; 1824.0 ; 5487.0 ; 1829.3 ; 1823.0 ; 1756.0 ; 5612.9 ; 5485.0 ; 5486.0 ; 6711.5 ; 1755.0 ; 5547.8

Table 5-1: Mass-to-charge ratios of the sixteen top-ranking features for recognizing the diseased left brain hemisphere from the healthy right brain hemisphere in dataset n^{o}1, using a logistic regression, a linear support vector machine, and a random forest classifier.

5-1-2 Dataset n^{o}2

5-1-2-1 Logistic regression

Five LR classifiers were developed to recognize each target organ of the mouse-pup in dataset n^{o}2. Figure 5-11 shows the results of classifying the brain from the other organs using the masks of Figure 4-3. Figure 5-13 shows the results of classifying the liver from the other organs using the masks of Figure 4-4. Figure 5-15 shows the results of classifying the heart from the other organs using the masks of Figure 4-5. Figure 5-17 shows the results of classifying the adipose tissue using the masks of Figure 4-7. Finally, Figure 5-19 shows the results of classifying the lungs from the other organs using the masks of Figure 4-6.

Validating the classification output was primarily done by visual inspection because we believe that the reported classification accuracy, precision, recall, and F-score may slightly overestimate classifier performance because of how the training and testing sets were defined. Indeed, our orange masks, from which is taken the positive class of the training and testing sets, do not exhaustively cover all pixels of each target organ. Pixels that are located on the boundaries of the target organs are not included in the training and testing sets because they are difficult to label manually. Yet it is precisely on the boundary pixels that a classifier is most at risk of making erroneous predictions.

In order to verify the decision-making process of the LR classifiers, we also present the ion maps of the eight top-ranking features. The features are ranked according to their respective predictive importance using the MDI measure that we discussed in subsubsection 3-1-2-1. Each ion image maps the spatial distribution and relative concentration of a specific biomolecule across the surface of the mouse-pup's full-body section. Ion images are displayed using a pseudo-color scale whose brightness is indicative of the signal intensity measured at a given pixel: signal intensity correlates with the relative molecular concentration at that location. We observe that our LR classifiers base their predictions on relevant biochemical patterns inherent to dataset n^{o}2. We also observe that LR classifiers are more sensitive to noise than RF classifiers, presented in subsubsection 5-1-2-2.

Recognition of the brain

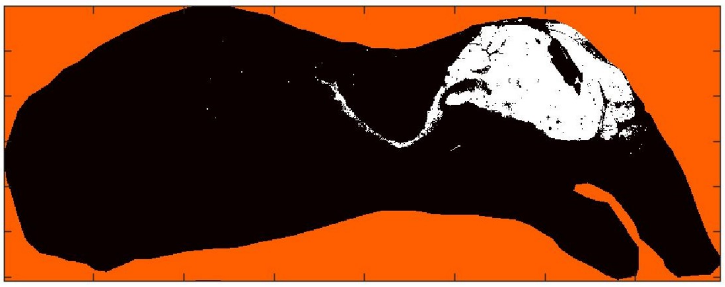
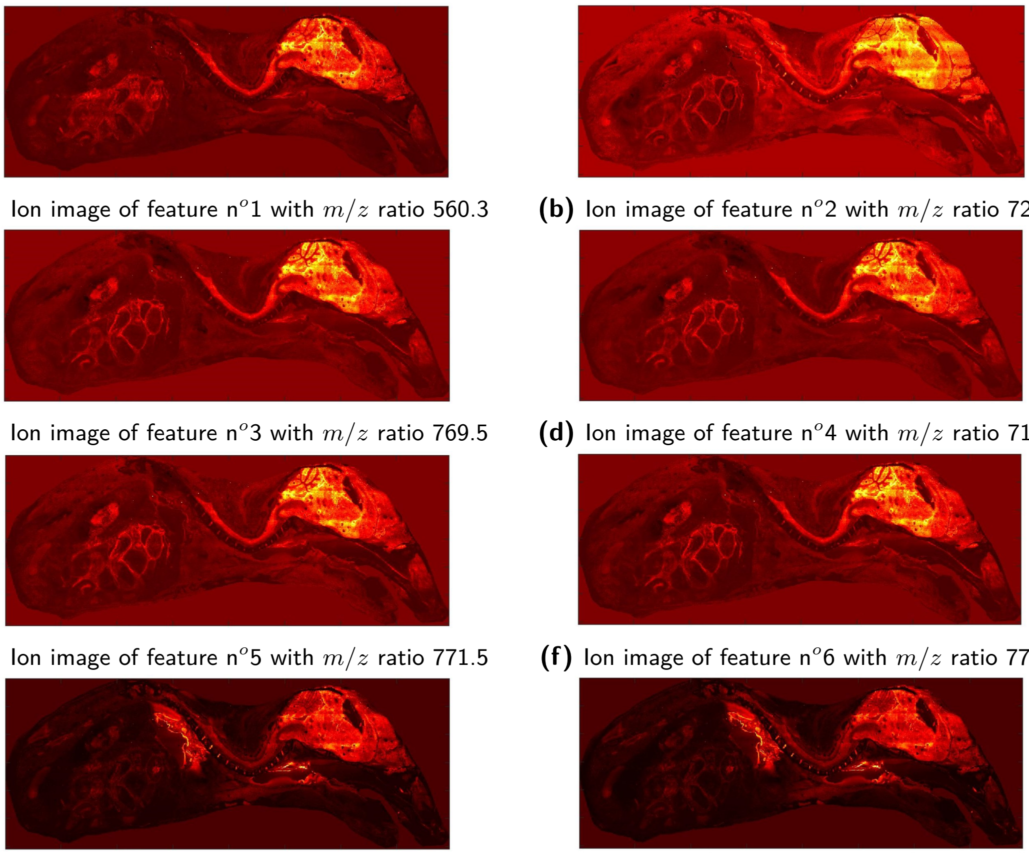


Figure 5-11: Classification of dataset n^{o}2 achieved by a logistic regression classifier whose regularization parameter is $\lambda = 900$. The positive class (white) is the predicted mouse brain, whereas the negative class (black) is predicted to be made up of the other organs. The accuracy, precision, recall, and F-score measured on the testing set are 98.83%, 98.23%, 98.89%, and 0.9856 respectively.



(a) Ion image of feature n^{o}1 with m/z ratio 560.3 **(b)** Ion image of feature n^{o}2 with m/z ratio 729.6

(c) Ion image of feature n^{o}3 with m/z ratio 769.5 **(d)** Ion image of feature n^{o}4 with m/z ratio 710.5

(e) Ion image of feature n^{o}5 with m/z ratio 771.5 **(f)** Ion image of feature n^{o}6 with m/z ratio 770.5

(g) Ion image of feature n^{o}7 with m/z ratio 711.4 **(h)** Ion image of feature n^{o}8 with m/z ratio 771.5

Figure 5-12: Ion images of the top-ranking features, from n^{o}1 to n^{o}8, according to a logistic regression classifier trained to recognize the mouse brain from the other organs in imaging mass spectrometry dataset n^{o}2.

Recognition of the liver

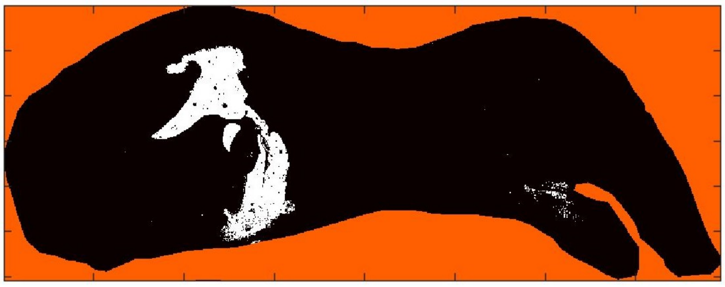
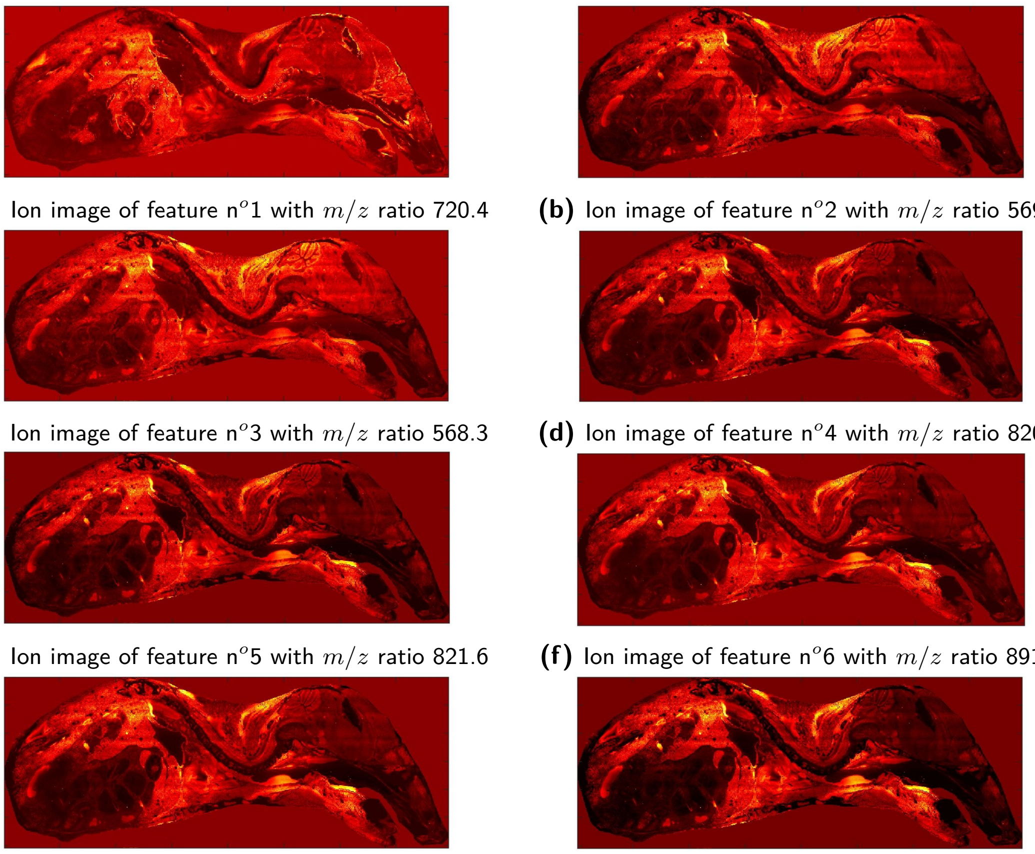


Figure 5-13: Classification of dataset n^{o}2 achieved by a logistic regression classifier a regularization parameter of $\lambda = 900$. The positive class (white) is the predicted mouse liver, whereas the negative class (black) is predicted to be made up of the other organs. The accuracy, precision, recall, and F-score measured on the testing set are 98.49%, 99.01%, 97.20%, and 0.9810.



(a) Ion image of feature n^{o}1 with m/z ratio 720.4 **(b)** Ion image of feature n^{o}2 with m/z ratio 569.3

(c) Ion image of feature n^{o}3 with m/z ratio 568.3 **(d)** Ion image of feature n^{o}4 with m/z ratio 820.6

(e) Ion image of feature n^{o}5 with m/z ratio 821.6 **(f)** Ion image of feature n^{o}6 with m/z ratio 891.6

(g) Ion image of feature n^{o}7 with m/z ratio 892.6 **(h)** Ion image of feature n^{o}8 with m/z ratio 806.5

Figure 5-14: Ion images of the top-ranking features, from n^{o}1 to n^{o}8, according to a logistic regression classifier trained to recognize the mouse liver from the other organs in imaging mass spectrometry dataset n^{o}2.

Recognition of the heart

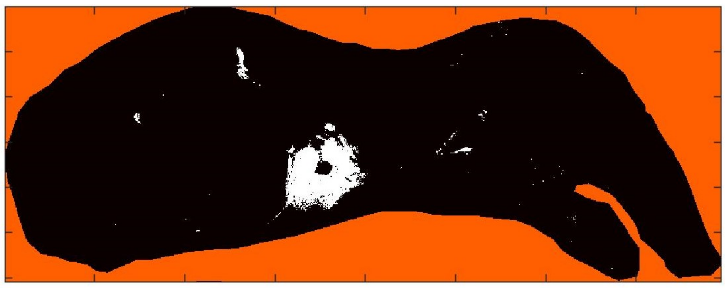
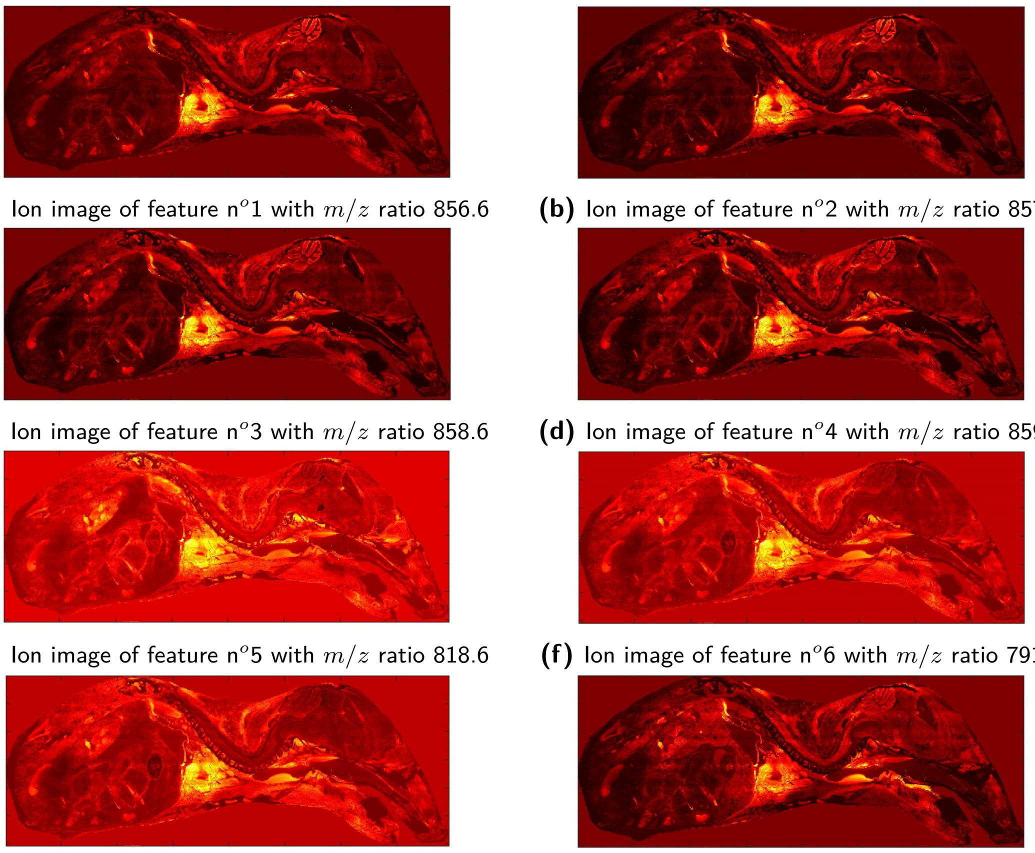


Figure 5-15: Classification of dataset n^{o}2 achieved by a logistic regression classifier with regularization parameter $\lambda = 1000$. The positive class (white) is the predicted mouse heart, whereas the negative class (black) is predicted to be made up of the other organs. The accuracy, precision, recall, and F-score measured on the testing set are 99.24%, 99.47%, 98.60%, and 0.9904 respectively.



(a) Ion image of feature n^{o}1 with m/z ratio 856.6
(b) Ion image of feature n^{o}2 with m/z ratio 857.6
(c) Ion image of feature n^{o}3 with m/z ratio 858.6
(d) Ion image of feature n^{o}4 with m/z ratio 859.6
(e) Ion image of feature n^{o}5 with m/z ratio 818.6
(f) Ion image of feature n^{o}6 with m/z ratio 791.5
(g) Ion image of feature n^{o}7 with m/z ratio 790.5
(h) Ion image of feature n^{o}8 with m/z ratio 769.4

Figure 5-16: Ion images of the top-ranking features, from n^{o}1 to n^{o}8, according to a logistic regression classifier trained to recognize the mouse heart from the other organs in imaging mass spectrometry dataset n^{o}2.

Recognition of the dorsal adipose tissue

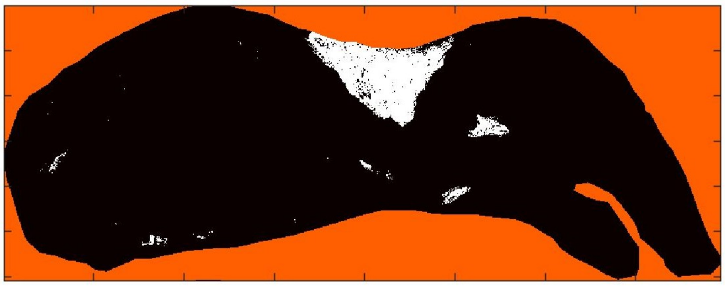
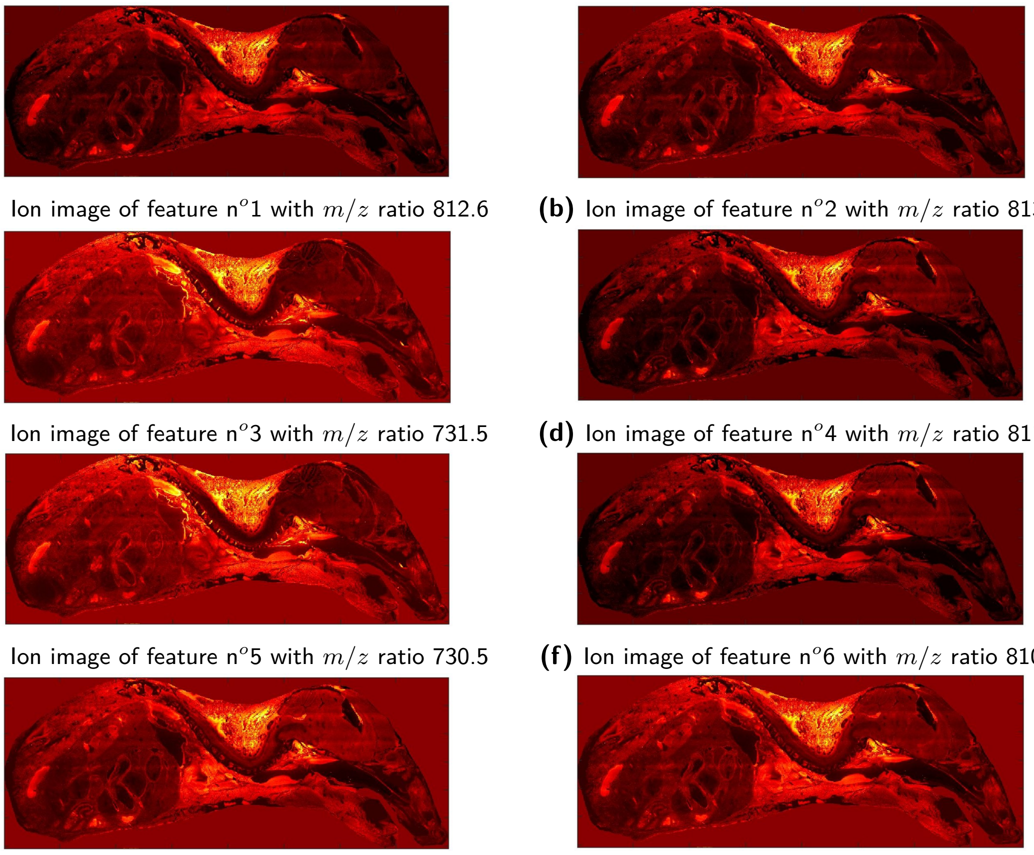


Figure 5-17: Classification of dataset n^{o}2 achieved by a logistic regression classifier with regularization parameter $\lambda = 900$. The positive class (white) is the predicted mouse's dorsal adipose tissue, whereas the negative class (black) is predicted to be made up of the other organs. The accuracy, precision, recall, and F-score measured on the testing set are 98.16%, 97.58%, 97.58%, and 0.9758 respectively.



(a) Ion image of feature n^{o}1 with m/z ratio 812.6 **(b)** Ion image of feature n^{o}2 with m/z ratio 813.6

(c) Ion image of feature n^{o}3 with m/z ratio 731.5 **(d)** Ion image of feature n^{o}4 with m/z ratio 811.6

(e) Ion image of feature n^{o}5 with m/z ratio 730.5 **(f)** Ion image of feature n^{o}6 with m/z ratio 810.6

(g) Ion image of feature n^{o}7 with m/z ratio 896.7 **(h)** Ion image of feature n^{o}8 with m/z ratio 897.7

Figure 5-18: Ion images of the top-ranking features, from n^{o}1 to n^{o}8, according to a logistic regression classifier trained to recognize the mouse's dorsal adipose tissue from the other organs in imaging mass spectrometry dataset n^{o}2.

Recognition of the lungs

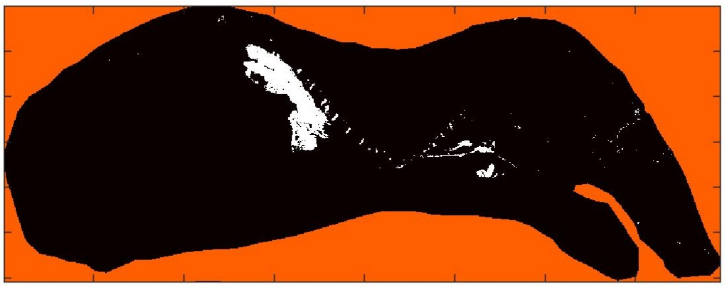
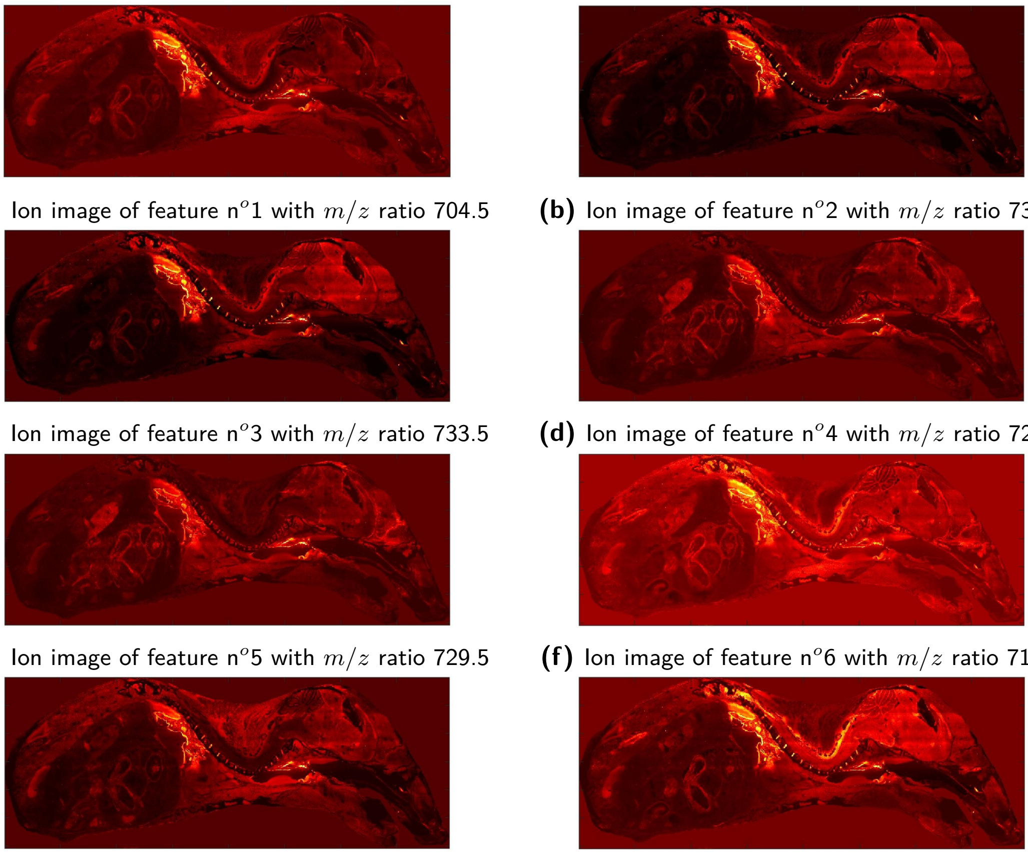


Figure 5-19: Classification of dataset n^{o}2 achieved by a logistic regression with regularization parameter $\lambda = 800$. The positive class (white) is the predicted mouse lungs, whereas the negative class (black) is predicted to be made up of the other organs. The accuracy, precision, recall, and F-score measured on the testing set are 98.49%, 98.80%, 97.39%, and 0.9809 respectively.



(a) Ion image of feature n^{o}1 with m/z ratio 704.5 **(b)** Ion image of feature n^{o}2 with m/z ratio 732.5

(c) Ion image of feature n^{o}3 with m/z ratio 733.5 **(d)** Ion image of feature n^{o}4 with m/z ratio 728.5

(e) Ion image of feature n^{o}5 with m/z ratio 729.5 **(f)** Ion image of feature n^{o}6 with m/z ratio 718.5

(g) Ion image of feature n^{o}7 with m/z ratio 754.5 **(h)** Ion image of feature n^{o}8 with m/z ratio 817.6

Figure 5-20: Ion images of the top-ranking features, from n^{o}1 to n^{o}8, according to a logistic regression classifier trained to recognize the mouse lungs from the other organs in imaging mass spectrometry dataset n^{o}2.

5-1-2-2 Random forest

Five RF classifiers were developed to recognize each target organ of the mouse-pup in dataset n^{o}2. Figure 5-21 shows the results of classifying the brain from the other organs using the masks of Figure 4-3. Figure 5-23 shows the results of classifying the liver from the other organs using the masks of Figure 4-4. Figure 5-25 shows the results of classifying the heart from the other organs using the masks of Figure 4-5. Figure 5-27 shows the results of classifying the adipose tissue using the masks of Figure 4-7. Finally, Figure 5-29 shows the results of classifying the lungs from the other organs using the masks of Figure 4-6.

Validating the classification output was primarily done by visual inspection because we believe that the reported classification accuracy, precision, recall, and F-score may slightly overestimate classifier performance because of how the training and testing sets were defined. Indeed, our orange masks, from which is taken the positive class of the training and testing sets, do not exhaustively cover all pixels of each target organ. Pixels that are located on the boundaries of the target organs are not included in the training and testing sets because they are difficult to label manually. Yet it is precisely on the boundary pixels that a classifier is most a risk of making erroneous predictions.

In order to verify the decision-making process of the RF classifiers, we also present the ion maps of the eight top-ranking features. The features are ranked according to their respective predictive importance using the MDI measure that we discussed in subsection 3-3-3. Each ion image maps the spatial distribution and relative concentration of a specific bio-molecule across the surface of the mouse-pup's full-body section. Ion images are displayed using a pseudo-color scale whose brightness is indicative of the signal intensity measured at a given pixel: signal intensity correlates with the relative molecular concentration at that location. We observe that our RF classifiers base their predictions on relevant biochemical patterns inherent to dataset n^{o}2. We also observe that RF models are less prone to overfitting than LR models, whose classification results are presented in subsubsection 5-1-2-1.

Recognition of the brain

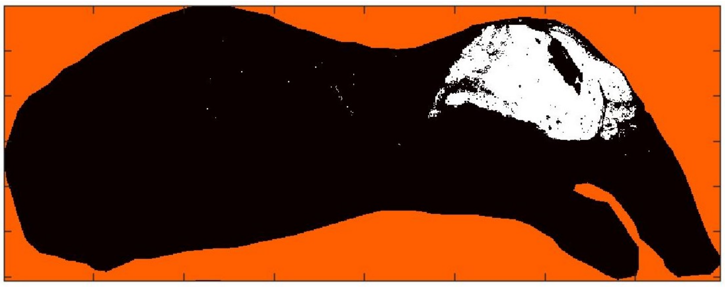
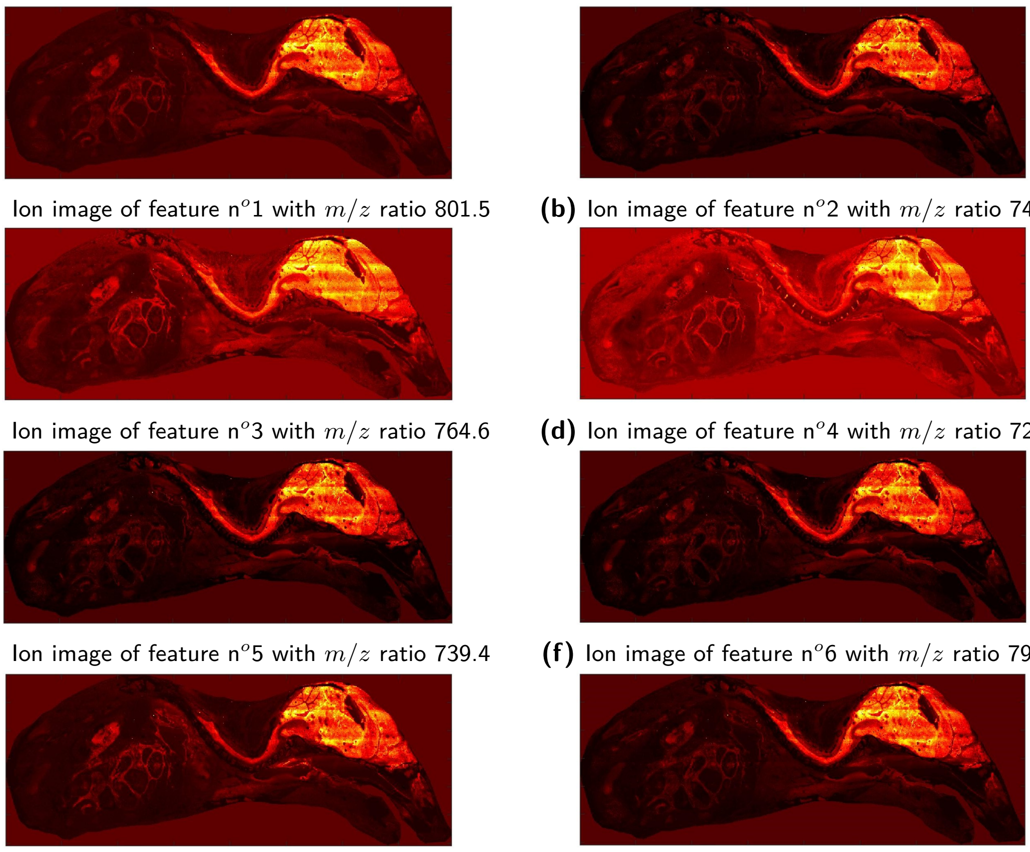


Figure 5-21: Classification of dataset n^{o}2 by a random forest classifier whose number of trees is 300, whose maximum tree depth is 10, whose number of splitting candidates per node is 60, whose maximum terminal node cardinality is 10, and whose measure of node impurity is the Gini index. The positive class (white) is the predicted mouse brain, whereas the negative class (black) is predicted to be made up of the other organs. The accuracy, precision, recall, and F-score measured on the testing set are 99.59%, 99.42%, 99.54%, and 0.9948 respectively.



(a) Ion image of feature n^{o}1 with m/z ratio 801.5 **(b)** Ion image of feature n^{o}2 with m/z ratio 740.4

(c) Ion image of feature n^{o}3 with m/z ratio 764.6 **(d)** Ion image of feature n^{o}4 with m/z ratio 729.6

(e) Ion image of feature n^{o}5 with m/z ratio 739.4 **(f)** Ion image of feature n^{o}6 with m/z ratio 798.5

(g) Ion image of feature n^{o}7 with m/z ratio 714.4 **(h)** Ion image of feature n^{o}8 with m/z ratio 800.5

Figure 5-22: Ion images of the top-ranking features, from n^{o}1 to n^{o}8, according to a random forest classifier trained to recognize the mouse brain from the other organs in dataset n^{o}2.

Recognition of the liver

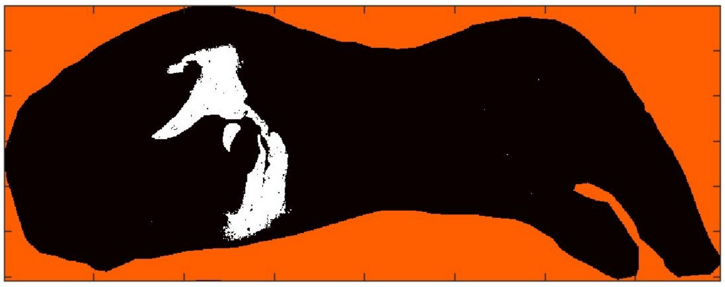
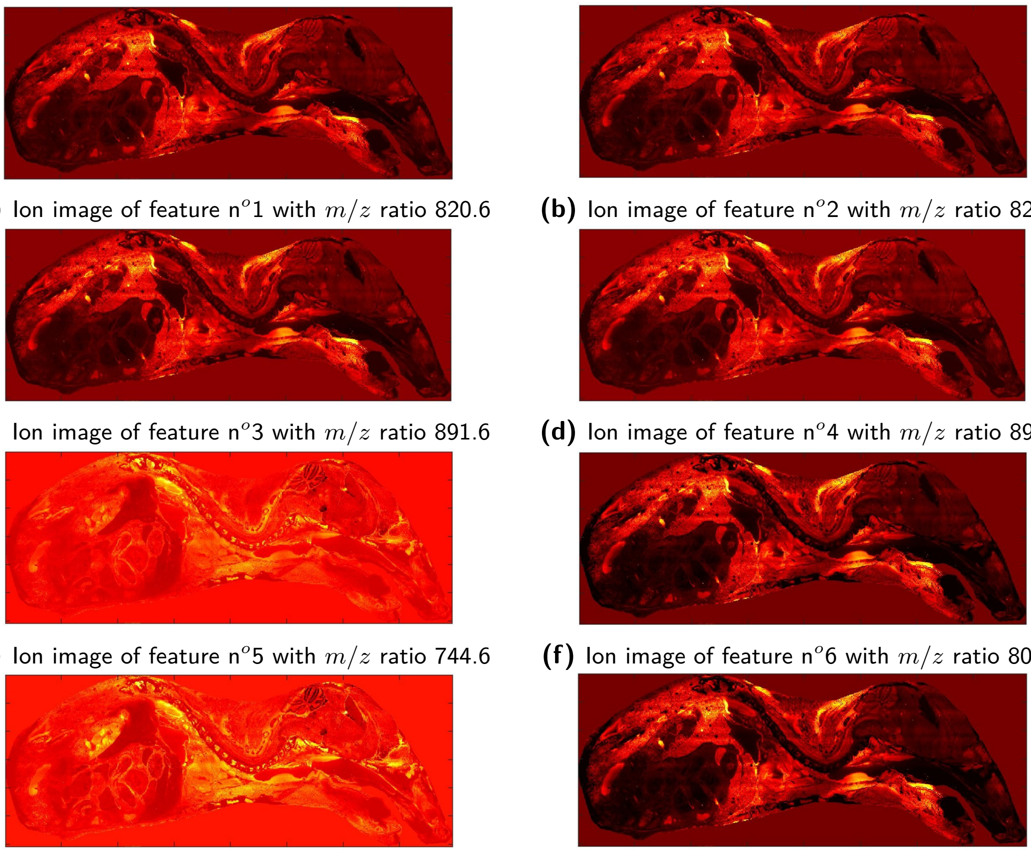


Figure 5-23: Classification of dataset n^{o}2 achieved by a random forest classifier whose number of trees is 300, whose maximum tree depth is 10, whose number of splitting candidates per node is 60, whose maximum terminal node cardinality is 10, and whose measure of node impurity is the Gini index. The positive class (white) is the predicted mouse liver, whereas the negative class (black) is predicted to be made up of the other organs. The accuracy, precision, recall, and F-score measured on the testing set are 99.48%, 99.33%, 99.33%, and 0.9933 respectively.



(a) Ion image of feature n^{o}1 with m/z ratio 820.6 **(b)** Ion image of feature n^{o}2 with m/z ratio 821.6

(c) Ion image of feature n^{o}3 with m/z ratio 891.6 **(d)** Ion image of feature n^{o}4 with m/z ratio 892.6

(e) Ion image of feature n^{o}5 with m/z ratio 744.6 **(f)** Ion image of feature n^{o}6 with m/z ratio 807.5

(g) Ion image of feature n^{o}7 with m/z ratio 768.6 **(h)** Ion image of feature n^{o}8 with m/z ratio 806.5

Figure 5-24: Ion images of the top-ranking features, from n^{o}1 to n^{o}8, according to a random forest classifier trained to recognize the mouse liver from the other organs in imaging mass spectrometry dataset n^{o}2.

Recognition of the heart

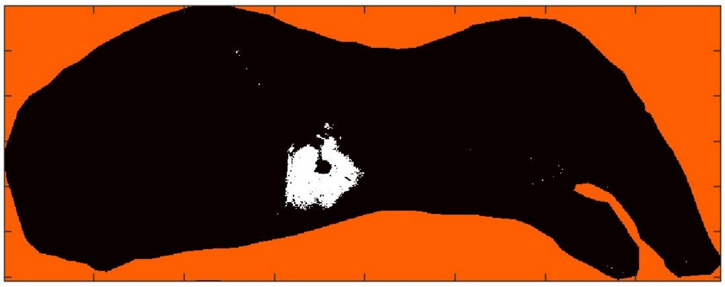
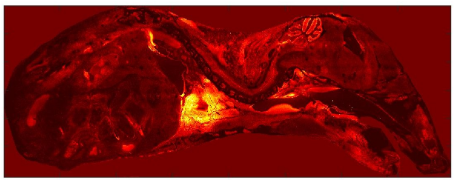
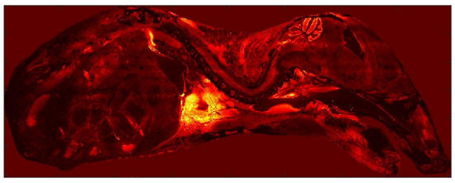


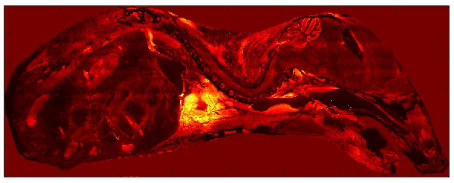
Figure 5-25: Classification of dataset n^{o}2 achieved by a random forest classifier whose number of trees is 300, whose maximum tree depth is 10, whose number of splitting candidates per node is 60, whose maximum terminal node cardinality is 10, and whose measure of node impurity is the Gini index. The positive class (white) is the predicted mouse heart, whereas the negative class (black) is predicted to be made up of the other organs. The accuracy, precision, recall, and F-score measured on the testing set are 99.24%, 99.64%, 98.38%, and 0.9900 respectively.



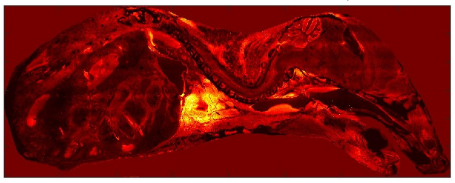
(a) Ion image of feature n^{o}1 with m/z ratio 856.6



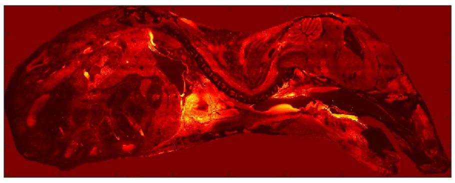
(b) Ion image of feature n^{o}2 with m/z ratio 857.6



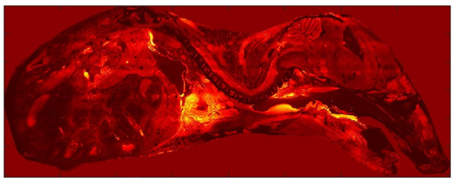
(c) Ion image of feature n^{o}3 with m/z ratio 858.6



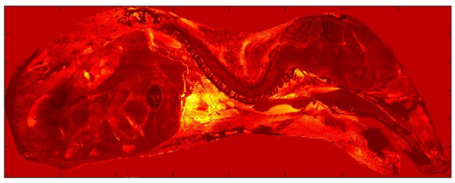
(d) Ion image of feature n^{o}4 with m/z ratio 859.6



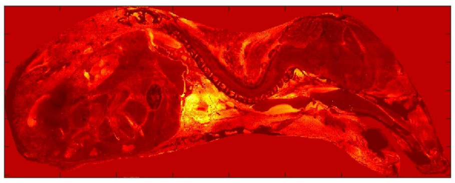
(e) Ion image of feature n^{o}5 with m/z ratio 828.5



(f) Ion image of feature n^{o}6 with m/z ratio 829.5



(g) Ion image of feature n^{o}7 with m/z ratio 790.5



(h) Ion image of feature n^{o}8 with m/z ratio 791.5

Figure 5-26: Ion images of the top-ranking features, from n^{o}1 to n^{o}8, according to a random forest classifier trained to recognize the mouse heart from the other organs in imaging mass spectrometry dataset n^{o}2.

Recognition of the dorsal adipose tissue

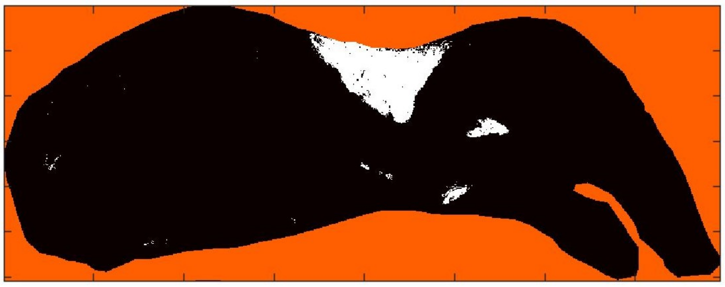
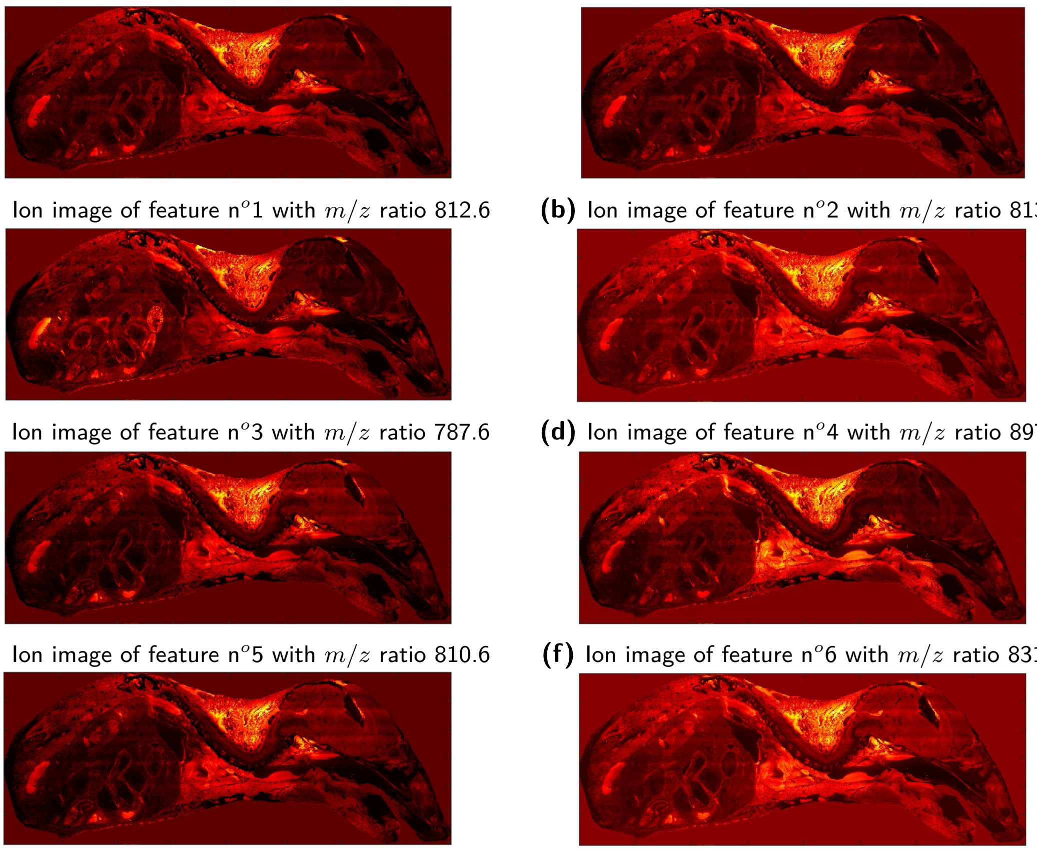


Figure 5-27: Classification of dataset n^{o}2 achieved by a random forest classifier whose number of trees is 300, whose maximum tree depth is 10, whose number of splitting candidates per node is 60, whose maximum terminal node cardinality is 10, and whose measure of node impurity is the Gini index. The positive class (white) is the predicted mouse's dorsal adipose tissue, whereas the negative class (black) is predicted to be made up of the other organs. The accuracy, precision, recall, and F-score measured on the testing set are 98.99%, 98.84%, 98.61%, and 0.9873 respectively.



(a) Ion image of feature n^{o}1 with m/z ratio 812.6 **(b)** Ion image of feature n^{o}2 with m/z ratio 813.6

(c) Ion image of feature n^{o}3 with m/z ratio 787.6 **(d)** Ion image of feature n^{o}4 with m/z ratio 897.7

(e) Ion image of feature n^{o}5 with m/z ratio 810.6 **(f)** Ion image of feature n^{o}6 with m/z ratio 831.5

(g) Ion image of feature n^{o}7 with m/z ratio 811.6 **(h)** Ion image of feature n^{o}8 with m/z ratio 895.7

Figure 5-28: Ion images of the top-ranking features, from n^{o}1 to n^{o}8, according to a random forest classifier trained to recognize the mouse's dorsal adipose tissue from the other organs in imaging mass spectrometry dataset n^{o}2.

Recognition of the lungs

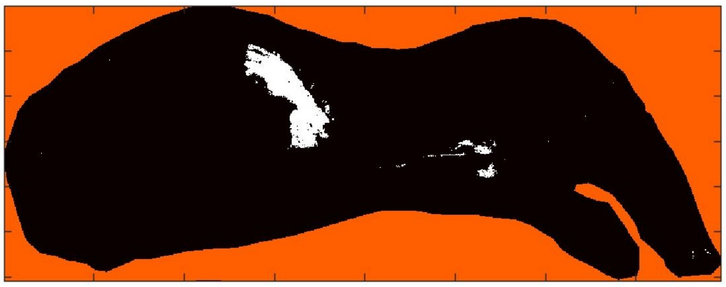
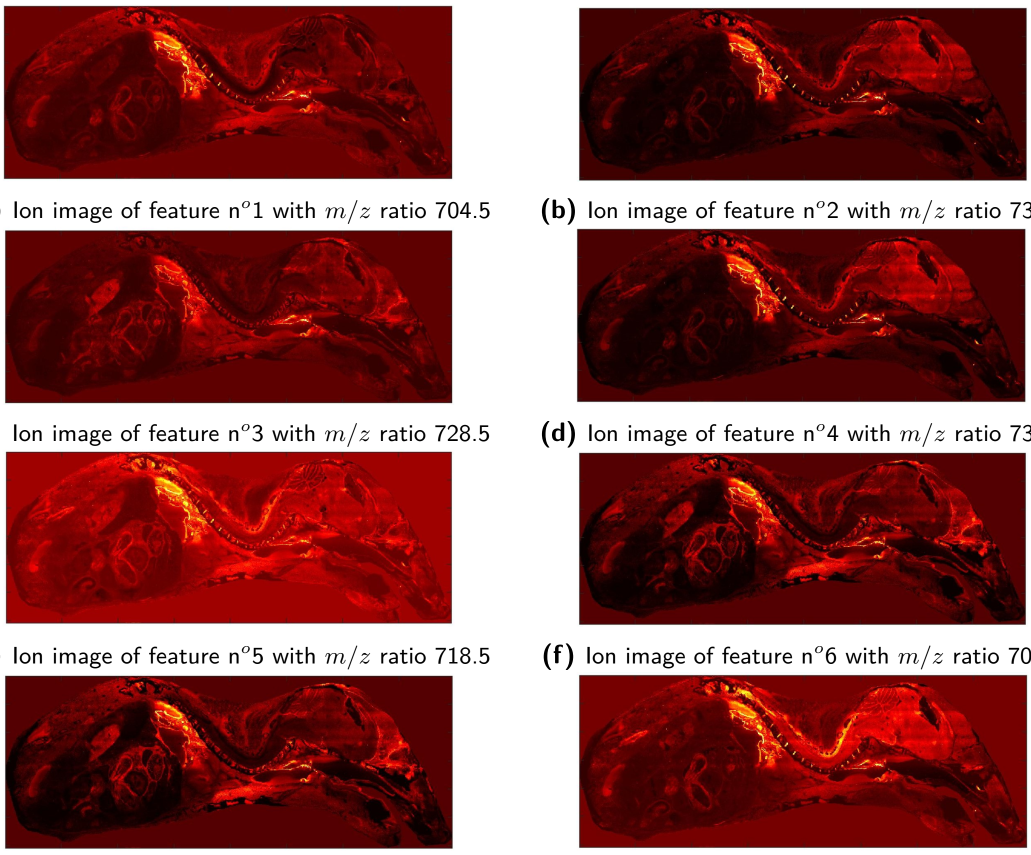


Figure 5-29: Classification of dataset n^{o}2 achieved by a random forest classifier whose number of trees is 300, whose maximum tree depth is 10, whose number of splitting candidates per node is 60, whose maximum terminal node cardinality is 10, and whose measure of node impurity is the Gini index. The positive class (white) is the predicted mouse lungs, whereas the negative class (black) is predicted to be made up of the other organs. The accuracy, precision, recall, and F-score measured on the testing set are 99.53%, 99.08%, 99.77%, and 0.9942 respectively.



(a) Ion image of feature n^{o}1 with m/z ratio 704.5 **(b)** Ion image of feature n^{o}2 with m/z ratio 732.5

(c) Ion image of feature n^{o}3 with m/z ratio 728.5 **(d)** Ion image of feature n^{o}4 with m/z ratio 733.5

(e) Ion image of feature n^{o}5 with m/z ratio 718.5 **(f)** Ion image of feature n^{o}6 with m/z ratio 706.5

(g) Ion image of feature n^{o}7 with m/z ratio 708.5 **(h)** Ion image of feature n^{o}8 with m/z ratio 817.6

Figure 5-30: Ion images of the top-ranking features, from n^{o}1 to n^{o}8, according to a random forest classifier trained to recognize the mouse lungs from the other organs in imaging mass spectrometry dataset n^{o}2.

5-1-2-3 Support Vector Machines

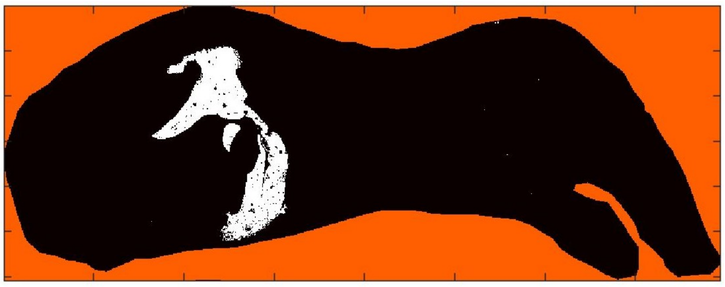
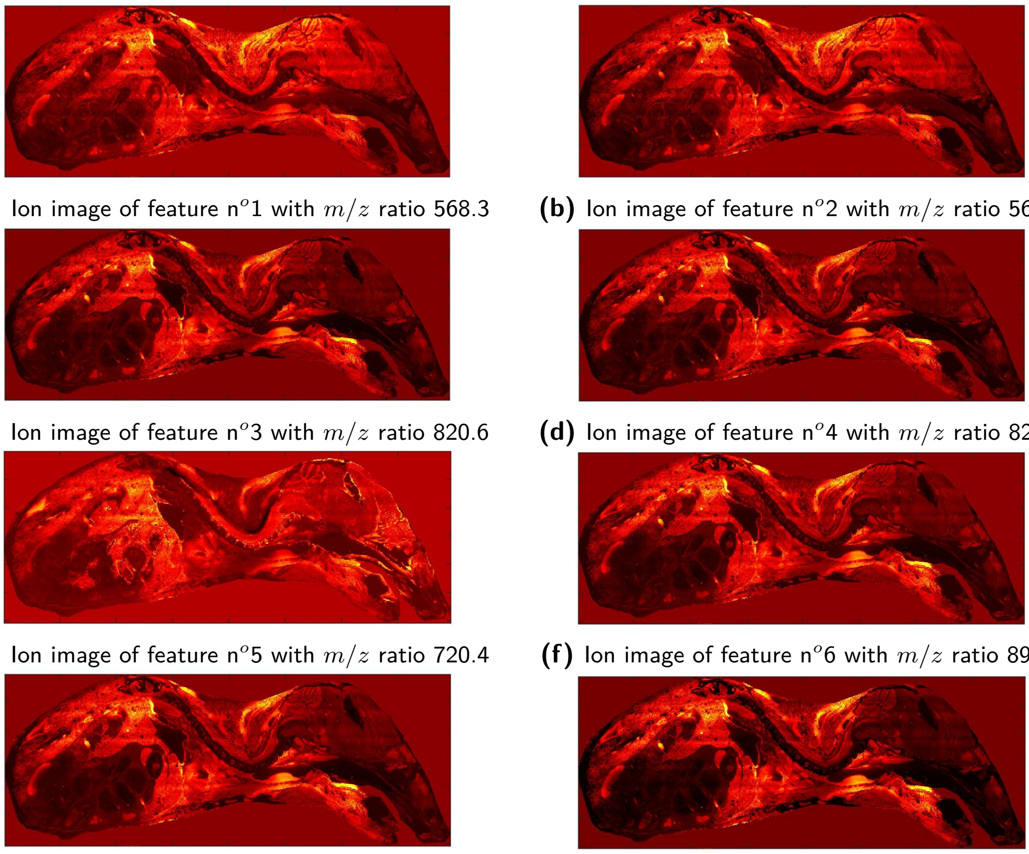


Figure 5-31: Classification of dataset n^{o}2 achieved by a linear support vector machine (SVM) classifier whose regularization parameter is $C = 0.01$: the positive class (white) is the predicted mouse liver, whereas the negative class (black) is predicted to be made of the other organs. The accuracy, precision, recall, and F-score measured on the test set are 98.38%, 98.86%, 97.11% and 0.9798 respectively.



(a) Ion image of feature n^{o}1 with m/z ratio 568.3
(b) Ion image of feature n^{o}2 with m/z ratio 569.3
(c) Ion image of feature n^{o}3 with m/z ratio 820.6
(d) Ion image of feature n^{o}4 with m/z ratio 821.6
(e) Ion image of feature n^{o}5 with m/z ratio 720.4
(f) Ion image of feature n^{o}6 with m/z ratio 892.6
(g) Ion image of feature n^{o}7 with m/z ratio 891.6
(h) Ion image of feature n^{o}8 with m/z ratio 807.5

Figure 5-32: Ion images of the top-ranking features, from n^{o}1 to n^{o}8, according to a linear support vector machine classifier trained to recognize the mouse liver from the other organs in imaging mass spectrometry dataset n^{o}2.

As explained in section 4-3, the computational cost of training support vector machines makes it impractical to train SVM classifiers to recognize all five target organs of the mouse-pup in dataset n^{o}2. We nevertheless present the results of using a linear SVM classifier to recognize the liver from the other organs.

Figure 5-31 presents the results of classifying dataset n^{o}2 with a linear support vector machine whose regularization parameter is $C = 0.01$, box constraint tolerance is 0.01 and whose Lagrange multiplier convergence precision is $\epsilon = 0.001$. The positively labeled pixels are shown in white, whereas the negatively labeled pixels are shown in black. The training accuracy of the SVM classifier corresponding to Figure 5-8 is 98.43% and its testing accuracy is 98.38%. The precision and recall achieved on the testing set are 98.86% and 97.11% respectively. The ion maps of the eight top-ranking features, as per the predictive importance measure of subsubsection 3-2-1-3, are given in Figure 5-32. These ion images are displayed using a pseudo-color scale whose brightness is indicative of the signal intensity measured at a given pixel: signal intensity correlates with the relative molecular concentration at that location.

5-1-2-4 Feature Ranking for Dataset n^{o}2

The following tables list the mass-to-charge ratios of the eight top-ranking features, according to the model-specific interpretability methods of the LR and RF classifiers developed in subsubsections 5-1-2-1 and 5-1-2-2 respectively. We observe a slight variability between interpretability methods. Table 5-2 corresponds to the recognition of the mouse-pup's brain from its other organs. Table 5-3 corresponds to the recognition of the mouse-pup's liver from its other organs. Table 5-4 corresponds to the recognition of the mouse-pup's heart from its other organs. Table 5-5 corresponds to the recognition of the mouse-pup's dorsal adipose tissue from its other organs. Table 5-6 corresponds to the recognition of the mouse-pup's lungs from its other organs.

	Mass-to-charge ratios of the eight top-ranking features
Logistic regression	560.3 ; 729.6; 769.5 ; 710.5 ; 771.5 ; 770.5 ; 711.4 ; 771.5
Random forest	801.5 ; 740.4 ; 764.6 ; 729.6 ; 739.4 ; 798.5 ; 714.4 ; 800.5

Table 5-2: Mass-to-charge ratios of the eight top-ranking features for recognizing the mouse-pup's brain from its other organs in dataset n^{o}2, using a logistic regression and a random forest classifier.

	Mass-to-charge ratios of the eight top-ranking features
Logistic regression	720.4 ; 569.3 ; 568.3 ; 820.6 ; 821.6 ; 891.6 ; 892.6 ; 806.5
Random forest	820.6 ; 821.6 ; 891.6 ; 892.6 ; 744.6 ; 807.5 ; 768.6 ; 806.5

Table 5-3: Mass-to-charge ratios of the eight top-ranking features for recognizing the mouse-pup's liver from its other organs in dataset n^{o}2, using a logistic regression and a random forest classifier.

	Mass-to-charge ratios of the eight top-ranking features
Logistic regression	856.6 ; 857.6 ; 858.6 ; 859.6 ; 818.6 ; 791.5 ; 790.5 ; 769.4
Random forest	856.6 ; 857.6 ; 858.6 ; 859.6 ; 828.5 ; 829.5 ; 790.5 ; 791.5

Table 5-4: Mass-to-charge ratios of the eight top-ranking features for recognizing the mouse-pup's heart from its other organs in dataset n^{o}2, using a logistic regression and a random forest classifier.

	Mass-to-charge ratios of the eight top-ranking features
Logistic regression	812.6 ; 813.6 ; 731.5 ; 811.6 ; 730.5 ; 810.6 ; 896.7 ; 897.7
Random forest	812.6 ; 813.6 ; 787.6 ; 897.7 ; 810.6 ; 831.5 ; 811.6 ; 895.7

Table 5-5: Mass-to-charge ratios of the eight top-ranking features for recognizing the mouse-pup's dorsal adipose tissue from its other organs in dataset n^{o}2, using a logistic regression and a random forest classifier.

	Mass-to-charge ratios of the eight top-ranking features
Logistic regression	704.5 ; 732.5 ; 733.5 ; 728.5 ; 729.5 ; 718.5 ; 754.5 ; 817.6
Random forest	704.5 ; 732.5 ; 728.5 ; 733.5 ; 718.5 ; 706.5 ; 708.5 ; 817.6

Table 5-6: Mass-to-charge ratios of the eight top-ranking features for recognizing the mouse-pup's lungs from its other organs in dataset n^{o}2, using a logistic regression and a random forest classifier.

5-1-3 Classifier Comparison

Table 5-7 summarizes some of the advantages and disadvantages of using LR, linear and non-linear SVM and RF models for the classification and feature ranking of IMS data. Note that the following observations were made on datasets n^{o}1 and n^{o}2 using our MATLAB implementations of the four different supervised machine learning algorithms. We conclude that RF models are better than LR and SVM models for classifying IMS data. As discussed in section 3-3, RF models are non-linear, non-parametric, intrinsically interpretable models that perform accurate classification of IMS data. We observe that training an RF model on large datasets is far less computationally costly than training an SVM model: our implementation of RF parallelizes the training of decision trees on bootstrapped datasets. Our implementation of RF is less prone to overfitting high-dimensionality data than LR. Unlike LR and SVM, random forests are scale-invariant so unit-variance scaling of the features prior to training is not necessary. As mentioned in section 4-3, we would prefer not having to scale the features of IMS data. Finally, RF classifiers provide an estimate of feature predictive importance thanks to the MDI measure, which is presented in subsection 3-3-3. In addition to demonstrating high predictive performance, random forests provide the user with insight into the biochemical phenomena being modeled by reporting which molecular features have the most influence on the classification of IMS pixels.

	Advantages	**Disadvantages**
Logistic regression	Intrinsically interpretable Only one hyper-parameter (i.e. regularization) Computationally efficient Probabilistic predictions	Low capacity model Sensitive to outliers and noise Sensitive to feature scaling
Linear support vector machine	Intrinsically interpretable Robust to outliers and noise	Low capacity model Linear decision boundary Sensitive to feature scaling Computationally costly
Nonlinear support vector machine	High capacity model Robust to outliers and noise	Several hyper-parameters (i.e. kernel & regularization) Not intrinsically interpretable Sensitive to feature scaling Computationally costly
Random forests	High capacity model Nonlinear decision boundary Intrinsically interpretable Robust to outliers and noise Scale invariant Computationally efficient through parallelization	Several hyper-parameters (i.e. number & depth of decision trees)

Table 5-7: Comparison of logistic regression, linear support vector machine, nonlinear support vector machine, and random forest models for the classification of imaging mass spectrometry data.

Having observed that inherently interpretable models, like LR, RF, and linear SVM, achieve high predictive performance on IMS data, we argue that - despite the large size and high-dimensionality of IMS data - there is no need for black-box models for the classification tasks presented in this thesis. Our results suggest that interpretability does not necessarily come at the expense of accuracy. Contrarily to widespread belief [37], having to choose between model interpretability and predictive performance is arguably a false dichotomy [96]. We therefore strongly recommend using intrinsically interpretable supervised machine learning models for the classification of IMS data. Note that although post-hoc model-agnostic interpretability methods are not necessary for explaining the predictions of inherently interpretable models, they provide a way of measuring the impact of each feature on the model's generalization performance. Furthermore, the fact that model-agnostic methods are non-parametric makes the comparison of different models and model types easier.

5-2 Biomarker Discovery

5-2-1 Feature Ranking Variability

We observe that different feature ranking methods partially disagree regarding which features (i.e. m/z bins) make up the top 10%. We measure the rate of agreement between pairs of feature ranking methods using the percentage of overlapping molecules (POM), as defined in Equation 4-1. Dataset n^{o}1 has 2003 features, and we consider the top-ranking 200 features for computing the POM. Dataset n^{o}2 has 321 features, and we consider the top-ranking 32 features for computing the POM. Linear SVM models, LR, and RF models have model-intrinsic interpretability methods that provide us with an estimate of the relative predictive importance of features. We would have to resort to model-agnostic interpretability methods, namely permutation importance (PI) and Shapley importance (SI), to estimate the relative predictive importance of the features of black-box models like non-linear SVMs or deep neural networks. Note that, **unlike model-specific methods that can only be applied to a classifier's training set, post-hoc model-agnostic interpretability methods can also be applied to the testing set and can therefore measure the impact of each feature on the classifier's generalization performance**. However, in order to facilitate the comparison of different interpretability methods, all feature rankings discussed in section 5-2 were obtained from the training sets of our respective classifiers. The classifiers we study in subsection 5-2-1 are those that were trained in section 5-1. Regarding dataset n^{o}1 (i.e. coronal section of a rat brain), our aim is to recognize the diseased left brain hemisphere from the right brain hemisphere. As for dataset n^{o}2 (i.e. sagittal section of a mouse-pup), the five classification tasks we consider involve recognizing the brain, liver, heart, lungs, and adipose tissue from the other organs.

In practice, the variability between feature ranking methods poses a problem for biomarker discovery. Indeed, consistency is the key to reproducible research and clinical applicability [97]. Ranking variability, also referred to as ranking instability or inconsistency, is known to impact the reliability of high-throughput genomic and proteomic studies [33]. In the following subsubsections, we investigate the causes for the observed feature ranking variability.

5-2-1-1 Feature Ranking Variability between Models

We observe a variability in feature rankings obtained by model-specific methods applied to different models. For example, the LR model of subsubsection 5-1-1-1 and the RF model of subsubsection 5-1-1-2 have a POM of 72% according to their respective model-specific interpretability methods: the LR and RF classifiers agree on 144 out of 200 top-ranking features. The reason why their POM score is not 100% is that their respective model-specific interpretability methods have a different way of quantifying feature predictive importance. In the case of LR, the importance of a feature is equal to the magnitude of the feature's coefficient in the LR classifier's decision boundary (refer to subsubsection 3-1-2-1). In the case of RF, the MDI importance of a feature is calculated by summing the weighted impurity decrease for all nodes that are split using that feature, averaged over all decision trees making up the random forest (refer to subsection 3-3-3). The feature ranking method intrinsic to LR does not account for feature inter-dependencies, whereas the MDI measure intrinsic to RF, also termed Gini importance, does. Table 5-8 presents the POM scores obtained by comparing the top-ranking (i.e. top 10%) features according to the model-specific interpretability methods intrinsic to LR and RF classifiers, on all six classification problems.

	Percentage of overlapping molecules
Diseased rat brain	72%
Mouse-pup brain	70%
Mouse-pup liver	79%
Mouse-pup heart	85%
Mouse-pup lungs	78%
Mouse-pup adipose tissue	82%

Table 5-8: Percentage of overlapping molecules obtained by comparing the lists of top-raking features provided by logistic regression and random forest classifiers, trained for six different classification tasks.

Another reason for the feature ranking variability observed between different models is the fact that training a supervised learning model involves a certain degree of stochasticity, especially when handling high-dimensionality IMS data whose features are correlated. For every group of highly correlated features, one or a few features will be randomly selected as representatives of the correlated group during training, whereas the others will be considered redundant by the supervised learning algorithm and will therefore be assigned little importance in the resulting classifier [33]. The problem is that, although we assume that statistically important features are biologically relevant, the inverse is not necessarily true: a feature that is not assigned a high importance within a machine learning model due to redundancy is not necessarily biologically irrelevant. Feature redundancy may therefore yield misleading estimates of the predictive importance of correlated features. It could also cause slight feature ranking variability if the machine learning algorithm used to train the classifier selects different representatives of each group of correlated features from one run to the next.

We address the issue of feature ranking variability due to differences between classifiers in subsection 5-2-3 by taking an ensemble approach for estimating the predictive importance of features.

5-2-1-2 Feature Ranking Variability between Interpretability Methods

For a given classifier, we observe a variability in the ranking provided by its intrinsic interpretability method to the rankings provided by PI and SI. PI and SI are model-agnostic interpretability methods to be used after having trained the classifier, whereas model-specific methods perform feature ranking during training. Post-hoc model-agnostic interpretability methods are non-parametric methods that evaluate feature importance by observing how perturbing the predictor variables impacts the predictive performance of the classifier. We argue that the difference between lists of top-ranking features returned by different interpretability methods is primarily due to how these methods account for correlation between features. We address the issue of feature variability due to differences between interpretability methods in subsection 5-2-2.

5-2-2 Post-whitening Feature Ranking

In order to demonstrate that the differences between feature rankings, for a given classifier, are primarily due to how different interpretability methods account for correlation between features, we apply whitening as a prepossessing step. As discussed in section 2-3, whitening, or sphering, is a linear transformation that decorrelates the features of IMS data by imposing an identity covariance matrix to the data. We choose to perform whitening by ZCA-cor, which is a scale-invariant version of whitening by zero-phase component analysis (ZCA). Since the uncorrelated features obtained by ZCA-cor are maximally correlated to the original features, they maintain their original interpretation. Note that whitening by ZCA-cor also involves standardization, that is zero-centering and unit-variance scaling of the features, prior to decorrelation.

In order to be able to fairly compare the classifiers trained on whitened and non-whitened IMS datasets, we must ensure that the features of both the whitened and non-whitened datasets have been standardized. We use two classifiers (one LR and one RF) that have been trained on IMS data to which we apply noise removal by principal component analysis (PCA), feature zero-centering, and feature unit-variance scaling (i.e. same pre-processing as in section 5-1). Note that SVM classifiers are not considered in subsection 5-2-2 because training several SVMs would have been computationally unfeasible within a practical time frame. We compare these classifiers to two other classifiers (one LR and one RF) that have been trained on IMS data to which we apply noise removal by PCA, feature zero-centering, feature unit-variance scaling, and feature decorrelation by ZCA-cor whitening. In Table 5-9, we compute the POM between different interpretability methods, for each classifier. We observe far more overlap (i.e. higher POM scores) between lists of differentially expressed molecules when the features are decorrelated prior to LR and RF classifier training. For each RF and LR classifier in Table 5-9, we compare the POM obtained by three different interpretability methods: one model-specific method, which is the magnitude of the feature weights for LR (refer to subsection 3-1-2) and the Gini importance for RF (refer to subsection 3-3-3), and two model-agnostic methods, namely PI (refer to subsection 3-4-1) and SI (refer to subsection 3-4-2). Since the only difference in preprocessing is whitening, we conclude that the variability between feature rankings is primarily due to how the different interpretability methods account for correlation between features. Decorrelating features seems to reduce the inconsistency we observe between lists of highly discriminative features, as provided by different interpretability methods. We argue that whitening is a partial solution to the problem of feature ranking inconsistency between different interpretability methods.

	Measures of feature importance	**Pre-processing**	**POM**
Logistic regression	Magnitude of feature weights & Permutation importance	Without withening	66%
		With withening	83%
	Magnitude of feature weights & Shapley importance	Without whitening	33%
		With whitening	50%
	Permutation importance & Shapley importance	Without whitening	50%
		With whitening	66%
Random forest	Gini importance & Permutation importance	Without whitening	50%
		With whitening	83%
	Gini importance & Shapley importance	Without whitening	50%
		With whitening	66%
	Permutation importance & Shapley importance	Without whitening	50%
		With whitening	66%

Table 5-9: Percentage of overlapping molecules obtained by logistic regression and random forest classifiers, according to different model-specific and model-agnostic interpretability methods. We observe that decorrelating the features of imaging mass spectrometry data by whitening prior to training classifiers increases the percentage of overlapping molecules (POM) between different interpretability methods. Each interpretability method has a slightly different way of measuring the predictive importance of features. We compare the top 10% of features and consider one target classes: the liver of a mouse-pup in dataset n^{o}2. Forty iterations were used to compute the permutation and Shapley importance scores of features. Note that we obtain higher percentage of overlapping molecules by considering more iterations per feature.

The classification task we consider for the experiment of Table 5-9 is the recognition of the liver of the mouse-pup in dataset n^{o}2 from the mouse-pup's other organs. In order to limit computational costs, we perform the decorrelation experiment on a version of dataset n^{o}2 that has fewer molecular features. We choose a version of dataset n^{o}2 that has only 49 features, hence a data matrix of 164808 rows and 49 columns. Note that, although the model-intrinsic interpretability methods of LR and RF classifiers are very efficient on high-dimensional data, the model-agnostic methods are computationally costly. Although we initially had a preference for SI over PI because of its rigorous mathematical foundations, SI only returns a reliable estimate of feature predictive importance if it is able to consider the contribution of each feature to many coalitions. We recommend setting the number of feature coalitions to consider for calculating the SI importance of each feature to at least 50% of the total number of features n. Yet, considering many feature coalitions is computationally very costly in practice. For example, computing the SI importance of the $n = 2003$ features of dataset n^{o}1 would have involved testing the classifier $2 \cdot 10^6$ times (i.e. 1000 feature coalitions to consider per feature). By contrast, a reliable PI score can be achieved by averaging the decrease in classifier predictive performance over but a few iterations, as a result of which computing the permutation importance of all n features of dataset n^{o}1 would require testing the classifier approximately $2 \cdot 10^3$ times. We therefore conclude, in terms of practical use, that PI is actually preferable to SI for the analysis of high-dimensional IMS data.

5-2-3 Ensemble Approach to Feature Ranking

We believe that ensemble feature ranking is also a promising way of solving feature ranking variability. Aggregating lists of top-ranking features provided by different interpretability methods improves the robustness and reliability of the resulting feature ranking [94]. Our approach to ensemble feature ranking consists in averaging the importance scores of each feature, as provided by different model-specific interpretability methods, and then ranking the molecules based on the aggregated predictive importance criterion. The reason why we choose to aggregate the rankings produced by model-specific interpretability methods, rather than model-agnostic methods, is that the model-specific methods are far more computationally efficient. Ensemble feature ranking is adapted from a method referred to in scientific literature as "ensemble feature selection with function perturbation", which consists in aggregating different ranking criteria to select a reduced set of important features [97]. As mentioned in subsection 1-2-2, feature selection is feature ranking followed by the omission of features with low predictive ability (i.e. features whose importance scores fall bellow an arbitrarily defined threshold). We choose not to discard features because it is not necessary for improving the explainability of our supervised learning models.

What follows is the list of top-ranking features obtained by computing the mean importance score of each feature and reordering the features as per the aggregated criterion. We focus on the same three classification tasks as in 5-2-2 and use the same LR and RF classifiers that were trained on standardized (non-whitened) IMS datasets n^{o}1 and n^{o}2. The aggregated feature importance criterion therefore combines the magnitude of the features weights in the decision boundary of the LR classifier with the MDI score, also termed Gini importance, provided by the RF classifier. Table 5-10, lists the mass-to-charge ratios of the top-ranking 2% of features, per classification task, according to the ensemble feature ranking approach. Dataset n^{o}1 has 2003 features so we present the most important 40 features, whereas dataset n^{o}2 has 321 features so we present the most important 6 features.

Target class	Mass-to-charge ratios of top-ranking features
Diseased rat brain	4060.2; 4051.6; 4043.1; 4068.8; 1824.0; 4060.7; 4052.1; 4069.2; 1823.0; 4043.6; 1828.7; 2441.0; 4086.1; 1927.9; 2440.0; 4052.6; 4061.2; 1929.9; 1829.3; 1829.0; 3940.8; 1756.0; 4069.8; 5564.9; 5610.9; 6711.5; 1757.0; 6712.5; 4044.1; 7670.4; 1755.0; 3939.8; 5612.9; 1353.1; 1884.9; 1353.0; 4272.2; 1353.1; 3938.8; 6713.5
Mouse-pup brain	729.6; 560.3; 801.5; 800.5; 710.5; 769.5
Mouse-pup liver	720.4; 569.3; 568.3; 820.6; 821.6; 891.6
Mouse-pup heart	856.6; 857.6; 858.6; 859.6; 791.5; 818.62
Mouse-pup lungs	704.5; 732.5; 733.5; 728.5; 729.5; 718.5
Mouse-pup adipose tissue	812.6; 813.6; 731.5; 811.6; 730.5; 810.6

Table 5-10: Table listing the top-ranking 2% of features obtained for each classification task using the ensemble approach to feature ranking. We consider six different target classes: the diseased left brain hemisphere of a rat brain (i.e. dataset n^{o}1) and the brain, liver, heart, lungs and adipose tissue of a mouse-pup (i.e. dataset n^{o}2). The ensemble approach combines the estimates of feature importance obtained by a logistic regression and a random forest classifier.

Chapter 6

Conclusion and Future Work

6-1 Conclusion

In this thesis, we developed a machine learning workflow for discovering biomarkers in imaging mass spectrometry data with a focus on interpretability, reproducibility, and computational efficiency. Biomarker discovery is the empirical identification of molecular markers that enable the recognition of a specific biological state. Identifying a panel of biomarkers can help generate insight into the underlying mechanism of disease and drug action. Our data-driven workflow encompasses unsupervised and supervised machine learning algorithms. Unsupervised machine learning algorithms, namely principal component analysis and non-negative matrix factorization, are used for exploratory analysis and dimensionality reduction. Supervised machine learning algorithms, namely logistic regression, random forests, and support vector machines, are used for classification. All algorithms were implemented from scratch in MATLAB. Having compared logistic regression, random forests, and support vector machines in terms of predictive performance (i.e. accuracy, precision, recall), scale invariance, sensitivity to noise, and computational efficiency, we strongly recommend random forests for the classification of imaging mass spectrometry data. Furthermore, our results suggest that, despite the large size and high-dimensionality of imaging mass spectrometry data, there is no need for black-box models. We argue that having to choose between interpretability and predictive performance for the classification of imaging mass spectrometry data is a false dichotomy.

Traditionally, the focus has been on maximizing the predictive performance of supervised machine learning models, without necessarily examining the models' decision-making processes. Yet, in order to identify which features, and thus which molecular species, drive physiological and pathophysiological processes, it is necessary to go beyond the scope of predictive modeling and learn from our models by investigating the relationship between the features and the model's predictions. We define machine learning model interpretability as the ability to explain a model's predictions by explicitly reporting the relative predictive importance of the features it uses as inputs. In the context of imaging mass spectrometry, interpretability is crucial for understanding how the spatial distribution and relative concentration of certain molecular features relate to the assignment of pixels to different biological, anatomical or pathological classes. We translate the problem of biomarker discovery into a feature ranking problem: the molecular features of supervised machine learning models are ranked in descending order of relative predictive importance in order to narrow the scope of further clinical investigation from thousands of candidates down to a short list of top-ranking, thus highly discriminative, features. We presented one model-specific interpretability method for each inherently interpretable classifier, namely logistic regression, linear support vector machines and random forests. The feature importance measure inherent to random forests, called mean decrease impurity, is the only model-specific method that accounts for feature interdependencies. We discuss the potentially misleading effect that correlation between features may have on their estimated predictive importance scores. We also presented two model-agnostic interpretability methods: permutation importance and Shapley importance, whose implementation as a global interpretability method, based on Shapley values from cooperative game theory, is novel. These model-agnostic methods provide post-hoc explanations by treating the original model as a black-box: the importance of a feature is determined by how much the model's predictive performance is degraded by randomly perturbing that feature. Permutation importance is observed to be computationally less costly than Shapley importance and is therefore preferred for applications to imaging mass spectrometry data.

Regarding the reproducibility of biomarker discovery, we proposed to measure the consistency of two lists of features by computing the percentage of overlapping molecules on the top-ranking 10% of features belonging to each list. Having observed a small variability between the rankings provided by interpretability methods, we proposed a robust ensemble approach that aggregates the importance scores attributed to each feature by different model-specific interpretability methods. Model-specific interpretability methods are chosen over model-agnostic methods because of their superior computational efficiency. Two imaging mass spectrometry datasets were used to demonstrate our methodology: dataset n^{o}1 is a MALDI-FTICR IMS dataset taken from the coronal section of a rat brain, dataset n^{o}2 is a MALDI-TOF IMS dataset taken from the sagittal whole-body section of a mouse-pup. We finally presented a list of candidate biomarkers, according to our ensemble feature ranking method, for the six following binary classification problems: the classification of diseased versus healthy brain tissue in dataset n^{o}1 and the classification of the mouse pup's brain, liver, heart, lungs and dorsal adipose tissue versus its other organs in dataset n^{o}2.

To conclude, this thesis lays the foundations for the application of state-of-the-art explainable artificial intelligence methods to the analysis of imaging mass spectrometry data. We believe that our machine learning workflow for discovering biomarkers in imaging mass spectrometry data will help researchers identify biomarker candidates. The methodology developed in this thesis is furthermore generalizable to other imaging methods, such as spectroscopic imaging.

6-2 Future Work

- *Improve the computational efficiency of post-hoc model-agnostic interpretability methods*: Permutation importance and Shapley importance are post-hoc interpretability methods that can explain the decision-making process of black-box models, such as deep neural networks. However, in order to produce reliable estimates of features' predictive importance, permutation importance and Shapley importance must average the predictive importance estimates of each feature over a large number of observation permutations and feature permutations, respectively. As a result, permutation importance and Shapley importance are impractical for high-dimensional classification problems. We therefore recommend improving the time complexity of permutation importance and Shapley importance.
- *Develop new ways of evaluating the rate of agreement between two lists of top-ranking features*: Measuring the consistency of two lists by simply counting features that appear in both lists may give the misleading impression that two lists are very different, whereas they are actually highly correlated. We would like to investigate alternatives to the percentage of overlapping molecules that take feature correlation into account: for example, by counting both the features shared by two lists, and the features belonging to one of the lists that are highly correlated with at least one feature from the other list.
- *Improve the computational efficiency of the sequential minimal optimization algorithm*: Our choice to train support vector machine classifiers using the sequential minimal optimization algorithm was largely motivated by its relatively small memory requirements. However, the time complexity of sequential minimal optimization is very large for large datasets like imaging mass spectrometry datasets. We therefore suggest improving the stopping condition, and perhaps implementing a parallel version of sequential minimal optimization.

Glossary

AI	artificial intelligence
ALS	alternating least squares
CART	classification and regression trees
DESI	desorption electrospray ionization
FPR	false positive rate
FTICR	Fourier transform ion cyclotron resonance
GDPR	General Data Protection Regulation
GLM	generalized linear model
HALS	hierarchical alternating least squares
IMS	imaging mass spectrometry
KKT	Karush-Kuhn-Tucker
LR	logistic regression
MALDI	matrix assisted laser desorption ionization
MDI	mean decrease impurity
MS	mass spectrometry
NNMF	non-negative matrix factorization
OOB	out-of-bag
OVA	one-versus-all
PC	principal component
PCA	principal component analysis
PDS	positive definite symmetric
PI	permutation importance
POM	percentage of overlapping molecules
QP	quadratic programming
RBF	radial basis function
RF	random forest
SHAP	Shapley additive explanations
SI	Shapley importance
SIMS	secondary ion mass spectrometry
SMO	sequential minimal optimization
SVD	singular value decomposition
SVM	support vector machine
TOF	time-of-flight
TPR	true positive rate
VC	Vapnik–Chervonenkis
XAI	explainable artificial intelligence
ZCA	zero-phase component analysis

Bibliography

[1] J. H. Gross, *Mass Spectrometry: A Textbook.* Springer-Verlag, 2 ed., 2011.

[2] F. P. Y. Barre, R. M. A. Heeren, and N. O. Potocnik, “Mass Spectrometry Imaging in Nanomedicine: Unraveling the Potential of MSI for the Detection of Nanoparticles in Neuroscience,” *Current Pharmaceutical Design*, vol. 23, no. 13, pp. 1974–1984, 2017.

[3] E. A. Jones, A. van Remoortere, R. J. M. van Zeijl, P. C. W. Hogendoorn, J. V. M. G. Bovée, A. M. Deelder, and L. A. McDonnell, “Multiple Statistical Analysis Techniques Corroborate Intratumor Heterogeneity in Imaging Mass Spectrometry Datasets of Myxofibrosarcoma,” *PLOS ONE*, vol. 6, Sept. 2011.

[4] I. Goodfellow, Y. Bengio, and A. Courville, “Part 1, Chapter 5: Machine Learning Basics,” in *Deep Learning*, Online MIT Press book: http://www.deeplearningbook.org/, 2016.

[5] K. Dunn, “Chapter 6: Latent variable modelling,” in *Process Improvement Using Data*, Online book: https://learnche.org/pid/, 2019.

[6] M. Mohri, A. Rostamizadeh, and A. Talwalkar, “Chapter 5: Support vector machines & Chapter 6: Kernel methods,” in *Foundations of Machine Learning*, MIT Press, 2nd ed., 2018.

[7] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, “Chapter 8: Support vector machines,” AMLBook, 2017.

[8] J. Platt, “Fast Training of Support Vector Machines Using Sequential Minimal Optimization,” *Microsoft Research*, Jan. 1998.

[9] V. Honavar, “Sequential Minimal Optimization for SVM,” tech. rep., Iowa State Univeristy, Department of Computer Science.

[10] “Rat Anatomy Guide for Whole Body Imaging (longitudinal) - InvivoPharm.” http://www.invivopharm.com/rat_anatomical_guide.html.

[11] N. Rifai, M. A. Gillette, and S. A. Carr, “Protein biomarker discovery and validation: The long and uncertain path to clinical utility,” *Nature Biotechnology*, vol. 24, pp. 971–983, Aug. 2006.

[12] S. S. Rubakhin and J. V. Sweedler, “A Mass Spectrometry Primer for Mass Spectrometry Imaging | Springer Nature Experiments,” *Mass Spectrometry Imaging*, 2010.

[13] S. A. Schwartz and R. M. Caprioli, “Imaging mass spectrometry: Viewing the future,” *Methods in Molecular Biology*, vol. 656, pp. 3–19, 2010.

[14] K. Chughtai and R. M. A. Heeren, “Mass Spectrometric Imaging for Biomedical Tissue Analysis | Chemical Reviews,” *Chemical Reviews*, Apr. 2010.

[15] C. Murayama, Y. Kimura, and M. Setou, “Imaging mass spectrometry: Principle and application,” *Biophysical Reviews*, vol. 1, Sept. 2009.

[16] D. Parrot, S. Papazian, D. Foil, and D. Tasdemir, “Imaging the Unimaginable: Desorption Electrospray Ionization - Imaging Mass Spectrometry (DESI-IMS) in Natural Product Research,” *Planta Medica*, vol. 84, pp. 584–593, July 2018.

[17] E. A. Jones, S.-O. Deininger, P. C. W. Hogendoorn, A. M. Deelder, and L. A. McDonnell, “Imaging mass spectrometry statistical analysis,” *Journal of Proteomics*, vol. 75, pp. 4962–4989, Aug. 2012.

[18] J. Grassl, N. L. Taylor, and A. Millar, “Matrix-assisted laser desorption/ionisation mass spectrometry imaging and its development for plant protein imaging,” *Plant Methods*, vol. 7, p. 21, July 2011.

[19] J. Spraggins, K. Djambazova, E. Rivera, L. Migas, E. Neumann, A. Fuetterer, J. Suetering, N. Goedecke, A. Ly, R. Van de Plas, and R. Caprioli, “High Performance Molecular Imaging with MALDI Trapped Ion Mobility Time-of-Flight (timsTOF) Mass Spectrometry,” Oct. 2019.

[20] M. W. Libbrecht and W. S. Noble, “Machine learning in genetics and genomics,” *Nature reviews. Genetics*, vol. 16, pp. 321–332, June 2015.

[21] G. Bonaccorso, *Mastering Machine Learning Algorithms.* Packt Publishing, Online book: https://learning.oreilly.com/library/view/mastering-machine-learning/9781788621113/, May 2018.

[22] S. Russell and P. Norvig, “Chapter 18: Learning from examples,” in *Artificial Intelligence, a Modern Approach*, Prentice Hall, 3rd ed.

[23] R. O. Duda, D. G. Stork, and P. E. Hart, “Chapter 9: Algorithm-independent machine learning,” in *Pattern Classification*, John Wiley & Sons, 2nd ed., 2012.

[24] O. Simeone, “Chapter 5: Statistical Learning Theory,” in *A Brief Introduction to Machine Learning for Engineers*, Foundations and Trends in Signal Processing, 2018.

[25] A. C. Müller and S. Guido, “Chapter 5: Model Evaluation and Improvement,” in *Introduction to Machine Learning with Python: A Guide for Data Scientists*, o’reilly media ed., 2016.

[26] M. Du, N. Liu, and X. Hu, “Techniques for Interpretable Machine Learning,” *arXiv:1808.00033 [cs, stat]*, July 2018.

[27] S. Rathi, *Legal Issues and Computational Measures at the Cross-Section of AI, Law and Policy.* PhD thesis, International Institute of Information Technology, Hyderabad, May 2019.

[28] Z. C. Lipton, “The Mythos of Model Interpretability,” *arXiv:1606.03490 [cs.LG]*, Mar. 2017.

[29] I. Guyon and A. Elisseeff, “An Introduction to Variable and Feature Selection,” *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[30] X. Sun, Y. Liu, J. Li, J. Zhu, X. Liu, and H. Chen, “Using cooperative game theory to optimize the feature selection problem,” *Neurocomputing*, vol. 97, pp. 86–93, Nov. 2012.

[31] C. Baumgartner, M. Osl, M. Netzer, and D. Baumgartner, “Bioinformatic-driven search for metabolic biomarkers in disease,” *Journal of Clinical Bioinformatics*, vol. 1, p. 2, Jan. 2011.

[32] E. L. Xu, X. Qian, Q. Yu, H. Zhang, and S. Cui, “Feature selection with interactions in logistic regression models using multivariate synergies for a GWAS application,” *BMC Genomics*, vol. 19, p. 170, Mar. 2018.

[33] L. Tolosi and T. Lengauer, “Classification with correlated features: Unreliability of feature ranking and solutions,” *Bioinformatics*, vol. 27, pp. 1986–1994, July 2011.

[34] C. Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.* Online book: https://christophm.github.io/interpretable-ml-book/, 2019.

[35] F. Doshi-Velez and B. Kim, “Towards A Rigorous Science of Interpretable Machine Learning,” *arXiv:1702.08608 [cs, stat]*, Feb. 2017.

[36] Y. Saeys, I. Inza, and P. Larranaga, “A review of feature selection techniques in bioinformatics,” *Bioinformatics*, vol. 23, pp. 2507–2517, Oct. 2007.

[37] M. T. Ribeiro, S. Singh, and C. Guestrin, “Model-Agnostic Interpretability of Machine Learning,” *arXiv:1606.05386 [cs, stat]*, June 2016.

[38] L. Bravo-Merodio, J. A. Williams, G. V. Gkoutos, and A. Acharjee, “-Omics biomarker identifcation pipeline for translational medicine,” *Journal of Translational Medicine*, vol. 17, no. 1, 2019.

[39] A. J. van Gool, F. Bietrix, E. Caldenhoven, K. Zatloukal, A. Scherer, J.-E. Litton, G. Meijer, N. Blomberg, A. Smith, B. Mons, J. Heringa, W.-J. Koot, M. J. Smit, M. Hajduch, T. Rijnders, and A. Ussi, “Bridging the translational innovation gap through good biomarker practice,” *Nature Reviews Drug Discovery*, vol. 16, no. 9, 2017.

[40] K. Strimbu and J. A. Tavel, “What are Biomarkers?,” *Current Opinion in HIV and AIDS*, vol. 5, no. 6, 2010.

[41] N. B. Erichson, A. Mendible, S. Wihlborn, and J. N. Kutz, “Randomized Nonnegative Matrix Factorization,” *Pattern Recognition Letters*, vol. 104, pp. 1–7, Mar. 2018.

[42] I. Jolliffe, *Principal Component Analysis.* Statistics, Springer, 2nd ed., 2002.

[43] E. Robotti, M. Manfredi, and E. Marengo, "Biomarkers Discovery through Multivariate Statistical Methods: A Review of Recently Developed Methods and Applications in Proteomics," *Journal of Proteomics & Bioinformatics*, vol. 0, no. 0, pp. 1–20, 2015.

[44] A. Ng, "Machine Learning by Stanford University - Coursera." https://www.coursera.org/learn/machine-learning.

[45] P. W. Siy, R. A. Moffitt, R. M. Parry, Y. Chen, Y. Liu, M. C. Sullards, A. H. Merrill, and M. D. Wang, "Matrix Factorization Techniques for Analysis of Imaging Mass Spectrometry Data," *2008 8th IEEE International Conference on BioInformatics and BioEngineering*, Oct. 2008.

[46] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, Oct. 1999.

[47] L. Badea and D. Tilivea, "Stable Biclustering of Gene Expression Data with Nonnegative Matrix Factorizations," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, (San Francisco, CA, USA), pp. 2651–2656, Morgan Kaufmann Publishers Inc., 2007.

[48] Q. Gu and K. Veselkov, "Bi-clustering of metabolic data using matrix factorization tools," *Methods*, vol. 151, pp. 12–20, Dec. 2018.

[49] J.-P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov, "Metagenes and molecular pattern discovery using matrix factorization," *Proceedings of the National Academy of Sciences*, vol. 101, pp. 4164–4169, Mar. 2004.

[50] P. Carmona-Saez, R. D. Pascual-Marqui, F. Tirado, J. M. Carazo, and A. Pascual-Montano, "Biclustering of gene expression data by non-smooth non-negative matrix factorization," *BMC Bioinformatics*, vol. 7, p. 78, Feb. 2006.

[51] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," *Computational Statistics & Data Analysis*, vol. 52, pp. 155–173, 2007.

[52] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," in *Computational Statistics and Data Analysis*, pp. 155–173, 2006.

[53] N. Gillis and F. Glineur, "Accelerated Multiplicative Updates and Hierarchical ALS Algorithms for Nonnegative Matrix Factorization," *Neural Computation*, vol. 24, pp. 1085–1105, Apr. 2012.

[54] A. Cichocki and A. H. Phan, "Fast Local Algorithms for Large Scale Nonnegative Matrix and Tensor Factorizations," *IEICE Transactions*, vol. 92-A, pp. 708–721, 2009.

[55] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, Nov. 1987.

[56] S. Monti, P. Tamayo, J. Mesirov, and T. Golub, “Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data | SpringerLink,” *Machine Learning*, vol. 52, pp. 91–118, July 2003.

[57] H. Kim and H. Park, “Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis,” *Bioinformatics*, vol. 23, pp. 1495–1502, June 2007.

[58] Y. Șenbabaoğlu, G. Michailidis, and J. Z. Li, “Critical limitations of consensus clustering in class discovery,” *Scientific Reports*, vol. 4, p. 6207, Aug. 2014.

[59] A. Kessy, A. Lewin, and K. Strimmer, “Optimal whitening and decorrelation,” *The American Statistician*, vol. 72, pp. 309–314, Oct. 2018.

[60] G. Li and J. Zhang, “Sphering and Its Properties,” *Sankhyā: The Indian Journal of Statistics*, vol. 60, p. 16, 1998.

[61] J. Gareth, D. Witten, T. Hastie, and R. Tibshirani, “Chapter 4: Classification,” in *An Introduction to Statistical Learning with Applications in R*, Springer, 2013.

[62] G. Hackeling, *Mastering Machine Learning with Scikit-Learn.* Packt Publishing, 2nd ed., 2017.

[63] J. W. Johnson, “A Heuristic Method for Estimating the Relative Weight of Predictor Variables in Multiple Regression,” *Multivariate Behavioral Research*, vol. 35, pp. 1–19, Jan. 2000.

[64] S. Tonidandel and J. M. LeBreton, “Determining the Relative Importance of Predictors in Logistic Regression: An Extension of Relative Weight Analysis,” *Organizational Research Methods*, vol. 13, pp. 767–781, Oct. 2010.

[65] J. Gareth, D. Witten, T. Hastie, and R. Tibshirani, “Chapter 9: Support vector machines,” in *An Introduction to Statistical Learning with Applications in R*, Springer, 2013.

[66] S. Theodoridis and K. Koutroumbas, “Chapter 3.7: Support Vector Machines & Chapter 4.18: Support vector machines, the nonlinear case,” in *Pattern Recognition*, Academic Press, Elsevier, 4th ed., 2009.

[67] X. Zhang, X. Lu, Q. Shi, X.-q. Xu, H.-c. E. Leung, L. N. Harris, J. D. Iglehart, A. Miron, J. S. Liu, and W. H. Wong, “Recursive SVM feature selection and sample classification for mass-spectrometry and microarray data,” *BMC Bioinformatics*, vol. 7, p. 197, Apr. 2006.

[68] H.-T. Lin, “A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods,” 2005.

[69] G. Louppe, *Understanding Random Forests: From Theory to Practice.* PhD thesis, University of Liège, Department of Electrical Engineering & Computer Science, July 2014.

[70] R. O. Duda, D. G. Stork, and P. E. Hart, “Chapter 8: Tree-based methods,” in *Pattern Classification*, John Wiley & Sons, 2nd ed., 2012.

[71] A. Criminisi, E. Konukoglu, and J. Shotton, "Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning," tech. rep., Microsoft Technical Report, Oct. 2011.

[72] M. Kuhn and K. Johnson, "Chapter 8: Regression Trees and Rule-Based Models & Chapter 14: Classification Trees and Rule-Based Models," in *Applied Predictive Modeling*, Springer, 1st ed.

[73] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 431–439, Curran Associates, Inc., 2013.

[74] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001.

[75] X. Chen and H. Ishwaran, "Random forests for genomic data analysis," *Genomics*, vol. 99, pp. 323–329, June 2012.

[76] J. Gareth, D. Witten, T. Hastie, and R. Tibshirani, "Chapter 8: Tree-based methods," in *An Introduction to Statistical Learning with Applications in R*, Springer, 2013.

[77] C. Strobl, *Statistical Issues in Machine Learning – Towards Reliable Split Selection and Variable Importance Measures.* PhD thesis, Ludwig Maximilian University of Munich, Faculty of Mathematics, Informatics and Statistics, May 2008.

[78] T. Hastie, R. Tibshirani, and J. Friedman, "Chapter 15: Random Forests," in *Elements of Statistical Learning: Data Mining, Inference, and Prediction.*, Springer, 2nd ed., 2017.

[79] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution," *BMC Bioinformatics*, vol. 8, p. 25, Jan. 2007.

[80] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: A corrected feature importance measure," *Bioinformatics*, vol. 26, pp. 1340–1347, May 2010.

[81] A. Fisher, C. Rudin, and F. Dominici, "All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously," *arXiv:1801.01489 [stat]*, Jan. 2018.

[82] C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis, "Conditional variable importance for random forests," *BMC Bioinformatics*, vol. 9, p. 307, July 2008.

[83] G. Casalicchio, C. Molnar, and B. Bischl, "Visualizing the Feature Importance for Black Box Models," *arXiv:1804.06620 [cs, stat]*, Apr. 2018.

[84] S. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," *arXiv:1705.07874 [cs, stat]*, May 2017.

[85] E. Strumbelj and I. Kononenko, "An Efficient Explanation of Individual Classifications Using Game Theory," *J. Mach. Learn. Res.*, vol. 11, pp. 1–18, Mar. 2010.

[86] S. Cohen, G. Dror, and E. Ruppin, "Feature Selection via Coalitional Game Theory," *Neural Computation*, vol. 19, pp. 1939–1961, July 2007.

[87] E. Štrumbelj and I. Kononenko, “Explaining prediction models and individual predictions with feature contributions,” *Knowledge and Information Systems*, vol. 41, pp. 647–665, Dec. 2014.

[88] K. Aas, M. Jullum, and A. Løland, “Explaining individual predictions when features are dependent: More accurate approximations to Shapley values,” *arXiv:1903.10464 [cs, stat]*, Mar. 2019.

[89] N. Verbeeck, J. M. Spraggins, M. J. M. Murphy, H.-D. Wang, A. Y. Deutch, R. M. Caprioli, and R. Van de Plas, “Connecting imaging mass spectrometry and magnetic resonance imaging-based anatomical atlases for automated anatomical interpretation and differential analysis,” *Biochimica Et Biophysica Acta. Proteins and Proteomics*, vol. 1865, pp. 967–977, July 2017.

[90] “Parkinson disease | Definition, Causes, Symptoms, & Treatment | Britannica.com.” https://www.britannica.com/science/Parkinson-disease.

[91] “Basal ganglia | anatomy | Britannica.com.” https://www.britannica.com/science/basal-ganglion.

[92] T. Hastie, R. Tibshirani, and J. Friedman, “Chapter 4: Linear Methods for Classification,” in *Elements of Statistical Learning: Data Mining, Inference, and Prediction.*, Springer-Verlag, 2nd ed., 2017.

[93] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, “A Practical Guide to Support Vector Classification,” tech. rep., Department of Computer Science, National Taiwan University, 2016.

[94] A.-L. Boulesteix and M. Slawski, “Stability and aggregation of ranked gene lists,” *Briefings in Bioinformatics*, vol. 10, pp. 556–568, Aug. 2009.

[95] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, “Improvements to Platt’s SMO Algorithm for SVM Classifier Design,” *Neural Computation*, vol. 13, pp. 637–649, Mar. 2001.

[96] “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” *Nature Machine Intelligence*, vol. 1, pp. 206–215, May 2019.

[97] Z. He and W. Yu, “Stable feature selection for biomarker discovery,” *Computational Biology and Chemistry*, vol. 34, pp. 215–225, Aug. 2010.