Feature Selection from
Heterogeneous Biomedical Data

Providing interpretable models for high dimensional data

Jérôme Paul

Thesis presented for the Ph.D. degree in Engineering Sciences

Thesis jury:
Prof. Pierre Dupont (Université catholique de Louvain), Advisor
Prof. Daniel Hernández-Lobato (Universidad Autónoma de Madrid)
Prof. John Lee (Université catholique de Louvain)
Prof. Charles Pecheur (Université catholique de Louvain), Chairman
Prof. Yvan Saey (Universiteit Gent)
Prof. Michel Verleysen (Université catholique de Louvain)

June 29, 2015
Abstract

Modern personalised medicine uses high dimensional genomic data to perform customised diagnostic/prognostic. In addition, physicians record several medical parameters to evaluate some clinical status. In this thesis we are interested in jointly using those different but complementary kinds of variables to perform classification tasks. Our main goal is to provide interpretability to predictive models by reducing the number of used variables to keep only the most relevant ones. Selecting a few variables that allow us to predict some clinical outcome greatly helps medical doctors to understand the studied biological process better.

Mixing gene expression data and clinical variables is challenging because of their different nature. Indeed genomic measurements are expressed on a continuous scale while clinical variables can be continuous or categorical. While the biomedical domain is the original incentive to this work, we tackle the more general problem of feature selection in the presence of heterogeneous variables. Few variable selection methods jointly handle both kinds of features directly. That is why we focus on tree ensemble methods and kernel approaches.

Tree ensemble methods, like random forests, successfully perform classification from data with heterogeneous variables. In addition, they propose a feature importance index that can rank variables according to their importance in the predictive model. Yet, that index suffers from two main drawbacks. Firstly, the provided feature rankings are highly sensitive to small variations of the datasets. Secondly, while the variables are accurately ranked, it is very difficult to decide which features actually play a role in the decision process. This work puts forward solutions to those two problems. We show in an analysis of tree ensemble methods stabilities that feature rankings get considerably stabler by growing more trees than needed to obtain good predictive performances. We also introduce a statistically interpretable feature selection index. It assesses
whether the variables are important in predicting the class of unseen samples. The output $p$-values offer a very natural threshold to decide which features are significant.

Apart from tree ensemble approaches, there are few feature selection methods that handle continuous and categorical variables in an embedded way. It is however possible to build classifiers that profit from both kinds of data by using kernels. In this thesis, we adapt those techniques to perform heterogeneous feature selection. We propose two kernel-based algorithms that rely on a recursive feature elimination procedure. The importance of the variables is extracted either from a non-linear SVM or multiple kernel learning. Those approaches are shown to provide state-of-the-art results in terms of predictive performances and feature selection stability.
Another quest will start from here. You’ll see these words appear on a black screen if you manage to finish ‘The Legend of Zelda’. As I reach the end of my 5-year quest, I would like to thank the people that helped me over this adventure.

Tout d’abord, merci, Pierre, de m’avoir accueilli dans ton équipe de recherche. J’ai beaucoup apprécé ces 5 années sous ta supervision et ton conseil. Nos nombreuses discussions et tes précieux avis ont forgé cette thèse. Je pense et j’espère que tes qualités ont un peu déteint sur moi, en particulier la rigueur scientifique et le souci du détail (le goût pour les bonnes bières, je l’avais déjà).

I also would like to thank all the members of my thesis committee and jury. Your questions and remarks helped me a lot improve this work. Special thanks go to Michel for your advices and guidance in the first years of my thesis and to John for the thorough feedback on the preliminary version of this manuscript.

Je voudrais également remercier mes collègues d’INGI et du Machine Learning Group. Sam, faire notre mémoire ensemble était déjà très chouette, mais travailler dans le même bureau, pouvoir discuter de nos sujets respectifs et de plein d’autres choses pendant ces 5 dernières années, c’était vraiment génial. Merci aussi, Adrien, de nous avoir rejoint dans ces discussions. Il y avait quand même une sacrée ambiance dans ce bureau :-). My thanks also go to my other colleagues of the MLG: Alexandra, Benoit, Dimitri, Emilie, Guillaume, Roberto. Our seminars and discussions were much enjoyable, as well as our trips to Bruges for the ESANN conference. I also thank all the people in INGI. It’s been a very nice working environment.

Je remercie également mes potes geeks de LLN: Antoine, Fwé, GBB, JB, JC, Ka, Minou, Nico, Sam, Simon, Xa. On en a fait du chemin depuis ces longues journées à rédiger nos mémoires ensemble en salle
Sun ! Toutes ces bières partagées, ces soirées, ces discussions et ces mails échangés ont été pour moi un solide appui pendant cette thèse. Merci à mes anciens et actuels colocs — JB et S, Xa, Læti, Jess et Ka — pour cette très chouette ambiance à l’appart. On en a bien besoin après un gros rush pour soumettre un papier à temps.

Merci aussi à mes amis musiciens. Les répétitions et concerts avec les Quantess Combo’s, Jack Gondry and his New Music, les Blue Caps et le CMN sont des moments précieux qui me permettent de m’évader complètement. Les troisièmes mi-temps en votre compagnie sont tout aussi appréciables !

Enfin, je voudrais remercier ma famille qui a toujours cru en moi et m’a supporté tout au long du chemin. Merci à mes parents, Luc et Anne-Françoise. Depuis que je suis tout petit, vous avez éveillé ma curiosité et m’avez soutenu dans mes activités scolaires, musicales et dans bien d’autres choses. Une bonne partie de cette thèse vous revient donc aussi ! Merci à mon frère, Guillaume, et ma sœur, Marie-Aline, de m’avoir soutenu et demandé de temps en temps, mais pas trop souvent, comment avançait ma recherche. Je pense également à ma grand-mère, Agnès, qui me posait régulièrement la question et qui n’est plus là pour voir le résultat.

For me, it is now time to press the start button again. Thank you all for accompanying me.
1 Introduction
  1.1 Machine learning .................................................. 1
  1.2 Heterogeneous biomedical data .................................. 2
  1.3 Feature selection: what and why ? .............................. 4
  1.4 What is a good feature selection method ? .................... 5
  1.5 Thesis focus .......................................................... 6
    1.5.1 Tree ensembles: better feature selection ................. 7
    1.5.2 Kernel methods: new feature selection schemes ......... 7
  1.6 Summary of the contributions .................................... 8
  1.7 Publication list ...................................................... 9
  1.8 Roadmap ............................................................... 9

I Background and materials ........................................... 11

2 Classification ........................................................ 13
  2.1 Definition and notations .......................................... 14
  2.2 Tree ensemble methods ............................................ 15
    2.2.1 Decision trees ................................................. 15
    2.2.2 Ensemble of trees ............................................ 24
    2.2.3 Limitations of approaches based on decision trees .... 28
  2.3 Support vector machines .......................................... 29
    2.3.1 Linear SVM .................................................... 29
    2.3.2 Non-linear classification with the kernel trick ....... 33
    2.3.3 Soft-margin SVM .............................................. 36
  2.4 Multiple kernel learning .......................................... 37
  2.5 Instance-based and prototype-based classifiers ............ 39
  2.6 Naive Bayes .......................................................... 41
  2.7 Other classifiers .................................................... 42
3 Feature selection 45
  3.1 Dimensionality reduction 45
  3.2 Feature selection paradigms 46
    3.2.1 Filters 47
    3.2.2 Wrappers 47
    3.2.3 Embedded methods 48
  3.3 Heterogeneous feature selection 50
    3.3.1 Naive approach 50
    3.3.2 Wrapping heterogeneous classifiers 51
    3.3.3 Variable importance from Random Forest 52
    3.3.4 Hybrid feature selection 53
    3.3.5 Recursive feature elimination 56
  3.4 Extracting statistically significant features from rankings 57
    3.4.1 mr-Test 57
    3.4.2 1Probe 58

4 Experimental setting 59
  4.1 Predictive performances 59
  4.2 Feature selection stability 60
  4.3 Experimental protocol 61

II More interpretable heterogeneous feature selection methods 65

5 Stable variable rankings from Random Forest 69
  5.1 Predictive performances 70
  5.2 Stability of class prediction 70
  5.3 Stability of feature selection 72
  5.4 Conclusion 73

6 $J_{\chi^2}$: A statistically interpretable importance index from Random Forest 75
  6.1 Definition of $J_{\chi^2}$ 76
  6.2 Concordance with $J_a$ 78
  6.3 $J_{\chi^2}$ highlights important variables 80
  6.4 Good prediction from significant features 82
  6.5 $J_{\chi^2}$ outperforms alternatives 82
  6.6 Discussion 83
  6.7 Conclusion 84
Contents

7 Two kernel approaches for heterogeneous feature selection 85
  7.1 Introducing $RFE^{MKL}$ and $RFE^{SVM}$ ................................. 85
  7.2 Performance assessment .................................................. 87
  7.3 Conclusion ............................................................... 88

8 Conclusion 89

III Main published papers 95

A The stability of feature selection and class prediction from ensemble tree classifiers 97
  A.1 Motivation ............................................................... 98
  A.2 Ensemble of tree classifiers ........................................... 98
  A.3 Experimental design and assessment ................................. 99
  A.4 Results ................................................................. 101
  A.5 Conclusion and perspectives ......................................... 102

B Inferring statistically significant features from Random Forests 107
  B.1 Introduction ........................................................... 108
  B.2 Material and methods ................................................ 108
    B.2.1 Context and notations ........................................... 109
    B.2.2 A statistical feature importance index from RF ............ 110
  B.3 Experiments ........................................................... 113
    B.3.1 Performance metrics ............................................. 113
    B.3.2 Experimental protocol .......................................... 114
    B.3.3 Datasets .......................................................... 115
  B.4 Results and discussion ............................................... 116
    B.4.1 Selecting statistically relevant features with $J_{\chi^2}$ ...... 117
    B.4.2 Concordance with $J_{\alpha}$ .................................... 120
    B.4.3 Prediction from significant features ........................... 120
    B.4.4 Comparison of $J_{\chi^2}$ to 1Probe and mr-Test ............. 123
  B.5 Conclusion and perspectives ......................................... 130

C Kernel methods for heterogeneous feature selection 135
  C.1 Introduction ........................................................... 135
  C.2 Material and methods ................................................ 137
    C.2.1 Recursive feature elimination ................................. 137
    C.2.2 Clinical kernel .................................................. 138
    C.2.3 Feature importance from non-linear Support Vector Machines .................................................. 138
Table of notations

<table>
<thead>
<tr>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>number of samples</td>
</tr>
<tr>
<td>$p$</td>
<td>number of features</td>
</tr>
<tr>
<td>$y$</td>
<td>vector of $n$ labels</td>
</tr>
<tr>
<td>$x_i$</td>
<td>$i$-th sample</td>
</tr>
<tr>
<td>$x_j$</td>
<td>$j$-th variable</td>
</tr>
<tr>
<td>$x_{ij}$</td>
<td>value of $x_i$ for feature $j$</td>
</tr>
<tr>
<td>$S$</td>
<td>set of pairs $(x_i, y_i)$ of labelled instances</td>
</tr>
<tr>
<td>$S_{tr}$</td>
<td>training set</td>
</tr>
<tr>
<td>$S_{te}$</td>
<td>test set</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$BCR$</td>
<td>balanced classification rate</td>
</tr>
<tr>
<td>$TP$</td>
<td>number of true positive predictions</td>
</tr>
<tr>
<td>$FP$</td>
<td>number of false positive predictions</td>
</tr>
<tr>
<td>$TN$</td>
<td>number of true negative predictions</td>
</tr>
<tr>
<td>$FN$</td>
<td>number of false negative predictions</td>
</tr>
<tr>
<td>$KI$</td>
<td>Kuncheva’s stability index</td>
</tr>
<tr>
<td>$FDR$</td>
<td>false discovery rate</td>
</tr>
<tr>
<td>$K$</td>
<td>number of feature sets in $KI$</td>
</tr>
<tr>
<td>$s$</td>
<td>feature set size</td>
</tr>
<tr>
<td>$F_1, \ldots, F_K$</td>
<td>feature sets</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(Ensemble of) Trees</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>set of variables</td>
</tr>
<tr>
<td>$T$</td>
<td>number of trees</td>
</tr>
<tr>
<td>$m$</td>
<td>number of candidate variable for splitting</td>
</tr>
<tr>
<td>$B_k$</td>
<td>bag of the $k$-th tree</td>
</tr>
<tr>
<td>$\overline{B}_k$</td>
<td>out-of-bag (OOB) of the $k$-th tree</td>
</tr>
</tbody>
</table>
\begin{itemize}
  \item $\mathcal{B}$ set of bags
  \item $\overline{\mathcal{B}}$ set of OOB
  \item $\tilde{x}_j$ a permutation of $j$-th variable
  \item $J_a(x_j)$ Breiman’s RF importance of variable $x_j$
  \item $J_{\chi^2}(x_j)$ statistically interpretable importance of feature $x_j$
  \item $h_k(i)$ predicted class label of $k$-th tree for sample $x_i$
  \item $h_k^{\tilde{x}_j}(i)$ idem with $x_j$ permuted
  \item $I(condition)$ indicator function
  \item $p_{\chi^2}(x_j)$ $p$-value of a $\chi^2$ test that $x_j$ is important
  \item $p_{\chi^2}^{dr}(x_j)$ idem, corrected for multiple testing

\textbf{Kernel methods}
\begin{itemize}
  \item $k(x_i, x_j)$ kernel value between $x_i$ and $x_j$
  \item $w$ SVM weight vector
  \item $\alpha_i, \alpha_j$ SVM dual variables
  \item $J_{SVM}(x_j)$ SVM-based importance of $x_j$
  \item $\mu_m$ MKL weight of $m$-th kernel
  \item $g(x)$ discriminant function
  \item $f(x)$ decision rule
  \item $\phi(x)$ explicit feature map
\end{itemize}

\textbf{Miscellaneous}
\begin{itemize}
  \item $R$ feature ranking
  \item $F$ feature set
  \item $d(x_i, x_j)$ distance between $x_i$ and $x_j$
  \item $L$ a learning algorithm
\end{itemize}
Chapter 1

Introduction

The topic of this thesis relates to machine learning methods applied to high dimensional heterogeneous biomedical data. The ultimate goal is to provide tools to medical doctors in order to help them understand biomedical processes better. Yet, this is not a thesis in biology or medicine. The focus of this work is clearly on the development of machine learning techniques.

Keeping that in mind, this chapter sets up the general context of the thesis. Section 1.1 informally explains what machine learning and classification are. Section 1.2 describes heterogeneous biomedical data, their various data sources and the intrinsic differences among them. It also briefly depicts other kinds of heterogeneous data, not related to the biomedical domain. Then, Section 1.3 explains what feature selection is about and how it leads to a better understanding of data. The criteria that make good feature selection methods are pictured in Section 1.4. After that, Section 1.5 explains the focus of this thesis and the tracks we explored. Section 1.6 gives a summary of the contributions of this work and Section 1.7 lists all the related papers. Finally, Section 1.8 is a roadmap of this work.

1.1 Machine learning

Scientia potentia est (Knowledge is power). This old Latin saying has never been so true. In recent years, the technological developments have greatly eased the acquisition and storage of big amounts of data. The per head capacity to store data has roughly doubled every 3.5 years since 1986 [HL11]. Many businesses such as finance, E-commerce, advertisement, social network service, meteorology, and personalised medicine
highly rely on these huge data to take important decisions. However, data itself is not knowledge; knowledge hides in the data. Since the amount of data forbids by-hand analyses, sophisticated computer-based methods are needed to extract and exploit that knowledge.

Machine learning (ML) techniques learn from data in order to generalise on new data. Broadly speaking, it encompass automated techniques that improve with experience i.e. as more data points become available for learning. Machine learning lies in the intersection of computer science, applied mathematics and statistics. It gets more and more attractive and important with the increasing volume of available data.

A specific area of ML consists in predicting some output variable, or response. The typical example, well known to the public, is spam filtering. For each incoming email, the anti-spam system decides if it is legitimate or not, based on the predictive model learned on previously classified mails. In the beginning, the system makes mistakes. Some mails are incorrectly discarded and some junk mails appear in the inbox. One has to manually correct the predicted outcome of the algorithm. Doing so, the anti-spam system becomes more accurate due to the increasing number of data it can learn from.

In this work, we are mainly interested in mining knowledge from biomedical data for personalised medicine. In that area, many tasks appear in the form of classification problems where the response variable is categorical. The goal is to answer diagnostic questions such as ‘Does that patient suffer from cancer ? (yes/no)’, ‘What kind of allergy is the patient suffering from ? (atopic/non-atopic/non-allergic)’, or prognostic questions such as ‘Will the patient positively react to that specific treatment ? (yes/no)’. A typical dataset would consist of biomedical measurements from several patients for each condition. In order to be able to generalise, i.e. to predict the clinical status or label of new patients, a learning algorithm has to find patterns that can differentiate between the various groups of patients in the data. The predictive model is then used to classify new unlabelled patients according to the various parameters learned from the dataset.

1.2 Heterogeneous biomedical data

In order to build a predictive model, the learning algorithm takes some variables (also referred to as features, dimensions, or predictors in this document) as input for each data point. A patient is therefore represented as a set of biomedical measurements. Those features can vary in their very nature and come from many different sources. In this work,
1.2. Heterogeneous biomedical data

we focus on two main classes of variables that are used in two complementary and yet different medical models.

On the one hand, personalised medicine takes advantage of high-throughput screening technologies and machine learning. It aims at customising health care for each individual patient. To do so, modern medical equipments measure an amazingly important number of variables to precisely characterise the medical state of a patient. For instance, the microarray technology [SSDB95] performs a genomic screening from a small biopsy. It estimates the level of activity of tens of thousands of genes in a single experiment. Flow cytometry analysis [CHR08] is another example. Instead of measuring gene activities, it evaluates the concentration of hundreds of proteins present in a sample. Those very high dimensional data are very difficult to analyse by hand. They are fed into machine learning algorithms in order to build medical decision systems.

On the other hand, physicians often register more traditional factors to perform their own prognostic or diagnostic. They usually consider medical and environmental factors such as the body temperature, blood pressure, age, gender, family history, smoking habits, etc. Those are typically encoded by the physician. In many cases, the clinical variables provide enough information to the general practitioner to make a diagnosis.

It is important to note a few differences between those two kinds of data. Firstly, they vary in their very nature. High-throughput technologies usually output quantitative variables, while clinical features can be of different types. Some are categorical e.g. sex (m/f), smoker (no/often/sometimes), pet (no/cat/dog) and family history (y/n). The values of categorical variables are unordered. They partition the population into different groups. Other are numerical e.g. age, blood pressure, and tumour size. They intrinsically encode a notion of order. Secondly, all clinical variables may not be available for all patients. Indeed, the feature acquisition process may require several visits to the doctor and some patients may miss some of them. In addition, some variables may be impossible to record at some point or not estimated useful by the clinician at that time. Finally, while the number of features is huge in the personalised medicine paradigm, it is usually limited to a tens for clinical factors.

In this work, we are mainly interested in combining these two complementary types of variables. The goal is to leverage information contained in both kinds of data in order to study diseases and help practitioners understand underlying medical processes.
Other heterogeneous data

The biomedical context described above is the initial motivation of this work. While handling different kinds of features is difficult \textit{per se}, the high dimensionality and quite reduced number of patients in medical studies make those prediction problems even more challenging. Yet, the contributions of this work are more general. They globally apply to datasets with continuous and categorical variables. Here are some concrete examples of other predictive tasks with heterogeneous data.

Spam detection techniques (\textit{e.g.} \cite{Mas02}) use continuous features to measure \textit{word frequencies} along with some categorical variables such as the \textit{language} or the \textit{character set} of the text. In the task of removing internet advertisements (\textit{e.g.} \cite{Kus99}), some machine learning based techniques use the \textit{presence or absence of terms} in the text or URL as categorical variables. Continuous features such as the \textit{height} and \textit{width} are also taken into account in the decision process. Similarly to biomedical data, those datasets are also high dimensional. However, they benefit from a greater availability of the data samples.

Other prediction tasks may involve datasets with much fewer dimensions. For instance, in the analysis of social behaviours (\textit{e.g.} Adult dataset in the UCI repository \cite{FA10}), continuous variables such as the \textit{income}, \textit{age} and \textit{working hours per week} are considered with categorical features like the \textit{marital status}, \textit{sex} and \textit{education}. A last example concerns the monitoring of mechanical devices such as car engines (see Auto MPG and Automobile in \cite{FA10}). Some categorical variables like the \textit{origin}, \textit{model}, \textit{fuel type} and \textit{engine location} are analysed together with continuous predictors such as the \textit{mileage} and \textit{horsepower}.

1.3 Feature selection: what and why ?

Building a predictive model that is able to make good predictions can help medical doctors to make important decisions. However, with such high-dimensional datasets, it is often quite difficult to get an insight on how and why the model outputs a particular class for a given sample. One way to help physicians to understand a disease better is to provide them with a small subset of biomedical variables, or signature, that are useful to make the prognostic or diagnostic prediction. Indeed, a large part of the thousands of biomedical variables may not be relevant for a particular prediction task. Small feature sets that allow to built good predictive models are way easier to analyse. Feature selection enables doctors to investigate the medical process of interest according to their own knowledge of the involved variables.
1.4 What is a good feature selection method ?

In addition to providing interpretability of the predictive models, feature selection also makes them more robust. During the learning process, most algorithms try to optimise the classification performance on the training data. Yet, it is a particularly easy task when the number \( p \) of dimensions is much higher than the number \( n \) of available data points. In fact, even with a very simple linear separator, it is always possible to achieve perfect discrimination between two classes as soon as \( p \geq n - 1 \) if the data points are not collinear [Vap95]. Biomedical datasets have \( p \gg n \). Therefore, optimising the training classification performance can trivially and rapidly reach 100% accuracy. However, a naively learned classification model may not generalise well on a set of unseen data because its decision boundary does not match the real data distribution. This phenomenon is known as overfitting. More advanced classification methods are designed to limit overfitting, to some extent. Decreasing the number of variables further increases the model robustness. It allows learning algorithms not to get lost in such big spaces and even reduces their computational complexity.

Yet, the task of feature selection is not straightforward. A naive approach such as an exhaustive search over all feature subsets is intractable, especially when dealing with high-dimensional data. For instance, let us suppose that we would like to find the ten most important features for a particular prediction task. Given a dataset of 2,000 variables, the number of possible subsets of 10 features is \( \binom{2000}{10} = 2.76 \times 10^{26} \). Supposing that it takes 0.1 seconds to build a model on one of those 10 features sets, it would take \( 7.5 \times 10^{21} \) hours to build all of them. For comparison purposes, that number has the same order of magnitude as the estimated number of grains of sand on planet Earth. More sophisticated and efficient feature selection techniques are thus needed.

1.4 What is a good feature selection method ?

Feature selection restricts the variables used in a model to a small subset. But what makes a good feature selection method, in general? In order to answer that question, two complementary criteria need to be taken into account: predictive performances and stability.

Of course, the selected feature sets must contain enough relevant information to build good predictive models. The goal is to keep good classification performances with much fewer variables than in the original dataset. Identifying useful features is not an easy task. Simple feature selection methods rely on univariate criteria. They only look for class
information in individual variables. However, most datasets are multi-
variate \( i.e. \) the class information is distributed in several features that
need to be combined to perform good classification. One challenge of
more sophisticated feature selection methods is to identify those groups
of variables that are useful together. It is particularly important with
high dimensional biomedical data where most features are expected to
be useless for a given prediction task.

The stability criterion is related to the interpretability of the selected
features. With biomedical datasets, it is often the case that several
different variable subsets produce equally good predictive models. The
selection algorithm has then little reason to prefer one set over the others.
In that situation, feature selection may be highly sensitive to the data.
Small changes in the dataset could lead to very different signatures \( i.e. \)
sets of selected features). Yet, in clinical studies, such dataset changes
appear very frequently. When some patient enters or leaves the study, we
do not want the selected variables to change drastically. It would make it
more difficult to understand the medical process and would weaken the
confidence of medical doctors in machine learning methods. Therefore,
stability is mandatory to provide interpretability. It is a challenge as
important as good predictive performances.

1.5 Thesis focus

In this work, a special attention is paid to the heterogeneous nature
of the data. Mixing continuous and categorical variables is a challenge
as such, both for classification and feature selection. Indeed, the joint
use of both kinds of features induces some mathematical difficulties.
A categorical feature encodes the membership to one among various
mutually exclusive groups, without any concept of order between these
groups. On the contrary, a continuous variable intrinsically represents
the notion of order. Methods designed for continuous data cannot easily
apply to categorical ones, and vice versa.

In order to provide interpretability to predictive models, we particu-
larly focus on feature selection methods that extract variable importance
from the very structure of classifiers. Two families of classification al-
gorithms naturally allow the use of different variable types: approaches
based on decision trees and kernel methods. In this thesis, we improve
the interpretability of feature selection from tree ensemble methods and
propose two heterogeneous feature selection schemes with kernel meth-
ods.
1.5. Thesis focus

1.5.1 Tree ensembles: better feature selection

Tree ensemble methods such as Random Forest [Bre01] (RF) build several decision trees on different parts of a dataset. The final classifier uses a consensus decision to make new predictions. In comparison with a simple decision tree, this reduces overfitting and improves the classification performances. Tree ensembles offer a way to measure to which extent a variable is important in the classification model. This index is useful to rank variables according to the role they play in the decision making process. This directly leads to a feature selection algorithm.

Predictive models built from RF’s top-ranked features usually produce good predictive performances. However, that importance index suffers from two main drawbacks that make it bad from an interpretability point of view. Firstly, the ranking of the variables is highly unstable i.e. it may change a lot if the dataset changes a bit. Secondly, it is very difficult to decide which features actually play a role in the decision process. In particular, there is no easy way to define an importance threshold that separates relevant features from irrelevant ones. Indeed, this importance index expresses on a scale which is hardly interpretable.

This thesis provides solutions to these two problems. We perform an analysis of Random Forest stabilities and show how to improve feature selection stability. We propose a statistically interpretable feature importance index for tree ensemble methods. It outputs p-values that variables are important in the decision process. Altogether, this greatly improves the interpretability of feature selection from tree ensemble methods. In a biomedical context, a stable variable ranking guides the physicians’ investigations. Moreover, they can focus their research on variables that significantly play a role in the medical process.

1.5.2 Kernel methods: new feature selection schemes

The most simple classifiers perform classification by looking for a linear separation between two classes. Support Vector Machines [Vap95] (SVM) is a popular example that is particularly robust to overfitting. It chooses the hyperplane that maximises the margin between the training instances and the decision boundary. SVM can also perform efficient non-linear classification using implicit feature maps through kernels. It simply searches for a linear separation in a projected feature space.

The use of kernels is not restricted to continuous variables. They can be defined for a lot of data types e.g. graphs, pictures, strings, . . . , and of course, categorical data. Combining different kernels enables
classification from heterogeneous data. However, heterogeneous feature selection through kernels remained unexplored.

We ventured on this side and propose two feature selection methods that combine continuous and categorical kernels to handle both kinds of variables. They provide computationally efficient feature selections that reach state-of-the-art performances.

1.6 Summary of the contributions

This section introduces the contributions of this thesis. The main related papers are integrally transcribed in Part III.

**An analysis of tree ensemble methods stabilities** Tree ensemble classifiers are known to increase in predictive performances up to a certain point with the number of trees. In [PVD12], we also analyse how the individual sample class predictions and the stability of feature selection improve with the forest size. When dealing with high-dimensional datasets, we show that a very large number of trees are needed to obtain stable feature selection.

**Significant features from Random Forest** Random Forest already provides variable importance indices but those are not easily interpretable in a statistical sense. In [PVD13], we present a methodology to assess the statistical relevance of features inside a forest. In particular, we introduce \( J_{\chi^2} \), a statistically interpretable feature importance index. It outputs \( p \)-values that variables are important in a given forest. To do so, it computes a permutation test on out-of-bag instances inside the forest and assesses the level of significance with a Pearson’s \( \chi^2 \) test. In [PD15, PD14b], we compare \( J_{\chi^2} \) to two recent alternatives and show that our index has a lower computational complexity by an order of magnitude while keeping similar performances. The paper [PD15], published in the Neurocomputing journal, can be found in Part III. It is an extended version of the other published papers.

**Heterogeneous feature selection with kernels** In [PD14a] and [PDD15], we propose two kernel methods to perform heterogeneous feature selection. We use a dedicated kernel, that handles both continuous and categorical variables, and plug it into a recursive feature elimination (RFE) procedure in order to perform feature selection. One approach internally uses the kernel weights of a multiple kernel learning model to guide the RFE towards important variables. The other
method uses weights from a non-linear support vector machine. These new approaches reach state-of-the-art performances. The Neurocomputing journal paper [PDD15] can be found in the appendices. It is an extended version of [PD14a] which was presented at the ESANN’14 conference.

1.7 Publication list


1.8 Roadmap

This document takes the form of an article thesis. It is divided in three main parts.
Chapter 1. Introduction

The first part, entitled ‘Background and materials’, presents the context of this work as well as the necessary tools our approaches are based on. Chapter 2 defines what is classification and presents some existing approaches that perform classification of heterogeneous data. Then, Chapter 3 introduce feature selection as well as state of the art methods in the context of mixed variables. Chapter 4 is the last one of the first part. It describes the tools and procedures we use to assess the performances of feature selection methods.

The second part summarises the three contributions of this thesis. Its chapters highlight the main results that were published. Each chapter refers to one or several published papers. Chapter 5 gives an overview of an analysis of tree ensemble methods from a stability point of view. Chapter 6 introduces a statistically interpretable feature importance index for Random Forest. The last contribution is reported in Chapter 7. It presents two new heterogeneous feature selection methods based on kernels. A summary of all the contributions is given in Chapter 8 with a few possible extensions to this work as perspectives.

The third part is a collection of the three main papers published during this thesis. The additional papers listed in the publication list (Section 1.7) are shorter and preliminary versions of those longer articles.
Part I

Background and materials
Chapter 2

Classification

In order to perform classification, a predictive model is learned from training data. A model is an internal representation of the prediction problem that allows us to classify previously unseen samples. As the most important objective of classification methods is generalisation, a predictive model should mimic the real world as good as possible.

Lack of data, high dimensionality and noise are the three main opponents to this generalisation objective. Indeed, a predictive model learned from a finite amount of data can only represent a partial view of the real world problem. In addition, as explained in Section 1.3, high-dimensionality makes model learning even harder. Finally, one cannot consider that the instances available for training are perfectly neat. For instance, measures can be imprecise, some instances may be misclassified by experts, or some unobserved variables may play a role in the classification task. To summarise, fitting data perfectly leads to overfitting. Proper learning algorithms need additional hypotheses to estimate good predictive models. Those assumptions, such as the form of the decision boundary or the way to choose some parameter when learning a classifier, form what is called the inductive bias.

As this work targets heterogeneous data, we will focus on classifiers that can deal with such data. In particular, we detail tree ensemble and kernel methods on which the work of this thesis is based. This chapter is organised as follows. Section 2.1 introduces what is classification and some notations that will be useful through this document. Section 2.2 details tree ensemble classifiers. It first describes popular decision tree induction methods and then moves to ensemble learning. Afterwards, we turn our attention to kernel methods. Section 2.3 pictures Support Vector Machines and the famous kernel trick to perform
Chapter 2. Classification

non-linear classification and handle different data types. Section 2.4 describes Multiple Kernel Learning which both learns a predictive model and a kernel from the data. Section 2.5 depicts instance-based methods such as the nearest-neighbour classifier and the learning vector quantisation method. Section 2.6 explains the naive Bayes classifier that also provides a way to deal with continuous and categorical variables. Finally, Section 2.7 describes an alternative way to deal with both kinds of features, by resorting on a data recoding.

2.1 Definition and notations

The essence of supervised learning is to infer general rules from examples. The base material is a set of input data and their desired output. The goal is to find a mapping that not only maps the sample data to their output but also generalise well on new unseen data. In classification, the output variable encodes a class membership. A learning algorithm has to find out how to differentiate between samples of different classes. More formally, it has to estimate a decision function $f : D \rightarrow Y$ that maps the space of input data $D$ to response labels in $Y$ from a finite set of instances $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ where $x_i \in D$ and $y_i \in Y$.

We consider the common case where a dataset is representable as a matrix $X^{n \times p}$ where $n$ denotes the number of samples and $p$ the number of dimensions. The data point $x_i$ is a $p$-dimensional vector consisting of various measurements. It corresponds to the $i$-th line of the matrix. When referring to a particular dimension, we use the notation $x_j$. In the data matrix $X$, it corresponds to the $j$-th column. Each dimension can either be continuous ($x_j \in \mathbb{R}^n$) or categorical. In the second case, the variable can only take a finite set of values representing the various categories. There is no notion of order between those categories. A categorical variable should rather be seen as a way to partition instances into different groups. Finally, the value of instance $i$ for feature $j$ is written $x_{ij}$.

When dealing with classification problems, the response variable $y$ encodes the class labels. The notation $y$ refers to the vector that contains the $n$ labels. The class label associated with instance $x_i$ is denoted $y_i$. By nature, the label variable is categorical. However, when there are only two classes, some methods expect $y$ to be numerical and to encode the two class labels as $-1$ and $1$. Those cases are explicitly mentioned in the text.
2.2 Tree ensemble methods

Decision trees are popular classification methods because of their simplicity. They naturally handle continuous and categorical variables, which makes them good candidates for our purpose. However, even if they correctly model their input data, they have limited generalisation capabilities. In addition, decision tree induction methods are very sensitive to the training instances: slightly different data leads to very different predictive models. It is a form of overfitting. Ensemble methods such as Random Forest (RF) [Bre01] take advantage of this instability and drastically improve the predictive performances and robustness of the models by growing many decision trees and taking committee decisions.

Section 2.2.1 first describes decision trees in a general way. Then it pictures three specific tree induction methods: CART [BFOS84], C4.5 [Sal94] and cTree [HHZ06]. In Section 2.2.2, we give a generic description of tree ensemble methods. We then refine it for three particular cases: Bagging of trees [Bre96], Random Forest [Bre01] and Extremely Randomised Trees [GEW06]. Finally, Section 2.2.3 discuss some limitations of methods based on decision trees.

2.2.1 Decision trees

As we can see on Figure 2.1, a decision tree is a quite simple and easy to interpret predictive model. Each internal node tests one single variable. A branch corresponds to a set or a range of values for a feature. Each leaf is assigned a class label. In order to make predictions, new samples are brought down to the leaves.

Decision trees are particularly appreciated because they are easy to understand. Indeed, each path from the root node to a leaf can be interpreted as a decision rule. Since there is only one variable per node, the decision boundaries of such trees are very simple. Those are piecewise linear and parallel to the axes, as shown in Figure 2.2.

The induction of decision trees is a recursive process. A general pseudo-code is given in Algorithm 2.1. In each node, a splitting criterion is chosen from a learning set \( S \). It is a set of pairs \((x_i, y_i)\) of training samples and their associated class labels. In the root node, \( S \) contains all the training instances. The set \( F \) contains variables that are candidate for splitting. For deterministic tree induction methods, \( F \) usually contains all the features of the training instances in \( S \). The splitting rule is found by maximising a quality criterion in function \text{get-Split}. This criterion is generally a measure of how much the new split
Figure 2.1: Example of a decision tree with continuous and categorical attributes. In each internal node, the population is shown in the upper rectangle. The lower rectangle shows the variable which is chosen to make the split. Splitting rules are shown on the edges. The final population is shown in each leaf, along with its attributed class label in bold.

Figure 2.2: Two dimensional example of a decision boundary of a classification tree for two continuous variables.
2.2. Tree ensemble methods

Algorithm 2.1: DecisionTree($S$) : build a decision tree

$F \leftarrow \text{getVariables}(S)$ \hfill // set of all variables

$spli t \leftarrow \text{getSplit}(S,F)$ \hfill //variable + splitting rule

if stop($S$,split) then

| return leaf($S$) \hfill //create leaf node and assign a class

else

foreach partition $k$ according to split do

| $S_k \leftarrow$ instances of $S$ that belong to partition $k$

| $child_k \leftarrow$ DecisionTree($S_k$)

end

end

separates well instances of different classes with respect to the current population $S$. The getSplit function computes the quality of all possible splits of all variables in $F$ and the best splitting rule is kept in the split object. The samples of $S$ are then distributed among the child nodes according to this rule. The whole process is repeated until a stopping criterion is met and leaves are created.

There are various kinds of tree induction methods that mainly differ in the way they choose a variable and perform a split in a node (cf. getSplit function in Algorithm 2.1). It is indeed a crucial point since the tree building process is a greedy top-down search. Once a split is decided, this choice is never questioned ever after.

One difficulty when computing the best split is that both continuous and categorical variables need to be assessed. This leads to different strategies to prevent biasing the search towards a variable type or the other. Some training algorithms, such as C4.5 [Sal94], use multi-way splits for categorical variables and binary splits for continuous features. They resort on a specific evaluation criterion that takes into account the nature of the variables. Other methods, such as CART [BFOS84] or cTree [HHZ06], perform binary splits no matter the feature type. Those strategies are described in the remaining of this section.

CART decision trees

The CART [BFOS84] methodology was proposed by Breiman in 1984. It follows the simple idea that splitting a node should increase its purity, or more precisely, decrease its impurity. This concept of node impurity is based on the class labels of the instances inside a node. It is measured
Chapter 2. Classification

Figure 2.3: Gini index value with respect to the proportion among two classes.

by the following Gini index:

\[
Gini(S) = 1 - \sum_{c \in \text{classes}(S)} \left( \frac{|S_c|}{|S|} \right)^2,
\]

(2.1)

where \( S \) is the set of learning samples inside a node, \( S_c \) is the subset of samples of \( S \) that belong to class \( c \) and \( |.| \) is the set size operator. This index is minimal when the node contains only samples of one class. It reaches a maximum when all classes are equally represented in the set \( S \).

As shown in Figure 2.3 for a two class example, the gini index is 0 when the node is totally pure (only one class in the node) and its maximal value is 0.5 when the two classes appear in the same proportions.

The quality of a potential split is given by the difference between the impurity of the parent node and the sum of the impurities of the child nodes. However, if there is no constraints on the number of child nodes, a variable with many different values would artificially lead to a greater drop in node impurity. The more extreme case would be a variable that has a different value for each learning sample, which would lead to \(|S|\) child nodes. In that case the drop in Gini would be maximal but this choice most likely overfits the learning data. To overcome this problem, the CART algorithm builds binary decision trees. A graphical example of such tree is given in Figure 2.4. Every node is either a leaf or has
2.2. Tree ensemble methods

![Decision Tree Diagram]

Figure 2.4: Example of a binary decision tree with continuous and categorical attributes. In each internal node, the population is shown in the upper rectangle. The lower rectangle shows the variable which is chosen to make the split. Splitting rules are shown on the edges. The final population is shown in each leaf, along with its attributed class label in bold.

exactly two children, no matter the kind of variable used for splitting. The two children are often referred to as ‘left’ and ‘right’ child nodes. The split quality criterion is therefore defined as follows:

\[
\text{Drop}(S, S_l, S_r) = \text{Gini}(S) - \frac{|S_l|}{|S|} \text{Gini}(S_l) - \frac{|S_r|}{|S|} \text{Gini}(S_r),
\]

where \( S \) is the set of samples that lie in a given node and \( S_l \) (resp. \( S_r \)) is the subset of \( S \) that goes to the left (resp. right) child-node.

Two different strategies are needed to perform binary splits with continuous and categorical variables. Splits based on a continuous feature are made according to a threshold \( t \). The instances for which the variable of interest has a lower value go in one child node and those with a higher value go to the other. In the case of categorical variables, the set of their possible values is partitioned in two subsets. The instances are thus spread among the two child nodes by comparison to those two sets. Algorithm 2.2 refines the general decision tree description given in Algorithm 2.1 in the case of CART decision trees. It shows how binary splits are made for each variable type. In the pseudo-code, the
function getVariables(S) returns all the variables of the learning sample S i.e. \( F = \{ x_i \mid i \in [1,p] \} \). The function getSplit(S,F) measures the Gini drop (Equation 2.2) of all possible splits of all variables and returns the best splitting rule. For each continuous variable in \( F \), it sorts the instances in \( S \) and evaluates the \(|S| - 1\) thresholds that are midway between two consecutive samples. For the categorical features in \( F \), it computes the Gini drop for each different way to partition the feature values in two sets.\(^1\) To reduce overfitting, the CART procedure relies on a pre-pruning strategy. It prevents a node from further splitting when it is sufficiently pure (\( \text{Gini}(S) < \alpha \)) or when it contains less than a pre-defined number of samples (\(|S| < n_0\)), with \( \alpha \) and \( n_0 \) as meta-parameters. Function stop(S,split) halts the growing process whenever one of those criteria is met or if no further split is possible.

**Algorithm 2.2: CART(S) : build a CART decision tree**

\[
\begin{align*}
F & \leftarrow \text{getVariables}(S) \\
\text{split} & \leftarrow \text{getSplit}(S, F) \quad \text{//variable + splitting rule} \\
\text{if} \text{stop}(S, \text{split}) \text{ then} \\
& \quad \text{return leaf}(S) \quad \text{//create leaf node and assign a class} \\
\text{else} \\
& \quad x_j \leftarrow \text{split.variable} \\
& \quad \text{if } x_j \text{ is categorical then} \\
& \quad \quad L \leftarrow \text{split.leftValues} \quad \text{//subset of values of } x_j \text{ for left child node} \\
& \quad \quad S_{x_j \in L} \leftarrow \text{subset of } S \text{ that have } x_j \in L \\
& \quad \quad S_{x_j \notin L} \leftarrow S \setminus S_{x_j \in L} \\
& \quad \quad \text{child}_{x_j \in L} \leftarrow \text{CART}(S_{x_j \in L}) \\
& \quad \quad \text{child}_{x_j \notin L} \leftarrow \text{CART}(S_{x_j \notin L}) \\
& \quad \text{else} \\
& \quad \quad \text{// } x_j \text{ is continuous} \\
& \quad \quad \quad t \leftarrow \text{split.threshold} \\
& \quad \quad \quad S_{x_j \leq t} \leftarrow \text{instances of } S \text{ that have a value } \leq t \text{ for } x_j \\
& \quad \quad \quad S_{x_j > t} \leftarrow S \setminus S_{x_j \leq t} \\
& \quad \quad \quad \text{child}_{x_j \leq t} \leftarrow \text{CART}(S_{x_j \leq t}) \\
& \quad \quad \quad \text{child}_{x_j > t} \leftarrow \text{CART}(S_{x_j > t}) \\
& \quad \text{end} \\
\end{align*}
\]

\(^1\)For two-class problems, a possible optimisation is to sort the categories according to the probabilities of one class. The optimal split then lies between two positions of the ordered list.
2.2. Tree ensemble methods

Contrarily to the CART methodology, the C4.5 [Sal94] algorithm does not grow binary trees. It is an extension of the basic ID3 [Qui86] algorithm which only considers categorical variables. In that method, one child node is created per possible value of the splitting variable. C4.5 additionally handle continuous variables. This method is described hereafter.

Like CART, the idea is to recursively improve the classification of training samples while further splitting nodes. C4.5 uses the entropy as node quality criterion. Its mathematical definition is given hereafter:

$$\text{Entropy}(S) = \sum_{c \in \text{classes}(S)} -\frac{|S_c|}{|S|} \log \frac{|S_c|}{|S|}.$$  \hspace{1cm} (2.3)

This measure captures the quantity of information in knowing the class label of one sample of the set $S$. For a two-class problem, optimising the entropy is the same as optimising the Gini index (Equation 2.1). Those two measures behave the very same way, as shown in Figure 2.5. The scale is however different. The entropy culminates at 1 while the Gini index has a maximum of 0.5.

The big difference with the CART method is that C4.5 grows multi-way splits for categorical variables. Whenever such split occurs, one child

---

**Figure 2.5**: Comparison between the entropy and the Gini index with respect to the proportion among two classes.

**C4.5**

Contrarily to the CART methodology, the C4.5 [Sal94] algorithm does not grow binary trees. It is an extension of the basic ID3 [Qui86] algorithm which only considers categorical variables. In that method, one child node is created per possible value of the splitting variable. C4.5 additionally handle continuous variables. This method is described hereafter.

Like CART, the idea is to recursively improve the classification of training samples while further splitting nodes. C4.5 uses the entropy as node quality criterion. Its mathematical definition is given hereafter:

\[
\text{Entropy}(S) = \sum_{c \in \text{classes}(S)} -\frac{|S_c|}{|S|} \log \frac{|S_c|}{|S|}. \tag{2.3}
\]

This measure captures the quantity of information in knowing the class label of one sample of the set $S$. For a two-class problem, optimising the entropy is the same as optimising the Gini index (Equation 2.1). Those two measures behave the very same way, as shown in Figure 2.5. The scale is however different. The entropy culminates at 1 while the Gini index has a maximum of 0.5.

The big difference with the CART method is that C4.5 grows multi-way splits for categorical variables. Whenever such split occurs, one child
node is created per category. That way, categorical features appear only once in the decision tree. For continuous features, a binary split is made according to a threshold, similarly to CART. As explained in previous section, this varying number of child nodes calls for an adapted metric in order to assess the split quality: the \textit{GainRatio} index. It is defined as:

\[
\text{Gain}(S, \{S_1, \ldots, S_K\}) = \text{Entropy}(S) - \sum_{k=1}^{K} \frac{|S_k|}{|S|} \text{Entropy}(S_k),
\]

\[
\text{SplitInfo}(S, \{S_1, \ldots, S_K\}) = \sum_{k=1}^{K} -\frac{|S_k|}{|S|} \log \frac{|S_k|}{|S|},
\]

\[
\text{GainRatio}(S, \{S_1, \ldots, S_K\}) = \frac{\text{Gain}(S, \{S_1, \ldots, S_K\})}{\text{SplitInfo}(S, \{S_1, \ldots, S_K\})},
\]

where \text{Entropy} is defined like in Equation 2.3, \(|\cdot|\) denotes the set size operator and the sets \(S_1, \ldots, S_K\) are the respective populations of the \(K\) child nodes. Those sets form a partition of \(S\). \text{Gain} computes the gain in entropy with respect to the class labels when splitting. It is equivalent to the Gini drop (Equation 2.2) in the CART methodology. \text{SplitInfo} computes the entropy of the split with respect to the values of the splitting attribute. It can be seen as the quantity of information that one gets in knowing the child node in which one sample of \(S\) falls into. Finally, \text{GainRatio} is a rescaled gain in entropy that discourages the use of attributes with many values by dividing the gain by the entropy of the split.

The general pseudo-code to grow C4.5 trees is given in Algorithm 2.3. It details how continuous and categorical splits are handled. In comparison with Algorithm 2.2, only the categorical split part changes. The C4.5 procedure basically builds full trees. The \textit{stop} function only halts the tree growing process whenever no additional split is possible. In order to reduce overfitting, a post-pruning strategy based on cross-validation removes nodes in the tree afterwards.

\textbf{Conditionally independent recursive partitioning}

The choice of splitting variables is essential during the tree growing process. It greatly influences the generalisation capabilities of decision trees. However, simple experiments exhibit some biases that depend on the variable type for popular tree learning methods. For instance in [Loh10], the author shows that C4.5 and CART both prefer categorical variables with a high number of possible values. CART also favour continuous features over variables with few categories. Those biases
Algorithm 2.3: C4.5($S$) : build a C4.5 decision tree

\begin{algorithm}
\begin{algorithmic}
\State $F \leftarrow \text{getVariables}(S)$
\State $\text{split} \leftarrow \text{getSplit}(S,F)$ \hfill //variable + splitting rule
\If {\text{stop}(S,\text{split})}
\State \Return leaf($S$) \hfill //create leaf node and assign a class
\Else
\State $x_j \leftarrow \text{split}.\text{variable}$
\If {$x_j$ is categorical}
\State \ForEach {$v \in x_j$}
\State $S_{x_j=v} \leftarrow$ instances of $S$ that have value $v$ for $x_j$
\State $\text{child}_{x_j=v} \leftarrow \text{C4.5}(S_{x_j=v})$
\EndFor
\Else
\State // $x_j$ is continuous
\State $t \leftarrow \text{split}.\text{threshold}$
\State $S_{x_j \leq t} \leftarrow$ instances of $S$ that have a value $\leq t$ for $x_j$
\State $S_{x_j > t} \leftarrow S \setminus S_{x_j \leq t}$
\State $\text{child}_{x_j \leq t} \leftarrow \text{C4.5}(S_{x_j \leq t})$
\State $\text{child}_{x_j > t} \leftarrow \text{C4.5}(S_{x_j > t})$
\EndIf
\EndIf
\end{algorithmic}
\end{algorithm}
come from the search over all possible splits for each variable. Features with more possible splits have a higher chance to maximise the splitting criterion.

To neutralise this bias, a conditional inference framework for recursive partitioning was proposed in [HHZ06]. It relies on a conditional independence test that is performed for each variable in each node of the tree. It tests if variables are independent with respect to the class labels.

Basically, the pseudo-code for growing those trees is similar to CART in Algorithm 2.2. The only differences lie in getSplit and stop where these permutation tests take place both to choose the splitting variable and to halt the process when no feature is significantly dependent of the class labels. From a usability point of view, even if the computational complexity is similar to growing traditional CART trees, the conditional independence tests performed for each variable in each node significantly increase the computational time.

2.2.2 Ensemble of trees

Ensemble methods are based on the idea that asking a committee of experts is better than asking only one. Because of their personal background and experience, experts may have different opinions about the same problem. They would need to argue and discuss to come up with a common solution that takes into account the various views.

The key aspect in such kind of strategy is to mix different points of view on the problem to reach a consensus. With ensemble classification methods, this is mimicked by building several base learners and promoting diversity among them. In bagging [Bre96] for instance, base classifiers are built from different subsets of the training data. The global decision to classify a new sample is taken democratically from all base learners, by a majority vote. This prevents overfitting and approximates the true class distribution better.

Ensemble methods particularly improve predictive performances over a simple classifier when the base predictors are very sensitive to changes in the dataset. They were even shown to benefit from base classifiers that overfit the learning data [SK96]. That is why several successful ensemble methods are based on unpruned decision trees. This allows us to build more complex decision frontiers while being robust to overfitting (cf. Figure 2.6). However, the main drawback is that predictive models become more complicated. In particular, if a single decision tree is quite easy to interpret in terms of decision rules, it is more difficult to gain insight into an ensemble of trees.
2.2. Tree ensemble methods

Figure 2.6: Decision boundaries of a single CART decision tree and a Random Forest. Top: models built on a random two-dimensional subset of Fisher’s Iris [Fis36] dataset. Bottom: 2D artificial dataset where the true class frontier is a circle (classes shown in white and grey).
A generic pseudo-code for growing an ensemble of decision trees from a learning set $S = \{(x_i, y_i) \mid i \in [1, n]\}$ is given in Algorithm 2.4. The sample function is intentionally left abstract. It could return the full set of instances $S$ or a sub-sample, with or without repeated elements.

**Algorithm 2.4: BuildTreeEnsemble($S$) : build an ensemble of $T$ decision trees**

```plaintext
for $k = 1$ to $T$
    $B_k \leftarrow$ sample($S$) // bag
    $\overline{B}_k \leftarrow S \setminus B_k$ // out-of-bag
    $h_k \leftarrow$ BuildTree($B_k$) // see Algorithm 2.1
end
```

Tree ensemble methods typically grow unpruned decision trees to increase their high variability. In Algorithm 2.1, the training instances are partitioned until no additional split is possible (cf. stop function). In order to grow different base learners, popular tree ensemble methods introduce some randomness in the forest growing process. There are essentially three key points which can be randomised:

1. the bag (set of training instances) of each tree  
   cf. $B_k \leftarrow$ sample($S$) in Algorithm 2.4

2. the set $F$ of candidate variables for splitting in each node  
   cf. $F \leftarrow$ getVariables($S$) in Algorithm 2.1

3. the way to define the split in each node  
   cf. $split \leftarrow$ getSplit($S,F$) in Algorithm 2.1

Three major approaches, Bagging [Bre96], Random Forest (RF) [Bre01] and Extremely Randomised Trees (Extra-Trees) [GEW06] fall into this framework. Bagging builds multiple trees from randomly selected training instances (point 1). The RF algorithm relies on points 1 and 2 while Extra Trees injects randomness in points 2 and 3.

All those methods grow several trees to form an ensemble classifier that makes predictions based on a majority vote. Even though they could use any kind of decision trees as base learners, they generally grow CART trees. The three methods are described hereafter.

**Bagging**

The bagging procedure was proposed by Breiman in [Bre96] as a simple way to improve predictive performances over single classifiers. Its name
2.2. Tree ensemble methods

is a shortened version of ‘bootstrap aggregating’. While it is originally defined for all kinds of predictor, we focus here on bagging of decision trees.

The bagging method grows $T$ decision trees from different learning sets, or bags. Since they are highly sensitive to the training data, the decision trees are expected to be different and to model different parts of the data. In Algorithm 2.4, each bag $B_k$ is built from $n$ samples which are drawn uniformly at random from the dataset $S$, with replacement. It results in bags containing as much training instances as the original training set. This sampling procedure is known as bootstrap [ET94]. Because samples are taken with replacement, each $B_k$ contains a subset of $S$ with some instances appearing multiple times. Indeed, each sample has roughly a 63.2% chance to be part of a specific bag. The instances that are not used to grow the $k$-th tree form the out-of-bag $B_k$. They can be used to obtain internal estimates of the aggregated predictor accuracy.

**Random forest**

To promote diversity among its decision trees, RF [Bre01] introduces randomness in two different ways. Firstly, following the bagging strategy [Bre96], the learning set $B_k$ of each tree is made from a bootstrap sample [ET94] of the training set $S$ (see Algorithm 2.4). Decision trees are fully grown, following the CART procedure (cf. Algorithm 2.2), without any kind of pruning. Secondly, the splitting criterion is also randomised. In each node of each tree, a subset $F$ of $m \leq p$ candidate variables is sampled uniformly at random, without replacement. The methodology to find the best split is the same as for the original CART algorithm (see Section 2.2.1) i.e. the drop in node impurity is maximised. However, only $m$ randomly sampled features are considered instead of the whole set of $p$ variables. This reduces the possibilities to find the best split which further increases the variability in the tree building process. The number $m$ of sampled features is generally quite small with respect to the total number of dimensions $p$. It is typically set to $m = \sqrt{p}$.

RF were shown to be very efficient classifiers with very few meta-parameters to tune. In particular, the predictive performances increase with the number of trees up to a certain point. The number $m$ of variables to be considered for splitting a node is quite robust. A forest of a few hundreds trees with a default value of $\sqrt{p}$ for parameter $m$ usually performs well for most classification tasks. Those elements make RF a very good first candidate to test if a dataset contains some signal about
class labels.

An ensemble of trees is much more difficult to interpret than a single decision tree. To remedy this situation, the author of the original RF paper [Bre01] proposes a way to estimate the importance of the different variables in the decision process. This is detailed in Chapter 3.

**Extremely randomised trees**

Like RF, Extra-Trees [GEW06] grows CART-like decision trees. Yet, this algorithm has a slightly different randomisation scheme. It replaces the bootstrap sampling of instances to form each bag by a more random split selection in each node.

With Extra-Trees, each tree is grown from the full learning sample \( S \) i.e. \( B_k = S = \{(x_i, y_i)\}_{i=1}^n \) in Algorithm 2.4. In order to grow different decision trees, some randomness is introduced when choosing the decision rule in each node of each tree. First, like in RF, \( m \) variables are chosen at random to form the set \( F \) of candidate variables for splitting. Then, for each variable in \( F \), a random cut-point is chosen. Among the \( m \) candidate decision rules, the one which maximises the drop in node impurity (Equation 2.2) is kept as the splitting rule of that particular node.

In comparison with RF, the splitting criterion in each node of each tree is thus more randomised. Indeed, RF evaluates all possible splits for each of the candidate features in \( F \) while Extra-Trees only considers one possible split per feature. This somehow compensates the fact that, unlike RF, Extra-trees grows all base learners from the same learning sample.

**2.2.3 Limitations of approaches based on decision trees**

As previously mentioned, the choice of the splitting variable in each node of a tree may be biased with respect to the nature of the candidate features. This problem is addressed in [HHZ06] where an unbiased scheme is proposed for tree induction at the expense of the computational time.

Another limitation comes from the very induction principle of decision trees. They may fail to uncover the class signal when it is ‘too multivariate’. Indeed, the recursive partitioning of the data follows a greedy selection mechanism. The first variable at the root node is chosen from a strictly univariate criterion. Deeper in the tree, the population inside a node is conditioned by the splits above it. However, the splitting variable will still be the one with the biggest univariate effect on that population. It follows that a multivariate class signal (i.e. that one would get by jointly considering several features) might be masked
2.3. Support vector machines

because a weaker but univariate signal is preferred by the training algorithm.

An illustrative but quite artificial example is given in Figure 2.7. This is a 3 dimensional classification problem with two classes. On the one hand, features \(x_1\) and \(x_2\) lead to perfect classification when considered jointly. Yet, there is no class information when those features are taken individually. On the other hand, variable \(x_3\) convey some class signal but it cannot perfectly predict the two labels on its own. In this case, a tree learning algorithm would select \(x_3\) as splitting variable for the root node, possibly breaking the multivariate signal in \(x_1\) and \(x_2\).

We observe the same drawback with tree ensemble classifiers. While the random sample selection at the core of bagging methods could induce some marginal effect in features such as \(x_1\) and \(x_2\), ensemble approaches also miss strong multivariate interactions in the presence of other noisy variables [AN09]. It is particularly true for high dimensional datasets with few samples, such as biomedical data. Some, more complex, alternatives try to palliate this problem. For instance, one can rely on an exhaustive search of the pairs of possible interacting variables [AN09]. It is also possible to build multivariate splits in each node e.g. with a regression [MKS+11].

2.3 Support vector machines

Support Vector Machine (SVM) [BGV92] is arguably one of the most famous classification algorithms. It is an elegant method that produces state-of-the-art performances, especially with high dimensional biomedical data [BHOS+08, Muk03]. In addition, it is quite resistant to overfitting and produces very lightweight models. In this section, we give an overview of the SVM classifier from a mathematical point of view. The interested reader is referred to the book of Christopher Bishop [Bis07] for more details. We will first describe linear SVM (Section 2.3.1). Then, Section 2.3.2 explains how to perform non-linear classification by projecting the data into a new feature space through the kernel trick. Finally, Section 2.3.3 will present soft-margin SVM that can find a predictive model even if the data is not linearly separable in the feature space.

2.3.1 Linear SVM

In its simplest form, SVM performs binary classification by building a hyperplane that separates two different classes. Such boundary is called a linear discriminant. In two dimensions it is a straight line.
Figure 2.7: Classification problem with 3 variables. The two classes are represented by the different dot and line styles.
2.3. Support vector machines

In three dimensions, it is a plane. In $p$ dimensions, it’s a subspace of dimensionality $p - 1$. In this section, we consider that the two class labels are scalars and take values 1 and $-1$.

Formally, for a binary classification problem, a linear discriminant can be described as a function of a data sample $x \in \mathbb{R}^p$

$$g(x) = \langle w, x \rangle + w_0,$$  \hspace{1cm} (2.7)

where $w$ is a weight vector in $\mathbb{R}^p$, $w_0$ a scalar and $\langle \cdot , \cdot \rangle$ denotes the scalar product. When $g(x) = 0$, the sample $x$ lies on the separating hyperplane. The decision rule to classify new samples is

$$f(x) = \text{sign} \left( g(x) \right)$$

$$= \begin{cases} 
-1 & \text{if } g(x) < 0 \\
1 & \text{if } g(x) > 0.
\end{cases}$$ \hspace{1cm} (2.8)

In order to reduce overfitting, SVM chooses the maximal margin hyperplane. For a given set of points, the margin is the distance between the hyperplane and its closest points. Those points are the support vectors. They are the only ones that actually matter to express the decision function.

The SVM method is formulated as an optimisation problem. It is detailed hereafter.

Primal

The SVM algorithm looks for the separating hyperplane that has the largest margin under constraints that it separates well instances from different classes. By convention, the support vector lie at a distance $g(x) = 1$ from the linear decision boundary. The size of the margin is thus $\frac{1}{\|w\|}$. A graphical example of maximal margin hyperplane is given in Figure 2.8.

In order to maximise the margin, one has to minimise $\|w\|$. For mathematical convenience, we minimise $\frac{1}{2}\|w\|^2$ which is an equivalent objective. The SVM optimisation problem is defined as follows:

$$\min_{w} \frac{1}{2}\|w\|^2$$

s.t. $y_i(\langle w, x_i \rangle + w_0) \geq 1$, $\forall i \in [1, n]$,

$$\text{minimize } \frac{1}{2}||w||^2$$

$$\text{subject to } y_i(\langle w, x_i \rangle + w_0) \geq 1, \forall i \in [1, n],$$ \hspace{1cm} (2.9)

where the constraints simply state that the training samples have to lie in the correct side of the hyperplane, outside of the margin.
Chapter 2. Classification

Maximal margin hyperplane

Figure 2.8: Two dimensional example of a linear separator that maximizes the distance between points of different classes. The margin is shown in dashed lines. White dots represent the positive class; black dots, the negative one. There are three support vectors. The two positive samples on the left dashed line and one point of the negative class on the right dashed line.

Dual

The constrained SVM optimisation problem of Equation 2.9 can be restated as an unconstrained equivalent by introducing Lagrange multipliers $\alpha_i \geq 0$ for each primal constraint. The following Lagrangian has to be minimised with respect to the primal variables $w$ and $w_0$ and maximised with respect to the $\alpha_i$:

$$L(w, w_0, \alpha) = \frac{1}{2} ||w||^2 - \sum_{i=1}^{n} \alpha_i (y_i (\langle w, x_i \rangle + w_0) - 1)$$  \hspace{1cm} (2.10)

By setting the partial derivative $\frac{\partial}{\partial w_0} L$ and $\frac{\partial}{\partial w} L$ to 0 and eliminating the primal variables, we obtain the dual form of the SVM optimisation problem:

$$\max_{\alpha} \quad W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

s.t. \hspace{1cm} $\alpha_i \geq 0$ \hspace{1cm} $\sum_{i=1}^{n} \alpha_i y_i = 0.$  \hspace{1cm} (2.11)
Contrarily to the primal formulation which looks for an optimal vector $w$ in $p$ dimensions, the dual optimisation problem depends on the number $n$ of input samples. This form is particularly convenient when dealing with biomedical data which are very high dimensional but generally have a limited number of samples ($n \ll p$).

In the dual formulation, the decision boundary and the decision function of the SVM can be expressed as follows:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i = \sum_{x_i \in SV} \alpha_i y_i x_i$$  \hspace{1cm} (2.12)

$$f(x) = \text{sign} \left( \sum_{x_i \in SV} \alpha_i y_i \langle x_i, x \rangle + w_0 \right).$$  \hspace{1cm} (2.13)

where $SV$ is the set of support vectors. As we can see in Equation 2.12, $w$ can be expressed only in terms of the support vectors. This is the result of the sparsity enforced in $\alpha$ combined to the Karush-Kuhn-Tucker conditions \cite{Kar39, KT51}. The scalar $w_0$ is computed from the fact that support vectors lie at a distance 1 from the hyperplane

$$y_i(\langle w, x_i \rangle + w_0) = 1,$$  \hspace{1cm} (2.14)

with $x_i$ a support vector.

### 2.3.2 Non-linear classification with the kernel trick

By design, the SVM algorithm builds a hyperplane decision boundary. However, the input data may not be linearly separable. The clever trick to be able to use SVM on those cases is to remap the data into a new space where they are more likely to be linearly separable. Let $\phi$ be a non-linear projection function that makes the input data linearly separable (example shown in Figure 2.9). The SVM dual objective function becomes

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle.$$  \hspace{1cm} (2.15)

However, we do not need to define explicitly such mapping function $\phi(x)$. One can leverage the fact that it only appears in the form of a scalar product and define a kernel instead. This can be seen as a symmetric similarity function defined as follows:

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle = k(x_j, x_i).$$  \hspace{1cm} (2.16)
Figure 2.9: Top: the sample data distribution is not linearly separable. Bottom: after undergoing a remapping $\phi$, it is linearly separable in a new feature space.
2.3. Support vector machines

As soon as the matrix $[k(x_i, x_j)]_{i,j=1}^{n}$ is positive semi-definite (cf. Mercer conditions [SS01]) the existence of a feature mapping $\phi$ is guaranteed. It means that we can directly define a similarity function $k$ without explicitly defining a feature mapping $\phi$. This also means that the training data no longer need to be continuous. All we need is a function to compare pairs of samples. For instance, kernels can be defined for pictures, graphs, character strings and even categorical data.

With kernels, the dual SVM optimisation problem and the decision function become

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \quad (2.17)$$

$$f(x) = \text{sign} \left( \sum_{x_i \in SV} \alpha_i y_i k(x_i, x) + w_0 \right). \quad (2.18)$$

A lot of kernels were designed for different purposes. The book ‘Learning with Kernels’ [SS01] gives a good overview of tens of them. The most simple one is the linear kernel which keeps data in their original space. It is defined as follows:

$$k(x_i, x_j) = \langle x_i, x_j \rangle. \quad (2.19)$$

Its explicit feature mapping $\phi$ is thus the identity function. A SVM that uses this kernel performs linear classification. Another very successful kernel projects data in a feature space of infinite dimensionality: the Gaussian radial basis function kernel

$$k(x_i, x_j) = \exp \left( -\frac{\|x_i - x_j\|^2}{2\sigma^2} \right), \quad (2.20)$$

where $\sigma$ is a meta-parameter. Due to the exponential, the kernel values range from 1 when $x_i$ is equal to $x_j$ to 0 (in the limit) when the two points are far apart.

In order to handle heterogeneous data, the so-called clinical kernel [DDM09] averages univariate sub-kernels for each variable. It is based on a similarity measure proposed by Gower in [Gow71]. Like the Gaussian kernel, its values range from 1 for two identical points to 0 for two most distant points. Here is its definition:

$$k(x_i, x_j) = \frac{1}{p} \sum_{f=1}^{p} k_f(x_{if}, x_{jf}) \quad (2.21)$$

$$k_f(a, b) = \begin{cases} I(a = b) & \text{if } x_f \text{ is categorical} \\ \frac{(\max_f - \min_f) - |a - b|}{\max_f - \min_f} & \text{if } x_f \text{ is continuous}. \end{cases} \quad (2.22)$$
where $I$ is the indicator function and $\max_f$ (resp. $\min_f$) is the maximal (resp. minimal) value of feature $x_f$. It is interesting to note that each sub-kernel $k_f$ expresses on the same range and that the final kernel is a simple average of the sub-kernels, giving the same importance to each original feature. However, the standard deviations of the $k_f$ are different for continuous and categorical variables [Gow71]. Yet, this kernel was successfully used to perform heterogeneous data classification. It is a building piece of two feature selection methods developed in this thesis.

### 2.3.3 Soft-margin SVM

The SVM method described in Sections 2.3.1 and 2.3.2 assumes that the data can be linearly separated either in the original space or in a new feature space. However, it might not be the case for some data distributions. In addition some mislabelled instances may mess up the search for the separating hyperplane. The same goes for outlying samples that have a very particular profile with respect to the other data points of the same class. In those cases, the optimisation problems of Equations 2.9 and 2.11 admit no solution.

To overcome these problems, soft-margin SVM [CV95] relaxes the constraints of Equation 2.9 (page 31) by allowing some samples to lie inside the margin or even on the wrong side of the decision boundary. The primal optimisation problem becomes

$$
\min_{w, \xi} \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad y_i((w, \phi(x_i)) + w_0) \geq 1 - \xi_i \\
\xi_i \geq 0,
$$

(2.23)

where $C$ is a positive constant and the $\xi_i$ are slack variables. Increasing them allows the constraints to always be met. When $0 < \xi_i < 1$, sample $x_i$ is correctly classified but within the margin. When $\xi_i > 1$, $x_i$ is misclassified. A graphical representation of those slack variables is shown in Figure 2.10.

It is important to note that this relaxation introduces a meta-parameter $C$. It determines the balance between two objectives. When $C$ is very small, the method will look for a maximal margin hyperplane at the expense of a large number of misclassified instances. When $C$ is big, it is very important to minimise the sum of $\xi_i$ i.e. training samples must be well classified even if the margin has to be smaller.
Figure 2.10: The soft-margin SVM allows some learning instances to be misclassified or within the margin through slack variables $\xi_i$. Those are represented with dotted lines.

The soft-margin SVM dual is the following one

$$\max_\alpha W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

$$s.t. \quad \alpha_i \in [0, C]$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$  \hfill (2.24)

Compared to Equation 2.11, only the constraints on the domain of each $\alpha_i$ change. Their interpretation is also a bit different. As in the hard-margin dual, $\alpha_i = 0$ means that $x_i$ is well classified. However, support vectors are identified by $\alpha_i$ lying in $[0, C]$. Points for which $\alpha_i = C$ are margin errors. They can be either well classified but inside the margin ($\xi_i < 1$), or misclassified ($\xi_i > 1$).

### 2.4 Multiple kernel learning

Similarly to the non-linear SVM (Section 2.3.2), Multiple Kernel Learning (MKL) [LDBC+04] learns a linear discriminant in a feature space induced by a kernel. But, in addition, it also learns the kernel itself. More precisely, it builds a final kernel $k$ from a weighted sum of $M$ base
kernels
\[ k(x_i, x_j) = \sum_{m=1}^{M} \mu_m k_m(x_i, x_j) \quad \text{s.t.} \quad \mu_m \geq 0. \]  
(2.25)

In order to understand what a sum of kernels is, one can have a look at Equation 2.16 (page 33). From the definition, we can rewrite such a sum in terms of scalar products in the following way:
\[ k_m(x_i, x_j) + k_{m'}(x_i, x_j) = \langle \phi_m(x_i), \phi_m(x_j) \rangle + \langle \phi_{m'}(x_i), \phi_{m'}(x_j) \rangle \\
= \langle \phi_m(x_i) \phi_{m'}(x_i), \phi_m(x_j) \phi_{m'}(x_j) \rangle, \]
(2.26)

where \( \phi_m(x) \phi_{m'}(x) \) is the concatenation of the two feature vectors. Summing kernels is thus equivalent to concatenating features in the projected feature space.

There exist different versions of MKL. The formulation in [BLJ04] promotes sparse kernel mixtures so that the final kernel is a weighted sum of a very limited number of base kernels (few \( \mu_i > 0 \) in Equation 2.25). Yet, we focus on a non-sparse alternative [KBL+09] that is shown to be more effective in high-dimensional biomedical problems. This approach uses a \( l_2 \)-norm regularisation on the vector of kernel weights \( \mu = [\mu_1, \ldots, \mu_M] \).

The primal formulation of the soft-margin \( l_2 \)-MKL is given hereafter.

\[
\min_{w_m, w_0, \mu_m, \xi_i} \frac{1}{2} \sum_{m=1}^{M} \|w_m\|^2 + C \sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad y_i \left( \sum_{m=1}^{M} \sqrt{\mu_m} \langle w_m, \phi_m(x_i) \rangle + w_0 \right) \geq 1 - \xi_i \\
\xi_i \geq 0 \\
\mu_m \geq 0 \\
\|\mu\|^2 \leq 1,
\]
(2.27)

where \( C \) is a positive constant, \( w_m, \mu_m \) and \( \phi_m \) are respectively the feature weight vector, the kernel weight and the explicit feature mapping corresponding to kernel \( k_m \), \( w_0 \) is the bias term (a scalar) and \( \xi_i \) are the slack variables.

The decision function of MKL is a generalised linear model in the feature space. It is given hereafter:
\[ f(x) = \text{sign} \left( \sum_{m=1}^{M} \sqrt{\mu_m} \langle w_m, \phi_m(x) \rangle + w_0 \right). \]
(2.28)
2.5 Instance-based and prototype-based classifiers

The nearest neighbours (k-NN) [Bis07] classifiers are among the most simple predictors. They essentially consist of one simple decision rule. In order to classify a new sample, one computes the distance between this point and all the training samples according to a distance $d$. A majority vote among the $k$ nearest points according to $d$ gives the predicted class. No predictive model is ever learned from the data. There are only two parameters to fix: the distance $d$ and the number $k$ of nearest neighbours to perform prediction. Figure 2.11 shows an example of a 5-NN classifier.

Parameter $k$ acts like a smoothing factor. When it is small, the

---

Figure 2.11: In order to classify a new data point (grey dot with interrogation mark), we look at the class labels of the 5 closest training instances. Here the predicted class, taken from a majority vote, is ‘white’. This problem is closely related to the soft-margin SVM described in Equation 2.23. Similarly, one can derive a dual optimisation problem by introducing a Lagrangian and eliminating primal variables by setting the various derivatives to zero. Since this formulation is not used later in this work, we do not detail those mathematical developments any further. The interested reader is referred to [KBL+09] for more information about the MKL dual problem.
classifier tends to overfit the training data and the space is divided into many small regions of different classes. When it is bigger, the decision boundaries split the space into fewer and larger regions making the classifier more robust. However, predictive performances will also drop if $k$ is too big and the decision regions are ‘too smooth’. This parameter requires a fine tuning.

Learning vector quantisation (LVQ) [Koh01] is another approach, closely related to $k$-NN. Instead of recording all the original training instances, the algorithm learns a set of representative prototypes in the observed data space. The goal is to find prototypes that minimise the distance between them and the training instances of the same class while maximising the distance to samples of the other class. The label of a new sample to classify is given by the closest prototype. Similarly to $k$-NN methods, the meta-parameters of LVQ include a distance $d$, both to learn the prototypes and to classify new data, and a number $\rho$ of prototypes per class. However, parameter $\rho$ plays the opposite role of $k$. If it is small, the decision rule partitions the input space in a few regions around the prototypes. When $\rho$ is big, there are many regions and the model tends to overfit the data.

In such kind of approaches, the choice of a specific distance $d$ greatly influences the decisions of the predictive model. When dealing with continuous variables, $d$ is often defined as the Euclidean distance which is recalled here:

$$d(x_i, x_j) = \|x_i - x_j\| = \sqrt{\sum_{f=1}^{p} (x_{if} - x_{jf})^2}. \quad (2.29)$$

Yet, different choices can be made to handle other data types and even heterogeneous data. For instance, the authors of [DV11] propose to use the heterogeneous euclidean-overlap metric (HEOM) [WM97] to cope with both continuous and categorical variables. It is defined as follows:

$$d(x_i, x_j) = \sqrt{\sum_{f=1}^{p} d_f(x_{if}, x_{jf})^2}. \quad (2.30)$$

$$d_f(a, b) = \begin{cases} I(a \neq b) & \text{if } x_f \text{ is categorical} \\ \frac{|a - b|}{\text{max}_f - \text{min}_f} & \text{if } x_f \text{ is continuous}, \end{cases} \quad (2.31)$$

where $\text{max}_f$ and $\text{min}_f$ are respectively the maximal and minimal observed value for feature $x_f$. As we can see, this distance is quite similar to the euclidean distance of Equation 2.29. It differs in the way it computes similarities at the feature level. HEOM uses different functions
2.6. Naive Bayes

d_f according to the nature of the variables. It is interesting to note that each auxiliary function \(d_f\) returns a feature based distance in the interval \([0, 1]\) in the objective of making them comparable in the global computation of the distance (cf. Equation 2.30). One can also note the close link between the partial distances \(d_f\) and the sub-kernels \(k_f\) of the clinical kernel (cf. Equation 2.22, page 35 in Section 2.3.2). Indeed, \(d_f(a, b) = 1 - k_f(a, b)\). It is also closely related to the similarity measure proposed in [Gow71].

Another way to benefit from different kinds of data consists in averaging several distances. For instance, the metric considered in [FHLD+11, LHLG+15] handles both continuous and binary variables. It is defined as a weighted sum of the Pearson correlation coefficient [Pea95] computed on the continuous features and the Euclidean distance (see Equation 2.29) computed on the binary variables encoded with values 0 and 1. They however have to correctly tune the weighting of the two metrics e.g. resorting on a cross-validation.

Finally, it is worth mentioning that distance-based methods can benefit from kernel approaches (see Section 2.3). Indeed, one can leverage the fact that the euclidean distance between two instances in the kernel space can be computed in the following way [Sch01]:

\[
d(x_i, x_j) = \sqrt{-k(x_i, x_j) + \frac{1}{2} (k(x_i, x_i) + k(x_j, x_j))}
\]

where \(k\) is a positive definite kernel and \(\phi\) is the explicit mapping function that projects samples into the kernel space. Therefore, we can use \(k\)-NN and similar methods with any kind of data, provided that there is a suitable kernel to handle them.

2.6 Naive Bayes

The naive Bayes classifier (NB) [Bis07] is very popular in text categorisation. It is a simple application of Bayes’ theorem combined to a strong hypothesis of variable independence.

According to Bayes’ theorem, the posterior probability of class \(c\) of data point \(x_i\) can be written as

\[
P(c|x_i) = \frac{P(x_i|c)P(c)}{P(x_i)},
\]

where \(P(x_i|c)\) is the likelihood of \(x_i\) being in class \(c\), \(P(c)\) the prior probability of class \(c\) and \(P(x_i)\) is the probability of the data. In those
conditions, the maximum a posteriori decision function would be the following one:

\[ f(x_i) = \arg\max_{c \in \text{classes}(y)} P(x_i|c)P(c) \]

\[ = \arg\max_{c \in \text{classes}(y)} P(x_i|c)P(c). \]

(2.34)

(2.35)

In order to compute \( P(x_i|c) \), NB makes the strong assumption that each feature is independent of the others, conditionally to class \( c \). Under that hypothesis, the decision function reads like this:

\[ f(x_i) = \arg\max_{c \in \text{classes}(y)} P(c) \prod_{j=1}^{p} P_j(x_{ij}|c), \]

(2.36)

where \( P_j(x_{ij}|c) \) is the probability of feature \( x_j \) to take value \( x_{ij} \) given that \( x_i \) belongs to class \( c \).

In order to train the model, one needs to estimate the class probabilities and each \( P_j(x_{ij}|c) \). Those probabilities are directly computed from the training set. When variable \( x_j \) is categorical, \( P_j(x_{ij}|c) \) is estimated from a contingency table that counts the occurrences of its different values per class. If \( x_j \) is continuous, it is more complicated. One typically assumes that the feature follows a given distribution per class e.g. a normal distribution. In that case, each \( P_j(x_{ij}|c) \) is replaced by an evaluation of the following density function

\[ f_j(x_{ij}|c) = \frac{1}{\sigma_{jc}\sqrt{2\pi}} \exp\left(-\frac{(x_{ij} - \mu_{jc})^2}{2\sigma_{jc}^2}\right), \]

(2.37)

where \( \mu_{jc} \) and \( \sigma_{jc} \) are the mean and standard deviation of feature \( x_j \) for the samples of class \( c \).

### 2.7 Other classifiers

Some classification methods only handle one kind of feature by design. For instance, most linear classifiers exclusively deal with continuous variables because their decision functions are based on a linear combination of the input features. Approaches such as the linear discriminant analysis (LDA), the quadratic discriminant analysis (QDA), the perceptron, the logistic regression (see [Bis07] for a description of those methods) and the linear SVM (see Section 2.3.1) look for a linear decision boundary in \( \mathbb{R}^p \), the continuous feature space. In that case, the standard trick to deal with categorical features is to encode them as continuous variables.
The transformed dataset can then be fed to any usual classification algorithm.

The most common approach is to use the disjunctive encoding. A variable with $C$ possible categories is encoded as $C$ continuous features. Each artificial feature represents the membership to a category. Therefore, only one of those variables takes value 1 for a given sample. The other ones are set to 0.

That being said, many linear classifiers also exist in a variant with kernels. Similarly to the non-linear SVM (see Section 2.3.2), this allows the use of many different data types e.g. graphs, strings, categorical variables, etc. For example, the kernelised versions of LDA [MRW+99], perceptron [ABR64] and logistic regression [GY85] make those approaches compliant with the classification of heterogeneous data without relying on a data recoding.
Chapter 3

Feature selection

Today’s very high-dimensional datasets require specific approaches in order to mine some knowledge hidden in the data. If efficient classifiers can be learned from those kinds of data, the high number of variables is an obstacle to a good comprehension of those predictive models. We are just lost in very big spaces.

Feature selection is a powerful ally in the quest towards interpretability. It reduces the number of variables to analyse, guiding experts towards a better understanding of the problem of interest. Even if many feature selection methods deal with either continuous or categorical datasets, few tackle the problem of heterogeneous data directly.

This chapter presents feature selection with a particular focus on methods that can deal with continuous and categorical variables. Section 3.1 briefly pictures the domain of dimensionality reduction. It defines what feature selection is and explains the difference with feature extraction. After that, Section 3.2 explains the three classes of feature selection methods, namely filters, wrappers and embedded approaches. Then, Section 3.3 focuses on feature selection from heterogeneous data. Finally, Section 3.4 presents two procedures that turn feature rankings into a statistically interpretable feature importance measure. They allow to clearly distinguish between relevant and irrelevant variables.

3.1 Dimensionality reduction

Dimensionality reduction consists in decreasing the number of variables for a given problem. As we focus here on classification problems, suppose we have an original dataset $X^{n \times p}$ and a vector of labels $y^n$. Instead of learning a predictive model on $X$, we would like to do so on a transformed
Chapter 3. Feature selection

dataset $X'$ of size $n \times s$ where the number of features $s$ is smaller than $p$. Each of the $n$ samples are thus kept but they are projected in a new space of $s < p$ dimensions. There are two major approaches to perform dimensionality reduction: feature extraction and feature selection.

The first one creates $s$ fresh variables by combining the $p$ original ones. Popular feature extraction approaches include principal components analysis [Pea01], auto-encoders [Ben09] and several non-linear dimensionality reduction methods [Kra91, LV07]. Those sophisticated techniques can drastically reduce the number of dimensions and allow to learn good predictive models from a very small number of variables. However they miss our main objective which is interpretability. It is indeed very difficult to find out which of the $p$ original variables are important. In addition, most of those methods are unsupervised in the sense that they make no use of the label vector $y$. They do not optimise class separability and are thus better suited for data visualisation than for classification. A few alternatives, like partial least squares discriminant analysis [BR03] or linear discriminant analysis [Fis38], project data along new dimensions that explain the response $y$.

Feature selection adopts another strategy. It keeps original features but focuses on the most important ones. The $s$ columns of $X'$ are a subset of the $p$ columns of $X$. As stressed before, this provides interpretability by focusing on a few informative variables. In a biomedical context, it can also help to develop cheap practical diagnosis kits if good predictive models can be build from a few biological and/or clinical factors.

Contrarily to feature extraction, most feature selection methods try to optimise class separability. However, reducing the dimensionality is often a trade-off between the number of variables and the predictive performances. Indeed, less informative features may still convey some useful signal for classification purposes. Removing them is a step towards interpretability but it may negatively impact the prediction.

3.2 Feature selection paradigms

There are three main types of feature selection techniques: filters, wrappers and embedded methods. These patterns vary in the way and where they perform feature selection in the global process of learning a predictive model. This section reviews the three different approaches.
3.2. Feature selection paradigms

3.2.1 Filters

Feature filtering is the most simple feature selection scheme. It is a standalone data preprocessing step that takes place before the induction of a predictive model. Algorithm 3.1 summarises this framework. The Filter function takes as input the training data with their respective class labels. It outputs a relevance score for each variable with respect to the task of making a distinction among classes.

---

**Algorithm 3.1: Filter pattern**

**Input:** $S$: set of instances $(x_i, y_i)$ where the $x_i$ are $p$-dimensional

**Output:** a ranking of the $p$ features

\[
\text{importances} \leftarrow \text{Filter}(S)
\]

\[
R \leftarrow \text{sortVariablesPer(importsences)} \quad \text{// variable ranking}
\]

**return** $R$

---

Filters can be either univariate or multivariate. Univariate methods assess the relevance of each feature separately, without considering the possible interactions with other variables in the dataset. The great advantage is in the computing time. Such approaches generally run in $O(p)$. A typical example of a univariate filter is the Welch’s $t$-test [Wel47] that tests if the mean value of a variable differs among the various classes. Multivariate methods can have a much higher computational complexity since they may consider groups of variables together. To make that task tractable, one has to resort on heuristic searches like minimum redundancy – maximum relevance [DP05] or minimum interaction – maximum relevance [BM10].

3.2.2 Wrappers

Wrapper methods look for good feature subsets by successively evaluating the quality of candidate variable sets. In order to assess the quality of a signature, they use the feedback of a predictive model. The feature selection is somehow ‘wrapped around’ the classifier. Wrappers can follow many different search strategies e.g. sequential search, genetic algorithms, simulated annealing (see [Guy06]). We focus here on two families of wrapper methods based on sequential searches: those which perform forward selection and those which select features backward. As illustrated in Algorithm 3.2, forward approaches start from an empty feature set and build a ranking by successively adding the most important variable to the previously selected ones. Backward methods build a feature ranking the other way around. Starting from all features, it
iteratively removes the most useless variables. The notion of feature importance can be assessed in various ways such as the gain in predictive performances [Guy06] or the change in class vote probabilities [DA12] when adding a specific variable. A pseudo-code is given in Algorithm 3.3.

Algorithm 3.2: Wrapper pattern : forward selection

| Input: S: set of instances \((x_i, y_i)\) where the \(x_i\) are \(p\)-dimensional |
| Input: \(L\): a learning algorithm |
| Output: a ranking of the \(p\) features |

\[
\begin{align*}
F & \leftarrow \text{set of all } p \text{ features in } S \\
R & \leftarrow \text{empty variable ranking} \quad \text{// ordered set of variables} \\
\text{while } F \text{ is not empty do} \\
& \quad \text{for } x_j \in F \text{ do} \\
& \quad \quad \text{Assess the predictive performances of } L \text{ on } S \\
& \quad \quad \text{using variables in } R \cup \{x_j\} \\
& \quad \quad \text{end} \\
& \quad x_{\text{best}} \leftarrow \text{variable } x_j \text{ that yields best predictive performances} \\
& \quad F = F \setminus \{x_{\text{best}}\} \\
& \quad R = \text{concatenate}(R, x_{\text{best}}) \quad \text{// add best variable} \\
& \quad \text{end} \\
\text{return } R
\end{align*}
\]

Forward methods have a computational advantage over the other ones. At the first iteration they build \(p\) predictive models containing one variable and at the last iteration, they build one model from \(p\) features. For backward wrappers, it is much worse. They first build \(p\) models of \(p - 1\) variables and end up with one model with one variable at the end of the process. This increased computational time is the price to pay for some benefits though. Backward approaches tend to miss less important features than forward ones. It is because they consider more interactions between variables at the beginning of the procedure. In order to keep this benefit while decreasing the computational time, some variants of backwards methods remove more than one feature at each iteration [GWBV02].

3.2.3 Embedded methods

Like wrappers, embedded methods also use a classifier to select variables. However, while wrappers use predictive models as black boxes to get some feedback, embedded methods extract feature importance from the
3.2. Feature selection paradigms

**Algorithm 3.3:** Wrapper pattern: backward selection

- **Input:** $S$: set of instances $(x_i, y_i)$ where the $x_i$ are $p$-dimensional
- **Input:** $L$: a learning algorithm
- **Output:** a ranking of the $p$ features

1. $F \leftarrow$ set of all $p$ features in $S$
2. $R \leftarrow$ empty variable ranking // ordered set of variables
3. while $F$ is not empty do
   4. for $x_j \in F$ do
      5. Assess the predictive performances of $L$ on $S$
         using variables in $F \setminus \{x_j\}$
   6. end
   7. $x_{toRemove} \leftarrow$ variable $x_j$ with smaller decrease in performance
   8. $F = F \setminus \{x_{toRemove}\}$
   9. $R = concatenate(x_{toRemove}, R)$ // add removed variable
      // at the top of the ranking
10. end
11. return $R$

very structure of classifiers. It can be because the induction algorithm promotes sparsity of the features in the model, or because we can get a sense of how the variables are useful in the prediction by analysing the model. In that sense, the importance of the variables is a by-product of the learning algorithm. This is shown in Algorithm 3.4.

**Algorithm 3.4:** Embedded pattern

- **Input:** $S$: set of instances $(x_i, y_i)$ where the $x_i$ are $p$-dimensional
- **Input:** $L$: a learning algorithm
- **Output:** a predictive model and the importance of each variable

1. $model \leftarrow$ learn a model with $L$ on training set $S$
2. return $(model.predictiveModel, \quad // to perform classification$
   $model.variableImportances)$

Even though embedded methods output both a classifier and the importance of variables, it was shown that predictive performances may benefit from a two-stage procedure [DMMTV09]. The idea is to first apply an embedded method to obtain a feature ranking or directly a subset of useful variables if the model is sparse. After that, another (non-sparse) classifier is induced from a reduced set of features. We mainly use that filter-like approach in this work.
Chapter 3. Feature selection

3.3 Heterogeneous feature selection

Even if datasets with different types of variables are very common, it seems that heterogeneous feature selection is still overlooked nowadays. We present here some feature selection methods that handle both continuous and categorical data. Section 3.3.1 pictures some naive approaches that transform heterogeneous datasets into fully continuous or categorical data. Section 3.3.2 explains how to perform heterogeneous feature selection by plugging adequate classifiers into the wrapper pattern. We then detail the state-of-the-art methods to which we compare to in this thesis. The first one, in Section 3.3.3, is an embedded feature importance index extracted from random forest. The second one is a wrapper-like forward selection method described in Section 3.3.4. Finally, Section 3.3.5 presents the Recursive Feature Elimination mechanism. While it is initially tailored for continuous data, we adapt it for heterogeneous features in this work.

3.3.1 Naive approach

To highlight important variables, one could transform heterogeneous data into either fully continuous or categorical variables before applying any standard feature selection algorithm.

For instance, categorical variables can be encoded as numerical values to get a continuous dataset. The specific choice of such numerical values is however arbitrary. It introduces an artificial order between the feature values and can lead to largely different distance measures between instances [DV11]. For example, in the context of allergy prediction, datasets may contain a variable that encodes the kind of pet a child is in contact with. Coding its values dog, cat, rat as 1, 2 and 3 would somehow mean that a cat is half-way from being a dog and a rat. Not only does it make no sense since categories encode conceptually different properties, but this order is totally arbitrary. Another choice leads to a totally different interpretation.

Another approach relies on a multivariate numerical encoding, such as the disjunctive encoding, to represent categorical variables. For instance, a feature with 3 categories as possible values could be encoded by considering 3 new numerical features instead: (1, 0, 0), (0, 1, 0) and (0, 0, 1). This method is very standard. Its major advantage is that one only need to plug the transformed dataset into any classical feature selection method. However, such an encoding is essentially a concatenation of binary indicator variables. Seeing those as numerical features is arguable since any intermediate value strictly between 0 and 1 has
3.3. Heterogeneous feature selection

actually no meaning. In addition, the new artificial features could be assigned very distinct ranks by a feature ranking procedure. Some of those variables may eventually be selected while others not and the interpretation of the original feature would then be completely lost. Specific approaches, such as group lasso [YL06], could be used to ensure feature selection at the granularity of the original variables. They either select all numerical features that encode a categorical variable or none of them. Yet, those artificial features may still be ranked very differently inside the selection.

One can also consider doing the other way around: transforming continuous variables into categorical ones. To do so, continuous features are split into intervals that are recoded as categories. This process, known as binning, can however be very sensitive to the specific discretisation [Hal99] i.e. the number of bins and the way they are defined. Another problem of this method is that we lose much of the original signal. Binning somehow reduces the resolution of the measurements. In addition, the notion of order in continuous variables is also lost by design of the categorical features.

3.3.2 Wrapping heterogeneous classifiers

The wrapper pattern also offers ways to deal with heterogeneous feature selection. Indeed, many predictive models can deal with continuous and categorical variables, either directly or by means of a feature recoding. For instance, they can be used as black-boxes in one of the forward or backward selection schemes (cf. Algorithms 3.2 and 3.3) to evaluate the predictive performances of various feature sets. The potential use of a disjunctive encoding for the categorical variables is not problematic in the wrapper pattern. Indeed, for a particular categorical feature $x_j$ in the original encoding, one either considers all the corresponding numerical features (when including $x_j$) or none of them (when excluding $x_j$) to learn a predictive model. The classification performances we obtain as a feedback of the quality of the different feature sets are thus consistent with the original variables.

Yet, wrapper methods can present some disadvantages when dealing with high-dimensional data with few samples. For instance, the estimation of the predictive performances of each feature set might be problematic. This is generally done through a procedure similar to cross-validation [KS95, Koh95] which repeatedly learns models on a fraction of the data to evaluate the predictive performances on the other, previously unseen, samples. If we take into account that the whole feature selection procedure is itself evaluated in an outer validation loop, the
already quite limited number of samples could become very small, twice reduced by the nested evaluation procedures. In those conditions, the predictive performances computed inside the wrapper may vary a lot. Another problem is that the computing time might become quite important because the number of models to learn in the sequential search directly depends on the number $p$ of dimensions.

### 3.3.3 Variable importance from Random Forest

As explained in Section 2.2.2, a RF model [Bre01] is made of an ensemble of trees, each of which is grown from a bootstrap sample of the $n$ data points. For each tree, the selected samples form the bag (denoted by $B$), the remaining samples form the out-of-bag (OOB) denoted by $\overline{B}$. Thereby, the number of bags is equal to the number of OOBs which is equal to the number $T$ of trees in the forest.

In order to compute feature importance, Breiman [Bre01] proposes a permutation test procedure based on classification error. It falls into the framework of embedded methods, described in Section 3.2.3. For each variable $x_j$, there is one permutation test per tree in the forest. For an OOB sample $\overline{B}_k$ corresponding to the $k$-th tree of the ensemble, one considers the original values of the variable $x_j$ and a random permutation $\tilde{x}_j$ of its values on $\overline{B}_k$. The difference in prediction error using the permuted and original variable is recorded and averaged over all the OOBs in the forest. The higher this index, the more important the variable is assumed because it corresponds to a stronger increase of the classification error when permuting it. Because decision trees make predictions according to the splitting variables chosen in each node, Breiman’s importance measure is multivariate. It compares the predictive performances of the decision trees when feature $x_j$ is jointly used with other variables to classify samples to the performances when the class information in $x_j$ is cancelled out by the permutation.

The importance measure $J_a$ of the variable $x_j$ is precisely defined as:

$$J_a(x_j) = \frac{1}{T} \sum_{k=1}^{T} \frac{1}{|\overline{B}_k|} \left( \sum_{i \in \overline{B}_k} I(h_k^\tilde{x}_j(i) \neq y_i) - I(h_k(i) \neq y_i) \right),$$  

where $y_i$ is the true class label of the OOB example $i$, $I$ is an indicator function, $h_k(i)$ is the class label of the example $i$ as predicted by the tree estimated on the bag $B_k$, $h_k^\tilde{x}_j(i)$ is the predicted class label from the same tree while the values of the variable $x_j$ have been permuted on $\overline{B}_k$. Such a permutation does not change the tree but potentially changes the prediction on the out-of-bag examples since its $j$-th dimension is
modified after the permutation. Since the predictors with the original variable \( h_k \) and the permuted variable \( \tilde{h}_k \) are individual decision trees, the sum over the various trees where this variable is present captures the ensemble behaviour, respectively from the original variable values and its various permutations. Whenever a specific variable does not appear in a tree, the prediction cannot be affected by permuting its value, which means that the specific term corresponding to this tree in Equation (3.1) is null.

While the error rate (proportion of misclassified examples) falls between 0 and 1, \( J_a \) takes values in a much smaller interval around 0. This is mainly explained by two facts. Firstly, a particular variable \( x_j \) is not expected to appear in all decision trees of the random forest. However, as we can see in Equation 3.1, \( J_a \) is normalised by the number \( T \) of trees. Since we measure the importance of variables in the global decision model, it is quite normal to penalise variables that are rarely used in the individual decision trees. Secondly, because the base predictive models are multivariate, permuting only one variable \( x_j \) at a time is not likely to perturb much their predictive performances, except for those decision trees with \( x_j \) at (or near) the root node. All in all, \( J_a \) is very convenient to rank variables according to their respective importance in a particular random forest but it is hardly interpretable on its own. Figure 3.1 shows an example of such a feature ranking with the \( J_a \) values of the most important variables. As we can see, the most important variable only decrease the average accuracy by 0.8% which is quite counterintuitive.

As discussed in Section 2.2.3, the induction of decision trees follows a greedy forward strategy. In a given node, the variable with the biggest univariate effect is selected for splitting. Therefore, the importance \( J_a \) does not reflect the quantity of class information in the features but rather how important they are to perform classification with the given RF model. Indeed some informative variables may appear totally irrelevant with respect to \( J_a \) because they are somehow masked by features with a stronger marginal effect [LWSG13] which appear higher in the trees.

### 3.3.4 Hybrid feature selection

Hybrid feature selection (HFS) is a wrapper-like approach introduced in [DV11]. It performs a greedy forward selection aggregating separate rankings for each type of variables into a global ranking.

HFS first builds two separate rankings for continuous and categorical features. In the original paper, those rankings are computed with
Chapter 3. Feature selection

Figure 3.1: Example of variable importance in a random forest. The $J_a$ index is reported for the 50 best variables.

the Mutual information (MI) multivariate filter. However, a reliable estimate of MI is difficult to obtain whenever fewer samples than dimensions are available. As this work targets high-dimensional data where $n \ll p$, one could instead use the p-values of a t-test to rank continuous features and of a Fisher exact test for categorical ones.

The two feature rankings are then combined into a global ranking by iteratively adding the first categorical or continuous variable that maximizes the predictive performance of a classifier. Algorithm 3.5 gives a pseudo-code for this procedure. As we can see, this method is more computationally efficient than the wrapper framework described in Section 3.2.2. Indeed, only two predictive models have to be learned in each iteration of the loop.

To assess the quality of the candidate features, the authors of [DV11] suggest using a naive Bayes or a 5-NN classifier (cf. Section 2.5). Their performances are indeed very sensitive to irrelevant variables which might help the wrapper procedure to choose the best feature to add in each iteration.

The combination of the HFS procedure with a naive Bayes or a 5-NN classifier are referred to as $HFS^{NB}$ and $HFS^{5NN}$ in the remainder of this document.
3.3. Heterogeneous feature selection

Algorithm 3.5: Hybrid feature selection

Input: $S$: set of instances $(x_i, y_i)$ where the $x_i$ are $p$-dimensional.
Input: $L$: a learning algorithm.
Output: a ranking of the $p$ features.

$R \leftarrow$ empty ranking // global feature ranking
$R_{cont} \leftarrow$ ranking of continuous variables w.r.t. their importance
$R_{cat} \leftarrow$ ranking of categorical variables w.r.t. their importance.

while $R_{cont}$ is not empty or $R_{cat}$ is not empty do

$x_{cont} \leftarrow$ first variable of $R_{cont}$
Assess the predictive performances of $L$ on $S$
using variables in $R \cup \{x_{cont}\}$

$x_{cat} \leftarrow$ first variable of $R_{cat}$
Assess the predictive performances of $L$ on $S$
using variables in $R \cup \{x_{cat}\}$

if $x_{cont}$ is better then

$R \leftarrow$ concatenate($R, x_{cont}$)
$R_{cont} \leftarrow R_{cont} \setminus x_{cont}$

else

$R \leftarrow$ concatenate($R, x_{cat}$)
$R_{cat} \leftarrow R_{cat} \setminus x_{cat}$

end

end

return $R$
3.3.5 Recursive feature elimination

In [GWBV02], Isabelle Guyon et al. propose a procedure to perform gene selection from linear SVMs. While their method was initially developed to deal with continuous features only (because of the linear kernel), we adapt it in an heterogeneous context in this thesis (cf. Chapter 7).

When using a linear SVM (see Section 2.3.1), one can look at the weight of each feature in the separating hyperplane to have an idea of their importance in the discriminant function. Yet, directly ranking features according to their respective weights might not be a good idea. In particular, selecting a small subset of the most weighted features could lead to very poor results in terms of predictive performances.

Recursive Feature Elimination (RFE) [GWBV02] improves this variable selection. It builds a global feature ranking by iteratively removing the least important variable of a linear SVM. Indeed, each feature weight only reflects the effect of one variable on the decision function. Removing them one by one and re-estimating a linear discriminant in each iteration allow to better identify the most important variables. This is because feature weights are re-computed while least important features are actually removed. The process is repeated until all variables are ranked.

Following [AHVdP+10], one can speed up the process by dropping a fixed proportion (e.g., 20%) of features at each iteration. The benefit of such a fixed proportion is that the actual number of variables removed at each step gradually decreases till be rounded to 1. This leads to a finer ranking for the most important features. It is particularly convenient when dealing with high-dimensional biomedical data where a lot of variables are supposed to be irrelevant. Algorithm 3.6 presents a pseudo-code for such a procedure.

---

**Algorithm 3.6: Recursive Feature Elimination**

**Input:** $S$: set of instances $(x_i, y_i)$ where the $x_i$ are $p$-dimensional  

**Output:** a ranking of the $p$ features

$R \leftarrow$ empty ranking  

$F \leftarrow$ set of all features  

while $F$ is not empty do  

train a linear SVM classifier using $F$  

remove the least important features from $F$  

put those features on top of $R$  

end  

return $R$
3.4 Extracting statistically significant features from rankings

While most feature selection methods rank features according to their relative importance, it is often very difficult to decide which variables are actually important with respect to a given prediction task. In other words, the choice of the number $s$ of most important variables that should eventually be kept is still challenging. It is usually assessed through a proper validation protocol.

To overcome this problem, the authors of [HTSWG12] compare several ways to obtain a statistically interpretable index from a feature relevance score. Their goal is to convert feature rankings to statistical measures such as the false discovery rate, the family wise error rate or $p$-values. Their proposed methods typically make use of a permutation procedure to compute some null distribution from which those metrics are estimated. The permutation tests repeatedly compute feature rankings on dataset variations, e.g. for which some features are randomly permuted.

Those procedures depend on at least two important meta-parameters: $N$, the number of permutations to compute the null-distribution and $J$, the feature importance index according to which features are ranked in each iteration.

Among the various methods presented in [HTSWG12], two techniques specifically estimate $p$-values that features are important in a prediction task. They are presented in the remainder of this section.

3.4.1 mr-Test

The mr-Test [ZLZ06] repetitively samples $\frac{n}{2}$ examples out of $n$ without replacement. It also assumes that a prescribed fraction (by default, $\frac{p}{2}$ out of $p$) of variables are irrelevant. Out of $N$ resamplings ($N \geq 100$ is typically chosen), the null distribution is defined as the rank distribution of the worst $\frac{p}{2}$ variables according to their average importance values (computed with $J$). For each remaining variable, its average rank over the $N$ resamplings is compared to the null distribution, which defines its associated $p$-value. Because the significance of several features are assessed on the same data, $p$-values are corrected for multiple testing using the Benjamini-Hochberg procedure [BH95].
3.4.2 1Probe

The 1Probe ranks $N$ times all the features of the dataset (the $n$ observations) after introducing an additional non-informative feature randomly sampled from $N(0, 1)$ in each iteration. The $p$-value of a feature is then estimated as the proportion of iterations for which the non-informative variable has a better rank according to $J$. Similarly to mr-Test, the final $p$-values are also corrected for multiple testing.
Chapter 4

Experimental setting

This chapter presents the metrics and experimental protocol used in this thesis. As we focus on feature selection methods, there are two aspects that need to be assessed (cf. Section 1.4). Firstly, the classification performances that can be obtained from the selected variables. Secondly, the stability of the selected variables i.e. how much the set of selected features varies when small changes appear in the dataset.

When assessing the performances of a predictive model, one is interested in the generalisation capabilities of the classifier. Therefore, we must be very careful to measure performances when predicting on data points that were not considered when learning the model. The contrary would result in a very optimistic evaluation of the performances. It is indeed much easier to predict the class of a sample whose label was observed during the model induction.

The same kind of optimistic bias exists for feature selection. The set of selected variables is likely to yield better predictive performances if the selection also considers the data used to assess those performances. This is known as the selection bias [AM02].

In this chapter, we present the various performance metrics as well as an experimental protocol that avoids optimistic biases. The metrics that assess the quality of prediction and feature selection are presented in Sections 4.1 and 4.2. The experimental protocol is presented in Section 4.3.

4.1 Predictive performances

When reporting predictive performances, one often resort to the proportion of correctly classified samples, or accuracy. Despite the popularity
of this index, it might be very misleading when the class distribution of the data is very imbalanced. For instance, if the natural class proportions of a hypothetical classification problem are 90\% for class $c_1$ and 10\% for class $c_2$, a trivial predictor that always output the $c_1$ label already obtains a very good accuracy: the prediction is correct 90\% of the time.

The Balanced Classification Rate (BCR) avoids this problem by giving the same importance to each class in the performance evaluation. It is defined as the mean of the classification accuracy in each class. For a two-class problem, BCR is defined as the average between sensitivity and specificity:

$$BCR = \frac{1}{2} \left( \frac{TP}{P} + \frac{TN}{N} \right),$$  \hspace{2cm} (4.1)

where $TP$ (resp. $TN$) is the number of correct predictions of the positive (resp. negative) class and $P$ (resp. $N$) is the total number of positive (resp. negative) samples. Its multi-class generalization takes the following form:

$$BCR = \frac{1}{c} \sum_{l=1}^{c} \frac{TC_l}{C_l},$$  \hspace{2cm} (4.2)

where $c$ is the number of classes, $TC_l$ is the number of correct predictions of class $l$ and $C_l$ is the total number of samples of class $l$. This metric has been used, for instance, in the performance prediction challenge\(^1\) held at WCCI 2006 precisely to deal with possible class imbalance while considering the calibration of specific models [GADB06].

### 4.2 Feature selection stability

Stability of feature selection indices quantifies how selected sets of features vary after small perturbations of the datasets. In this thesis, we report the Kuncheva index (KI) [Kun07]. This index is particularly convenient when one compares signatures of the same number of selected features. In addition, it has the interesting property to include a correction for random selection.

KI specifically measures to which extent $K$ sets of $s$ selected features share common elements.

$$KI(\{F_1, \ldots, F_K\}) = \frac{2}{K(K-1)} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \frac{|F_i \cap F_j| - \frac{s^2}{p}}{s - \frac{s^2}{p}},$$  \hspace{2cm} (4.3)

\(^1\)The evaluation metric in this challenge actually relied on $BER$, the balanced error rate, which conveys the same information since $BCR = 100\% - BER$. 

where \( p \) is the total number of features and \( \frac{s^2}{p} \) is a term correcting the random chance, for 2 feature sets \( F_i \) and \( F_j \), to share common features. The larger its value, the larger the number of commonly selected features.

This index ranges within \((-1, 1]\). A value of 0 is the expected stability for a selection performed uniformly at random. KI reaches the maximal value of 1 when the sets \( F_1, \ldots, F_K \) of selected features are all identical. Values below 0 would mean that the selection is less stable than when repeatedly sampling uniformly at random \( s \) features among \( p \), i.e. the various features sets would share less elements than with random selection.

The correction with respect to the random chance of selecting common features is particularly useful when comparing large sets of variables. Without the \( \frac{s^2}{p} \) term, the value of the stability index would artificially increase towards 1 with the number \( s \) of selected features. KI ensures that feature selection has to be better than random to yield a positive value.

One can also choose some alternative stability index. For instance, Jaccard’s index [Jac12] also measures how much \( K \) signature sets have common elements. It is quite similar to KI and is well defined for sets of different sizes. However, it lacks a correction for the random chance of selecting common features. Haibe-Kains’ index [HK09] has another approach. It defines the stability of \( K \) sets of \( s \) features as the average frequency of the \( s \) most frequent features. Those indices are presented and compared to KI in Chapter D. We show that they behave very similarly. Hence, we only report KI in the experiments.

4.3 Experimental protocol

In order to avoid several pitfalls mentioned in the introduction of this chapter, the datasets used to perform the learning steps and the assessment must be different and non-overlapping. When a sufficient amount of data is available, the usual approach is to perform 10-fold cross validation (10-CV) that provides a reliable estimate of model performances [Koh95]. It consists in splitting the dataset into 10 partitions of equal size. Each in turn, one partition is removed from the dataset and kept apart. The classification algorithm is trained on 90% of the data, the training set. Its predictive performances can then be assessed on the remaining 10% data, the test set. The 10-CV procedure repeats the process 10 times, one for each partition as test set. It outputs the 10 measures of predictive performances. One can typically compute the
average performance as well as the variance of the results. It is described in Algorithm 4.1.

**Algorithm 4.1: 10-CV**

**Input**: $L$ : a learning algorithm  
**Input**: a dataset  
**Output**: the 10 measures of predictive performances

Partition data into 10 disjoint subsets of equal size

for $i ← 1$ to 10 do

    $S_{te} ← i$-th subset  // test set : pairs $(x_i, y_i)$
    $S_{tr} ←$ remaining data  // training set : pairs $(x_i, y_i)$
    $m ←$ train $L$ on $S_{tr}$  // predictive model
    $pred ←$ predict labels of $S_{te}$ with $m$  // vector of predictions
    // true labels in $S_{te}$ not used
    compute predictive performances from $pred$ and labels in $S_{te}$

end

Most biomedical datasets are quite small in term of number of samples. In such conditions, 10-CV may lead to inaccurate estimates due to a higher variability in the different folds. We thus make use of a resampling strategy that allows us to build more predictive models and obtain more reliable performance estimates. Instead of partitioning the dataset, we randomly sample without replacement 90% data as training set $S_{tr}$ and use the 10% remaining as test set $S_{te}$. We typically repeat this process 200 times in our experiments. Such a protocol has the same training/test proportions as 10-CV but benefits from a larger number of tests. It also keeps the training size sufficiently large so as to report performances close enough to those of a model estimated on the whole available data. However, the various test sets are no longer independent from each other. This is problematic when comparing methods or when computing confidence intervals with statistical approaches that assume independence between the measurements. Fortunately, some alternatives such as Nadeau’s rescaled statistic [NB03] take this into account.

Before feeding a learning algorithm with data, it is often necessary to perform some pre-processing steps. Firstly, all features may not be available for all samples in a dataset. We handle those missing values by a quite simple fix, for each training and test sets. Each missing value of a continuous variable is replaced by the mean feature value, computed on the training set. For categorical variables, missing values are replaced by the most frequent category of the training set. Even if this scheme is quite simple, we note that the impact of imputing bad values is quite limited. Indeed, feature selection would discard those features which are
4.3. Experimental protocol

corroded by an inadequate imputation of the missing values. Secondly, many classifiers, like the linear SVM (cf. Section 2.3.1) and a Euclidean norm nearest-neighbour (cf. Section 2.5), require each feature to be normalised. A standard way to do it is to center them and rescale to unit variance. To avoid what is called a normalisation bias, the normalisation parameters are computed on the training set only. Finally, when dealing with very high-dimensional biomedical data, it is very frequent to pre-filter several variables out to decrease the computational time. One typically drops features that have the lowest standard deviations before normalisation. The underlying assumption is that variables that do not change a lot are not likely to encode any biological signal and are mostly made of background noise. Indeed, features that do not vary on their own will not vary when conditioned to other variables. We should note that we can only compare the variances of features of the same nature e.g. gene expression levels. In this work, we filter 75% of genomic features out when dealing with such datasets. While this may seem quite severe, it typically keeps a few thousands of variables.

Algorithm 4.2 gives a pseudo-code for a 200-resamplings procedure. It also shows how pre-processing steps, feature selection and model induction avoid optimistic biases by considering only the training instances to compute their various parameters. For each signature size, the procedure returns the 200 measures of predictive performances as well as the 200 feature sets used to learn the models. One can then compute the mean and variance of the predictive performances and the feature selection stability.
Algorithm 4.2: 200-resamplings

Input: $L$: a learning algorithm

Input: $prefilter$: a Boolean indicating if pre-filtering is to be performed

Input: rankFeatures: a feature ranking method

Input: The sizes of the desired signatures

Input: A dataset

Output: All the measured predictive performances and the feature sets on which models were learned

for $i \leftarrow 1$ to 200 do

$S_{tr} \leftarrow$ uniformly sample 90\% data without replacement
$S_{te} \leftarrow$ remaining 10\% data

// imputation of missing values
Fix missing values of $S_{tr}$
Fix missing values of $S_{te}$ with parameters of $S_{tr}$

// normalisation of continuous features
compute mean and standard deviation of all features in $S_{tr}$
normalise features of $S_{tr}$ to 0 mean and unit variance
normalise features of $S_{te}$ with parameters computed on $S_{tr}$

// pre-filtering
if $prefilter$ then
    drop 75\% of features with lowest standard deviation
    // s.d. computed before normalisation
end

$R \leftarrow$ rankFeatures($S_{tr}$) // feature ranking

for each signature size $s$ do

// Feature selection
restrict $S_{tr}$ and $S_{te}$ to the top $s$ features of $R$

// Model induction and assessment of the performances
$m \leftarrow$ train $L$ on $S_{tr}$ // predictive model
$pred \leftarrow$ predict labels of $S_{te}$ with $m$ // vector of predictions
// true labels in $S_{te}$ not used
compute predictive perf. from $pred$ and labels in $S_{te}$
record set of $s$ selected features

end

end
Part II

More interpretable heterogeneous feature selection methods
This part of the document summarises our scientific contributions. While the full papers are annexed, the three following chapters explain the developed methods and give a brief overview of the results.

Chapter 5 outlines an analysis of tree ensemble methods stabilities. Chapter 6 presents a statistically interpretable feature importance measure for Random Forest. Chapter 7 introduces two kernel-based heterogeneous feature selection methods. Finally, Chapter 8 presents the general conclusion and perspectives about our contributions.

In order to tease the reader, the take-home message of each contribution is summarised in one sentence at the beginning of each chapter.
Chapter 5

Stable variable rankings from Random Forest

Breiman’s importance index requires 100 times more trees than needed to reach the predictive performances plateau to provide stable feature selection with genomic data.

Tree ensemble methods (see Section 2.2) generally reach state-of-the-art predictive performances. They also handle continuous and categorical variables by design and provide ways to rank features according to their importance in the prediction of unseen samples (see Section 3.3.3). It looks like tree ensembles are perfect candidates to perform heterogeneous feature selection. However, the variable rankings extracted from methods such as Random Forest (RF) [Bre01] suffer from high instability. This is a consequence of the randomness used in the forest growing process in order to promote diversity among the various trees (see Section 2.2). The very process of learning decision trees includes a greedy selection of a most discriminant feature at each node, according to a relevance index. Tree ensemble methods choose this variable from a random subset of the features. While most tree ensemble approaches focus on improving predictive performances, our first contribution is to analyse them from a stability point of view.

This is the object of a paper presented at the ESANN conference in 2012 [PVD12]. This paper performs an empirical analysis of the convergence of the stabilities of a representative set of tree ensemble methods on four high-dimensional biomedical datasets (3 genomic datasets with several thousands of variables, and one ECG dataset with more than 250
Chapter 5. Stable variable rankings from Random Forest

features). We report here a summary of the results. The full analysis is available in the paper annexed in Chapter A.

The generalization capabilities of RF, and similar tree ensemble classifiers, is known to increase up to a certain point with the number of trees in the forest. Section 5.1 shows that only a few trees are needed to reach a plateau of the predictive performances. Section 5.2 assesses to which extent the predictive performance convergence appears earlier than a stable class prediction. Section 5.3 presents the main contribution of our paper. It analyses the stability of feature selection with respect to the ensemble size and shows that the asymptotic stability can be reached at the cost of growing even more trees than for stable class prediction. Finally, Section 5.4 summarises the results.

5.1 Predictive performances

It is a well-known fact that ensemble methods can increase their predictive performances by growing more base classifiers, until they reach a threshold beyond which no further improvement is observed [SFBL98, HLMMS13, MMHLS09]. Intuitively, it is better to seek the advice of several experts than asking only a few ones. When growing tree ensembles, such as RF, one typically tunes the number of trees to the smallest possible ensemble size while still reaching the asymptotic predictive performance.

We experimentally confirm this fact by growing tree ensembles of different sizes on 200 resamplings of four biomedical datasets. We show that the best predictive performances are generally quickly reached, even with very high-dimensional datasets. The convergence is typically obtained after 10 or 20 trees are grown. Figure 5.1 pictures a typical graph of the evolution of the Balanced Classification Rate (cf. Section 4.1) with respect to the number of trees in a RF. It is computed on the DLBCL genomic dataset (see Chapter F, annexed). Similar results on more datasets with different tree ensemble approaches are available in [PVD12].

5.2 Stability of class prediction

We confirmed that the number of trees is important from a classification point of view. A distinct but related issue is the stability of the class prediction, that is to which extent the class label predicted for each test example stays the same over different data resamplings. As the bootstrap mechanism at the core of the estimation of ensemble classifiers
5.2. Stability of class prediction

is known to reduce variance in most cases, stabilizing the individual class prediction is expected but possibly with a larger number of trees. Indeed, the predictive performances may be stable while the prediction of individual samples still varies.

The stability of the class prediction measures to which extent each individual test example is assigned the same class label across various resamplings. For each example $\mathbf{x}_i$, let $c^*$ denote the most commonly predicted class label (across all resamplings for which that example appears in a test fold); let $n_{\mathbf{x}_i}^{c^*}$ be the number of times such a prediction occurs out of the $n_{\mathbf{x}_i}$ occurrences of $\mathbf{x}_i$ in a test fold. The class prediction stability is given by:

$$\frac{1}{n} \sum_{i=1}^{n} \frac{n_{\mathbf{x}_i}^{c^*}}{n_{\mathbf{x}_i}},$$

where $n$ denotes the total number of examples, each of them appearing approximately $0.10 \times 200 = 20$ times in a test fold. Such a stability index falls in the interval $[\frac{1}{|C|}, 1]$ with $|C|$ classes. The stability is equal to 1 when each test example is always assigned the same, although not necessarily correct, class label.

Experimental results show that class prediction stability appears later than predictive performances stability. It typically requires 10 times more trees. Indeed, even though the predictive performances are

Figure 5.1: Average predictive performances of RF with respect to the number of trees, computed on 200 resamplings of the DLBCL dataset.
already at their best, individual sample predictions keep changing. In addition, the proportion of label switch among correctly classified and misclassified instances is the same on average because the predictive performances are already stabilised. The class prediction stability curve of RF on the DLBCL dataset is shown in Figure 5.2.

5.3 Stability of feature selection

The diversity among the base classifiers is the key to the good predictive performances of tree ensemble methods [Bre96, SK96]. As explained in Section 2.2, ensemble classifiers such as RF make use of a randomised growing scheme. The goal is to build a committee of very different decision trees that model alternative points of view of the data and then, to make consensus predictions. However, that very diversity among base learners is at the root of the feature instability problem of tree ensemble methods. The used variables differ from one tree to the other, leading to very large sets of features involved in the ensemble decision process. This is particularly the case when dealing with high-dimensional datasets such as biomedical data.

In order to better understand tree ensemble models, bagging-based ensembles, such as RF, provide a feature importance index $J_a$ (cf. Sec-
5.4 Conclusion

This experimental study shows that further growing trees after the convergence of predictive performances is beneficial in two aspects. The individual class predictions stabilise after growing one order of magni-

Figure 5.3: Feature selection stability (as measured by Kuncheva’s index) of RF with respect to the number of trees, computed on 200 resamplings of the DLBCL dataset.

tion 3.3.3). It can be used to rank variables according to their importance in the classification of new unseen samples. We study here the stability of the most prominent features resulting from this additional selection and we compare it to the class prediction stability.

We measure the stability of feature selection through Kuncheva’s index, described in Section 4.2. Experiments in [PVD12], show that the stability of the top 25 most important features attains an asymptotic value with at least one order of magnitude more trees than for stable class prediction. Those conclusions remain true for other signature sizes.

As we can see on Figure 5.3, RFs of at least 10,000 trees are needed in order to converge to the best feature selection stability on the DLBCL dataset that contains $p = 7129$ variables. Other experiments on an ECG dataset of $p = 262$ variables show that this convergence rate depends on the number of variables. It appears after 1000 or 2000 trees are grown.
Feature selection stability converges at the cost of growing an additional order of magnitude more trees. The actual number of trees needed to reach the different stabilities may vary on other tasks. It is proportional to the number of features in the data. Yet, we expect that the various convergences appear in the same order in all cases.

When used to perform feature selection on high dimensional genomic datasets, we observe that at least 10,000 trees are needed to reach the plateau of stability with Random Forests.
Chapter 6

$J_{\chi^2}$: A statistically interpretable importance index from Random Forest

$J_{\chi^2}(x_j)$ is the p-value that feature $x_j$ is important.

As explained in Section 3.3.3, Random Forest (RF) [Bre01] provides an index, $J_a$, that ranks variables according to their importance in predicting new samples. It is multivariate as it measures how important a variable is in a prediction task that also involve other features. In addition, it handles heterogeneous data. It is however not straightforward to interpret $J_a$ in a statistical sense. In particular, it is very difficult to decide which variables are truly important in the decision process. It only provides a feature ranking.

The second contribution of this thesis is to propose a statistical procedure to measure variable importance that tests if variables are significantly useful in combination with others in a forest. Like $J_a$, the proposed multivariate RF feature importance index uses out-of-bag (OOB) samples (see Section 2.2.2) to measure changes in the distribution of class votes when permuting a particular variable. It produces $p$-values measuring to which extent features are useful in combination with other variables of the model. Such $p$-values offer a natural threshold for deciding which variables are statistically relevant.

This chapter gives an overview of a work published in the Neurocomputing journal. The paper [PD15] is annexed in Chapter B. The remainder of this chapter is organised as follows. Section 6.1 gives the
Chapter 6. $J_{\chi^2}$

6.1 Definition of $J_{\chi^2}$

$J_{\chi^2}$ is the combination of the idea of Breiman’s $J_a$ (cf. Section 3.3.3) to use a permutation test with an analysis of the tree class vote distribution of the forest. We propose to perform a statistical test that assesses whether permuting a variable significantly influences that distribution. The hypothesis is that removing an important variable signal by permuting it should change individual tree predictions, hence the class vote distribution.

In order to estimate this distribution, we rely on the out-of-bag (OOB) data to simulate unseen samples. Indeed, each decision tree can predict the labels of its corresponding OOB samples. The result of such classification can be compared to the true labels using a confusion matrix. In a binary classification setting, the confusion matrix of the $k$-th tree classifying its OOB takes the following form:

\[
\begin{array}{c|cc}
\text{true} & \text{pred} & -1 & 1 \\
-1 & TN_k & FP_k \\
1 & FN_k & TP_k \\
\end{array}
\]

where $TN_k$ is the number of correct prediction of class $-1$ (true negative), $FP_k$ incorrect prediction of class $1$ (false positive), etc. Similarly to $J_a$, we can do the same when permuting a particular feature $x_j$ in order to remove its signal. We obtain one new confusion matrix per tree which might be different from the one computed on the original OOB. The two confusion matrices of each tree can then be re-arranged into a contingency table that encodes the change in class vote prediction when permuting variable $x_j$. The two columns of that contingency table corresponds to the two vectorised confusion matrices (one for the original OOB, the second for the OOB with permuted $x_j$). If there are only two
6.1. Definition of $J_{\chi^2}$

classes, the table of the $k$-th tree for variable $x_j$ is

<table>
<thead>
<tr>
<th>class vote</th>
<th>is $x_j$ permuted</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$TN_k$</td>
<td>$TN_{k,j}^\circ$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$FP_k$</td>
<td>$FP_{k,j}^\circ$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$FN_k$</td>
<td>$FN_{k,j}^\circ$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$TP_k$</td>
<td>$TP_{k,j}^\circ$</td>
<td></td>
</tr>
</tbody>
</table>

\[ (6.2) \]

where $TN_{k,j}^\circ$ denotes the number of true negative predictions when variable $x_j$ is permuted, etc. Summing those tables over all the OOBs gives an estimate of the class vote distribution of the whole forest on unseen examples, with and without permuting variable $x_j$:

\[
\begin{array}{c|c|c|c|c}
\text{class vote} & \text{is } x_j \text{ permuted} & \text{No} & \text{Yes} \\
\hline
TN & TN_{k,j} & \sum_{k=1}^T TN_k & TN_{k,j} \\
FP & FP_{k,j} & FP_k & FP_{k,j} \\
FN & FN_{k,j} & FN_k & FN_{k,j} \\
TP & TP_{k,j} & TP_k & TP_{k,j} \\
\end{array}
\]

\[ (6.3) \]

where $T$ is the total number of trees in the forest.

A Pearson’s $\chi^2$ test is then used to assess whether the frequencies of those events significantly differ from the original $x_j$ and its permuted version $\tilde{x}_j$. Rejecting the null hypothesis with a low $p$-value means that the permutation significantly influences the class vote distribution and, therefore, that $x_j$ is important in the current predictive model. We note that, even on small datasets, there is no need to consider a Fisher’s exact test instead of Pearson’s $\chi^2$ test since cell counts are generally sufficiently large: the sum of all counts is twice the sum of all OOB sizes.

Since the importance of several features is typically assessed through this test on the same data, $p$-values must be corrected for multiple testing. We use the popular Benjamini-Hochberg correction [BH95] to control the false discovery rate. The new importance measure $J_{\chi^2}(x_j)$ is defined as the $p$-value that $x_j$ influences the class vote distribution, corrected for multiple testing.

The proposed index can easily be generalized to multi-class problems. In such cases, the contingency table of Equation (6.3) simply has $c^2 \times 2$ entries, where $c$ is the number of classes.

$J_{\chi^2}$ is implemented inside $jForest$. This open-source library is available on https://github.com/jeromepaul/jForest.
6.2 Concordance with $J_a$

This statistical importance index is closely related to Breiman’s $J_a$ index (see Section 3.3.3). $J_a$ computes the average increase in error rate when permuting a variable $x_j$. In $J_{\chi^2}$, it corresponds to counts of FP and FN for permuted and non permuted variable $x_j$, the second and third lines of the contingency table in Equation (6.3). There are some important differences between both approaches however. Firstly, $J_a$ aggregates both types of errors in a single measure, which might lose important information in case of unbalanced class priors. Secondly, the central term of $J_a$ (Equation (3.1), page 52) is normalized by each OOB size while the contingency table of $J_{\chi^2}$ (Equation (6.3)) considers global counts. This follows from the fact that $J_a$ estimates an average increase in classification error on the OOB samples while $J_{\chi^2}$ measures a distribution shift on those samples. Finally, the very nature of those importance indices differ. $J_a$ is an average measure of differences between prediction performances whereas $J_{\chi^2}$ is a corrected $p$-value from a $\chi^2$ test. The higher $J_a$, the more important the corresponding variable. In contrast, the lower $J_{\chi^2}$, the stronger the evidence to reject the null hypothesis that permuting this variable does not affect the voting process of a RF. There is also a natural significance threshold for $J_{\chi^2}$ since any corrected $p$-value lower than 5% is commonly accepted as significant [Sti08].

Experiments in [PD15] analyse the concordance between $J_a$ and $J_{\chi^2}$. The top plot of Figure 6.1 compares the rankings of those two importance measures on one particular resampling of the DLBCL dataset (cf. Chapter F). It shows that feature ranks in the top 500 are highly correlated. Spearman’s rank correlation coefficient is 0.97 between both rankings. The main differences are observed in the poorly ranked features, which are those very unlikely to be considered significant. While $J_a$ penalizes features whose permuted versions would increase the prediction accuracy, $J_{\chi^2}$ would favour such features since they affect the class vote distribution. In particular, after rank 1,250 on the horizontal axis, features have a negative $J_a$ value for they lower the prediction performance of the forest. Yet, since they influence the class vote distribution, they are considered more important by $J_{\chi^2}$.

This behavior of $J_{\chi^2}$ could be considered undesirable but the actual effect is negligible in practice because the large ranks of those variables indicate that they are very unlikely to be eventually selected. This is further confirmed by the bottom plot of Figure 6.1 where the mean rank of each variable is computed over 200 resamplings. We can see that this effect totally disappears and is only due to random variations on those features. To sum up, this particular behavior of $J_{\chi^2}$ has virtually no
6.2. Concordance with $J_a$

Figure 6.1: Top: Rankings produced by $J_a$ and $J_{\chi^2}$ on one resampling of the DLBCL dataset. Bottom: Mean rankings produced by $J_a$ and $J_{\chi^2}$, averaged over 200 resamplings of the DLBCL dataset. Approximately 1,800 features are ranked after pre-filtering 75% of features with the lowest variances.
practical impact since only top ranked features will typically be selected based on their low corrected \( p \)-values. It follows that feature selection based only on the ranks of \( J_a \) and \( J_{\chi^2} \) perform equally well.

In addition, the time complexity of computing both indices is exactly the same. If we assume, to simplify the analysis, that each tree node splits its instances into two sets of equal sizes until having one observation per leaf, then the depth of a tree is in \( O(\log n) \) and the time complexity of classifying one example by a single tree is \( O(\log n) \). The global time complexity of computing a ranking of \( p \) variables from an ensemble of \( T \) trees is in \( O(T \cdot p \cdot n \cdot \log n) \).

### 6.3 \( J_{\chi^2} \) highlights important variables

The expected advantage of \( J_{\chi^2} \) over \( J_a \) is that it provides a natural threshold to select variables \( i.e. \) a feature with a \( p \)-value \( \leq 0.05 \) is deemed relevant. Experiments show that this criterion matches the selection of relevant variables provided that a sufficient amount of trees are grown.

In order to control to which extent \( J_{\chi^2} \) is able to highlight important variables, we resort on artificial data to control by design the signal in different features. A dataset with a linear decision boundary is generated with \( n = 500 \) samples and \( p = 110 \) features among which only 10 are informative.

Experiments in [PD15] compute variable importance with \( J_a \) and \( J_{\chi^2} \) from RF of different sizes \( T \) and \( m \) values \( (i.e. \) number of candidate variables in each split, see Section 2.2.2). They show that the original Breiman’s \( J_a \) index does not offer a clear threshold to decide which variables are relevant. Our \( J_{\chi^2} \) index appears to distinguish more clearly between relevant and irrelevant variables. It however requires a relatively large number of trees to gain confidence that a feature is indeed relevant. Figure 6.2 shows the importance of the top ranked features with a forest of 10,000 trees.

When computed on small forests, \( J_{\chi^2} \) may fail to identify variables as significantly important. Nevertheless those variables are still correctly ranked. Increasing the value of the \( m \) meta-parameter also tends to positively impact the identification of those variables when the number of trees is low. This beneficial effect appears less strongly as the number of trees increases. In general, the larger the forests the better, in terms of the significance of the test.

Those results are consistent with the analysis of Chapter 5. Indeed a high number of trees provides a better feature selection stability. Similarly, we gain confidence that features are useful in the predictive model.
6.3. $\chi^2$ highlights important variables

Figure 6.2: Importance indices computed on an artificial dataset with 10 informative features out of 110 features in total. For the sake of visibility, $J_a$ has been rescaled between 0 and 1. The horizontal line is set at 0.05. $J_{\chi^2}(x_j)$ below this line are deemed statistically relevant. All 10 informative features appear at the top of each ranking.
by growing many trees.

### 6.4 Good prediction from significant features

$J_{\chi^2}$ is a multivariate feature importance measure that considers each variable jointly with the others in the forest. Experiments presented on 10 real-life datasets show that restricting the final classifier to be built strictly on the features which are deemed significant still offers good predictive results.

Results in [PD15] show that the number of variables eventually considered significant largely varies across datasets. Like in Section 6.3, the number of significant variables increases with the number of trees considered to compute $J_{\chi^2}$. For most genomic datasets, a forest of 10,000 trees highlights from 5 to 10 significant features on average. They yield predictive performances similar to those of a RF built on the top 50 features according to $J_a$. The observations on lower dimensional datasets are fully consistent with results on genomic data. However, the number of trees needed to highlight relevant features that lead to good predictive performances tends to be lower.

### 6.5 $J_{\chi^2}$ outperforms alternatives

Like $J_{\chi^2}$, mr-Test and 1Probe (see Section 3.4) are two alternative approaches that can convert Breiman’s $J_a$ index to $p$-values that features are important. We show in [PD15] that $J_{\chi^2}$ is conceptually simpler than both approaches and that it is more efficient in terms of number of trees to be grown.

mr-Test and 1Probe transform feature rankings to statistically interpretable variable importance indices. They are more general than $J_{\chi^2}$ because they work with any feature ranking algorithm. Yet, their $p$-values require the computation of $N$ feature rankings each obtained from some dataset variation. For instance, mr-Test and 1Probe can extract $p$-values from RF when combined to $J_a$. To do so, they have to repeatedly build $N$ forests in order to rank the variables with $J_a$. This approach is thus conceptually more complicated than $J_{\chi^2}$ that is computed from only one RF.

Like for $J_{\chi^2}$, experiments on artificial data ($n = 500$, $p = 110$ with 10 informative features) show that 1Probe and mr-Test gain confidence that features are relevant when growing more and more trees. Performances comparable to $J_{\chi^2}$ with $T = 10,000$ trees are obtained with 1Probe with 100,000 trees ($N = 100$ and $T = 1000$). On the other side, mr-Test
looks too conservative and keeps missing informative features even with a very high number of trees.

Experiments on 10 real-life datasets confirm all the observations on artificial datasets. In addition, a finer analysis of the selected features show that all significant variables from $J_{\chi^2}$ are also selected by 1Probe. To sum up, significant features selected by $J_{\chi^2}$ or 1Probe lead to good predictive performances on all datasets. However, 1Probe requires a much larger number of trees to reach those performances. mr-Test is too conservative and fail to select important variables.

6.6 Discussion

The number of significant features highlighted by $J_{\chi^2}$ depends on the size of the forest. It is to be expected since feature selection tends to get more stable with the number of trees. However, another phenomenon has to be taken into account: the sample size of the statistical test. $J_{\chi^2}$ relies on a $\chi^2$ test applied on the contingency table of Equation 6.3. As the number of observations in that contingency table is proportional to the number of trees in the forest, growing more trees increases the sample size of the test. Because of that, even small differences in the class vote distribution become significant, provided a sufficiently large number of observations. Indeed, $\chi^2$ test will find out more and more significant variables until, in the limit, all features that marginally change the OOB predictions become significant. It follows that $J_{\chi^2}$ is not suitable to estimate the number of trees needed to highlight important features. It is only a statistical measure of how much variables are useful in the prediction of the current RF model. For feature selection, we therefore advise to scale the forest to reach the plateau of feature selection stability and then to estimate the significativity of the variables with $J_{\chi^2}$.

Another limitation of $J_{\chi^2}$ concerns the detection of informative features that are correlated. During the tree growing process, the algorithm has little reason to prefer one of the correlated features over the others. Because they share the same information, each variable is selected less often inside a node than if they were not correlated. As a result, their importance is undervalued and $J_{\chi^2}$ could fail to identify them as significantly important. This problem is inherent in the forest growing process and also appears with $J_D$. A possible solution is to detect correlated features a priori and to keep only one of them as a surrogate for the whole set.

We will close this section with a remark on the use of Pearson’s $\chi^2$ test to assess feature importance. This statistical test assumes the
independence of observations. However, one can argue that observations in the contingency table of Equation 6.3 are not independent because the various OOBs can share common samples. Yet, some preliminary experiments using a non-parametric Kolmogorov-Smirnov test on the accuracies and permuted accuracies of the trees provide results that are very similar to those of $\chi^2$. Therefore, the (potential) dependence between observations does not seem to be a problem, probably because class votes are issued from quite different decision trees.

6.7 Conclusion

$J_{\chi^2}$ is a multivariate feature importance index that is statistically interpretable. It is designed for ensemble methods that leave out-of-bag samples for each base classifier, such as Random Forest. It outputs $p$-values of a $\chi^2$ test that the class vote distribution of the ensemble changes when permuting a particular variable. Selecting features with the natural threshold of a $p$-value $\leq 0.05$ clearly distinguishes between informative and uninformative variables. While $J_{\chi^2}$ requires a high number of trees to find out important features, it outperforms recent alternatives that demand an order of magnitude even more trees to reach similar performances.
Chapter 7

Two kernel approaches for heterogeneous feature selection

RFE with a specific kernel is good at heterogeneous feature selection.

To the best of our knowledge, little effort has been dedicated to develop feature selection methods tailored for datasets with both categorical and numerical values. Section 3.3 presents a few methods that deal with heterogeneous data, among which $J_a$ which is based on Random Forest (RF) and a greedy forward selection called ‘hybrid feature selection’.

The third contribution of this thesis is to present two new heterogeneous feature selection methods. They are based on the Recursive Feature Elimination (RFE) procedure using either a non-linear SVM or Multiple Kernel Learning (MKL). This chapter gives an overview of the main results published in [PDD15]. The full paper is included in Chapter C.

The current chapter is organised as follows. Section 7.1 describes $RFE^{MKL}$ and $RFE^{SVM}$, the two proposed methods. Then, Section 7.2 summarises the experimental assessment of those approaches. Finally, Section 7.3 concludes this chapter.

7.1 Introducing $RFE^{MKL}$ and $RFE^{SVM}$

RFE (cf. Section 3.3.5) is an embedded backward elimination strategy that iteratively builds a feature ranking by removing the least important features in a classification model at each step. It is most commonly
used in combination with a linear SVM from which feature weights are extracted. However, it can be used with any classification model from which individual feature importance can be deduced. We propose to compute the importance of both continuous and categorical features from MKL or SVM combined with the clinical kernel (cf. Section 2.3.2).

As a recall, the clinical kernel (see Equation 2.22, page 35) is an unweighted average of \( p \) base kernels, one per feature. In addition, MKL (see Section 2.4) learns a linear combination of a series of base kernels. Following the idea developed in [CLW07] and [JVN14], we naturally adapt the clinical kernel with MKL by defining one kernel per feature. Those \( p \) kernels correspond to the sub-kernels \( k_f \) of the clinical kernel. Instead of simply averaging them, we let MKL compute the weights of each sub-kernel in the final combination. The obtained kernel has the following form:

\[
k(x_i, x_j) = \sum_{f=1}^{p} \mu_f k_f(x_i, x_j) \quad \text{s.t. } \mu_f \geq 0
\]

\[
k_f(x_i, x_j) = \begin{cases} 
I(x_{if} = x_{jf}) & \text{if } x_f \text{ is categorical} \\
\frac{(\max_f - \min_f) - |x_{if} - x_{jf}|}{\max_f - \min_f} & \text{if } x_f \text{ is continuous}
\end{cases}
\]  

where \( \mu_f \) is the kernel weight learned by MKL for the \( f \)-th sub-kernel. As \( \mu_f \) reflects the influence of kernel \( k_f \) in the decision function (cf. Equation 2.28, page 38), it can be seen as the importance of feature \( x_f \). In order to provide a full ranking of the features, the kernel weight vector \( \mu \) is \( l_2 \)-regularised (see Section 2.4). This regularisation has the additional advantage to provide more stable feature rankings than \( l_1 \) methods [ZY06]. We call the combination of RFE with this feature importance \( RFE^{MKL} \). It specifically uses the kernel weights \( |\mu_f| \) as feature importance values to eliminate at each iteration a prescribed fraction of the least relevant features.

The second method proposed, \( RFE^{SVM} \), uses feature importance of a non-linear SVM with the clinical kernel. In the case of a linear SVM, one can measure the importance of the features by looking at their respective weights in the hyperplane. When dealing with a non-linear SVM, we can instead look at the variation in margin size \( \frac{1}{||w||} \).

Since the larger the margin, the lower the generalisation error (at least in terms of bound) [Vap95], a feature that does not decrease much the margin size is not deemed important for generalisation purposes. From the SVM dual formulation (cf. Section 2.3), we know that the margin is
7.2. Performance assessment

inversely proportional to

\[ W^2(\alpha) = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) = \|w\|^2, \quad (7.2) \]

where \( \alpha_i \) and \( \alpha_j \) are the dual variables of a SVM, \( y_i \) and \( y_j \) the labels of \( x_i \) and \( x_j \), out of \( n \) training examples, and \( k \) a kernel. Therefore, the importance of a particular feature \( x_f \) can be approximated without re-estimating \( \alpha \) by the following formula:

\[ J_{SVM}(x_f) = |W^2(\alpha) - W^2_{(-f)}(\alpha)| \quad (7.3) \]

\[ W^2_{(-f)}(\alpha) = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i^{-f}, x_j^{-f}), \quad (7.4) \]

where \( x_i^{-f} \) is the \( i \)th training example without considering feature \( x_f \). This importance index computes the impact on the margin of removing one feature from the current hyperplane. Indeed, recomputing a SVM (hence \( \alpha \)) could lead to a quite different decision function. In that case, measuring the difference between this margin size and the original one would not make much sense.

The \( RFE^{SVM} \) approach is simply the combination of RFE with the \( J_{SVM} \) feature importance.

7.2 Performance assessment

There are essentially two competitors that perform heterogeneous feature selection: tree ensemble methods such as RF and the hybrid feature selection (HFS) approaches (see Section 3.3). Experiments in [PDD15] compare \( RFE^{MKL} \) and \( RFE^{SVM} \) to \( HFS^{NB} \), \( HFS^{5NN} \) and RF of 10,000 trees on 7 real-life datasets. We use a resampling procedure to estimate the predictive performances and feature selection stability of the methods for various signature sizes.

Chapter E extends those analyses to an additional baseline that relies on a continuous encoding of the categorical variables. This additional baseline does not modify the conclusions drawn from the experiments reported in this chapter.

The comparison of all selection techniques across all feature set sizes and datasets show that \( RFE^{MKL} \) outputs the best feature sets from a predictive performances point of view. Statistical tests show that \( RFE^{MKL} \) performs significantly better than \( HFS^{5NN} \) and \( RFE^{SVM} \).
which appear at the end of the ranking. The data does not show significant differences between the predictive performances of $RFE^{MKL}$, $RF10000$ and $HFS^{NB}$.

Regarding stability, it appears that the $RFE$ approaches perform worse than the other methods. $RFE^{MKL}$ is however not significantly less stable than $HFS^{NB}$ and $RF10000$. In addition, the two $HFS$ approaches may have the natural advantage that they are based on filter methods that are more stable than embedded methods. This is studied in [HGV11] and a discussion about the stability of multivariate approaches in presence of redundant features is annexed in Chapter D. Moreover, the RFs had to be run with a very large number of trees (10,000) to provide a stable feature selection (as advocated in Chapter 5).

A complementary analysis compares the various feature selection methods for a fixed number of features. For each dataset, we choose the smaller feature set size such that the BCR of $RFE^{MKL}$ lies in the 95% confidence interval of the best $RFE^{MKL}$ predictive performance. Results show that the two best ranked methods, RF10000 and $RFE^{MKL}$, perform significantly better than $RFE^{SVM}$ in terms of BCR. For the stability, the only significant difference is between RF10000, the most stable method, and $RFE^{SVM}$, the less stable one. Those results on a fixed number of features show that the $RFE^{MKL}$ and RF10000 are the two best performing methods without significant differences between them, but at a larger computational cost for the latter.

### 7.3 Conclusion

$RFE^{MKL}$ and $RFE^{SVM}$ are two new heterogeneous feature selection techniques that complement the few existing approaches. Experiments show that $RFE^{MKL}$ produces state-of-the-art predictive performances and is as good as competing methods in terms of feature selection stability. It offers results similar to Random Forests with smaller computational times. The second approach, $RFE^{SVM}$, performs worse than $RFE^{MKL}$. It also seems less efficient in terms of prediction and stability than competing approaches, even though not significantly different from all competitors.
Chapter 8

Conclusion

This thesis proposes feature selection methods for high-dimensional heterogeneous data that include both continuous and categorical variables. The main objective is to improve the interpretability of predictive models by highlighting variables that are useful to do classification. To do so, we focus on two families of approaches that deal with heterogeneous data: tree ensemble and kernel methods. The different contributions of this work are summarised hereafter.

Tree ensemble methods perform classification of heterogeneous data and provide a feature index to rank variables according to their importance in the predictive model. However, the randomness required in those methods induces a high variability in the feature rankings. Our analysis shows that growing more trees than needed for prediction is beneficial in two aspects. Firstly, the class prediction of individual samples stabilises with an order of magnitude more trees. Secondly, the feature rankings reach a threshold in stability with an additional order of magnitude more trees. On high dimensional genomic datasets, this threshold is attained with at least 10,000 trees.

We also propose a solution to a second problem of the feature importance index extracted from tree ensemble methods. Breiman’s original index is good at ranking variables but it is very hard to deduce which are the features that are actually important in the decision process. To alleviate this limitation, we introduce a new importance index, $J_{\chi^2}$. It has the interpretability of a statistical test and produces $p$-values that features are important to predict the class of previously unseen data points. We can thus highlight relevant variables with the usual 0.05 threshold. Experiments showed that $J_{\chi^2}$ ranks features similarly to Breiman’s index with the additional benefit to clearly highlight relevant variables.
without increasing the computational cost. In order to gain confidence that variables are relevant, $J_{\chi^2}$ requires a high number of trees, typically 10,000 for genomic datasets. It corresponds to the number of trees needed to reach the asymptotic feature selection stability with ensemble methods. In comparison to recent alternatives, $J_{\chi^2}$ produces similar sets of significant variables with a gain of an order of magnitude less trees.

Kernel methods are another way to deal with heterogeneous data. Carefully designed kernels can successfully handle both continuous and categorical variables to perform classification. Our last contribution is to enlarge the family of few heterogeneous feature selection methods by proposing two kernel-based approaches. They make use of the recursive feature elimination (RFE) procedure that extracts variable importance from a multiple kernel learning (MKL) model or a non-linear SVM. We show experimentally that $RFE^{MKL}$ offers state-of-the-art performances. The second approach, $RFE^{SVM}$ has the same level of performances as some competitors but it seems less good than $RFE^{MKL}$. These methods also outperform the standard alternative that considers a linear RFE with the categorical features re-encoded into continuous variables.

All in all, the contributions of this thesis tend to improve the interpretability of predictive models that contain different kinds of variables. In particular, good feature selection methods are needed in the biomedical domain where the number of dimensions often prevent physicians from understanding why predictive models accurately predict some medical outcome. Firstly, our analysis of tree ensemble stabilities provides a simple way to obtain stable variable rankings with high-dimensional data. Secondly, we improve the variable importance index of tree ensemble methods by making it statistically interpretable. $J_{\chi^2}$ highlights which variables significantly help in the prediction task. On genomic datasets, it is able to identify small sets of significant features that lead to good predictive performances. Finally, two kernel methods were proposed to increase the very small number of feature selection approaches that deal both with continuous and categorical variables.

**Perspectives**

The field of heterogeneous feature selection remains quite unexplored. This thesis proposes a few methods that open paths for further developments.

The experimental analysis of the stabilities of tree ensemble methods (see Chapter 5) lacks a formal analysis of the observed convergence rate of feature selection stability. An analysis of the stability of in-
individual class predictions is proposed in [HLMMS11, HLMMS13]. A similar approach for feature selection stability may lead to a better understanding of variable importance in an ensemble. An important step in this direction is made in [LWSG13] to understand the nature of tree ensemble variable importance indices, but we are not aware of any theoretical analysis of the feature selection stability with respect to the number of trees. Another interesting research target should be to promote stability with smaller forests, without losing the benefits of the ensemble. Techniques to prune forest may be a first line of approach. Previous work (e.g. [HL09]) explore pruning techniques in order to obtain smaller forests with similar predictive performances. In that case, trees of pruned forests should be grown from important variables, which may be in favour of stability. However, feature selection stability is a metric defined over different signatures, obtained from different forests. As pruning may reduce the number of variables used in ensembles, it could further increase the variability among the selected feature sets, which would negatively impact stability.

\( J_{\chi^2} \), the statistically interpretable feature importance index is presented as a measure extracted from random forests (see Chapter 6). However, it can be computed from any tree ensemble method that leaves aside some out-of-bag samples while growing the forest (e.g. bagging [Bre96], cForests [SBZH07]). Future studies should assess the impact of using \( J_{\chi^2} \) jointly with those methods. In addition to introducing \( J_{\chi^2} \), we show that measuring the distribution shift of class votes before and after permuting a feature in a tree ensemble conveys some useful information. The specific test to characterize such distribution shift is a bit less central and other alternatives can be explored. For instance, some preliminary experiments show that a Kolmogorov-Smirnov test comparing the accuracy distributions, with and without permuting a particular variable, presents similar results to those obtained with \( J_{\chi^2} \). Many other alternative could be evaluated. In the same spirit, one could easily design further variants to focus on some specific functions of the confusion matrix. For instance, when dealing with very imbalanced datasets, one could use a Kolmogorov-Smirnov test to assess the change in balanced classification rate which is preferred to accuracy in those cases. As another example, we could also promote the selection of features that play a more critical role in the sensitivity of the classifier while putting less emphasis on the specificity. Future works in that direction could make use of \textit{jForest} \footnote{https://github.com/jeromepaul/jForest}, a very flexible open-source implementation of tree ensemble methods, developed during this thesis.
Another interesting perspective is related to the sample size of the statistical test performed in $J_{\chi^2}$. Indeed, enlarging the RF size naturally leads to an increase of the number of features that are deemed statistically significant. Beyond significance itself, it would also be interesting to study the effect size evaluated by such a statistical procedure. Yet, this particular issue seems quite difficult. Indeed, the change in class prediction in $J_{\chi^2}$ or the decrease in accuracy in $J_a$ is expected to be very small when randomly permuting a feature. This is because the base learners are multivariate and also because each particular variable does not appear in each tree (see Section 3.3.3). Therefore, we would probably be limited to compare the change in class vote distribution of the best ranked variables to the one of lower ranked features.

The two kernel-based feature selection methods presented in Chapter 7 clearly suffer from a poor stability. Even though they lead to good predictive performances, the variability in the feature rankings is misleading when trying to interpret the results. A possible way to improve that aspect is to resort on ensemble feature selection methods [AHVdP+10]. This is not to be confused with feature selection from ensemble methods. The idea is to build a consensus signature from different sets of selected variables e.g. keeping only the most common ones.

This thesis also leads to more global extensions. In particular, we only propose feature selection methods to perform classification. Yet, tree ensemble approaches also perform regression [Bre01] and even survival prediction [IKBL08]. It would be interesting to see if the analysis of stabilities still holds and to create a statistically interpretable feature selection index similar to $J_{\chi^2}$ in those cases. One could, for instance, consider a paired t-test to measure the difference between the distributions of residuals with and without permuting a particular variable. Kernel methods also perform regression [DBK+97] and survival analysis [VBPVHS11]. Another possible extension of this work would consider adapting the kernel-based feature selection techniques to those two areas.

If we take a step back and think about the heterogeneity of the data, there is also a room for future developments. Indeed, we only considered two kinds of variables: those which express on a continuous scale and those which partition the samples into different categories. There are, of course, many other different types of features which can further increase the heterogeneity of the data. For instance, ordered variables (e.g. young/middle-aged/old), string data, graphs, pictures or sequences would require other feature selection schemes. Kernel methods could be a good starting point since they can handle and mix those
different kinds of variables. Feature selection could therefore be applied with approaches similar to \( RFE^{MKL} \) which is presented in this thesis. However, a special attention should be paid to the normalisation of the different kernels that are used together. In this work, \( RFE^{MKL} \) relies on a specific kernel designed so that continuous and categorical variables have the same influence. Correctly scaling other kernels will be mandatory to interpret the kernel weights of \( MKL \) as a feature importance measure [KBSZ11].

Finally, we would also suggest to assess the interpretability of predictive models in a real setting, with a team of clinicians. The goal of the methods developed in this thesis is to help medical doctors to understand diseases or biomedical processes better. We are convinced that stable feature selection is one of the keys to get into high dimensional biomedical problems. Yet, physicians that have to come up with a biological interpretation of predictive models have the final say. It would certainly be interesting to directly collaborate with them to further assess and improve their understanding of machine learning methods for personalised medicine. For instance, as predictive models are generally multivariate, we should put some energy into the development of tools that analyse and explain interactions between the involved biomedical parameters with respect to the literature.
Part III

Main published papers
Chapter A

The stability of feature selection and class prediction from ensemble tree classifiers


Abstract

The bootstrap aggregating procedure at the core of ensemble tree classifiers reduces, in most cases, the variance of such models while offering good generalization capabilities. The average predictive performance of those ensembles is known to improve up to a certain point while increasing the ensemble size. The present work studies this convergence in contrast to the stability of the class prediction and the variable selection performed while and after growing the ensemble. Experiments on several biomedical datasets, using random forests or bagging of decision trees, show that class prediction and, most notably, variable selection typically require orders of magnitude more trees to get stable.
A.1 Motivation

The generalization capabilities of Random Forests (RF) [Bre01], and similar tree ensemble classifiers, is known to increase up to a certain point with the number of trees in the forest. This number is commonly chosen, typically through an internal cross-validation, to reach a plateau of the predictive performance. However, stability issues of such ensemble have not been extensively studied so far. Our first objective is to assess to which extent the predictive performance convergence appears earlier than a stable class prediction. In other words, while the average predictive performance no longer changes significantly once this plateau has been reached, the specific labels assigned to each test example can still vary. The bootstrap mechanism at the core of the estimation of such classifiers is known to reduce variance in most cases and indeed stabilizing the individual class prediction is expected but possibly with a larger number of trees.

Tree ensemble techniques also perform an embedded variable selection. Such a selection already occurs at each node while growing the various trees. It can also be performed globally once the forest is built. Breiman suggests in particular a permutation test to select the most relevant features from a Random Forest [Bre01]. In the present work, we study the stability of this variable selection in contrast to the convergence of the average predictive performance and of the class predictions. Our central question of interest is to assess to which extent the variable selection is more brittle than the individual class predictions. Our experiments conducted on various biomedical datasets, with RF and bagging of decision trees or stumps, show that orders of magnitude more trees are typically required to get a stable variable selection as compared to reaching a stable class prediction.

A.2 Ensemble of tree classifiers

Bagging of decision trees is arguably among the simplest approaches to overcome the strong tendency of a single decision tree to over-fit the learning data. Bagging (bootstrap aggregating) builds a set of classifiers from successive bootstrap samples of the original training set. The final classifier combines individual decision trees by a majority vote, a form of unweighted averaging [Bre96]. The diversity of the ensemble combined with the final averaging is known to increase the robustness of the aggregated classifier. Random Forests (RF) go one step further to promote the ensemble diversity by randomly sub-sampling the set of
features to be evaluated at each node while growing the trees [Bre01].

The approaches mentioned so far mostly focus on improving the predictive accuracy of the tree ensemble under various settings. The number of trees is typically tuned to the smallest possible ensemble size while still reaching the asymptotic predictive performance. Pruning techniques can also be used to further reduce the ensemble size with a marginal loss, or sometimes even a gain, in predictive accuracy [MMHLS09].

A distinct but related issue is the stability of the class prediction, that is to which extent the class label predicted for each test example stays the same over different data resamplings. The experiments reported in section A.4 show that stable class predictions can be observed but at the cost of increasing the ensemble size beyond the convergence of the prediction accuracy.

Ensembles of tree classifiers are also commonly used to select features. The very process of learning decision trees includes a greedy selection of a most discriminant feature at each node, according to a relevance index such as information gain, gain ratio or the Gini index. However the diversity of the various trees, even though beneficial for the predictive accuracy, may result in a large set of used features, some of them only marginally present in the ensemble. Further selecting the most prominent variables increases the interpretability of the combined classifier, a key aspect for applications such as medical diagnosis from gene expression measurements. Such a selection can be performed according to the number of times a given feature appears in the forest, possibly weighting each feature occurrence by its relevance at the corresponding node. An even better alternative to estimate the importance of each feature to classify unseen examples relies on a permutation test computed on the out-of-bag examples from each bootstrap round [Bre01]. For each variable, one compares the out-of-bag classification accuracy after permuting the feature values on those examples with the accuracy obtained from the original values. The more the classification performance drops after permutation the more important is estimated the corresponding feature. We study here the stability of the most prominent features resulting from this additional selection and we compare it to the class prediction stability.

A.3 Experimental design and assessment

We aim at assessing the predictive and stability performances of ensemble tree classifiers while growing the number $T$ of trees in the ensemble. For the sake of this study, we compare a representative set of
such classification methods: bagging of unpruned CART trees [BFOS84], RF [Bre01], as well as bagging of decision stumps and RF of decision stumps. Practical experiments are conducted on several biomedical datasets described below. Most of them fall into the small $n$ (number of examples), large $p$ (number of features) setting. In those cases, a standard 10-CV protocol is likely to show highly variable results when the number of examples is limited to a few tens. Hence, we rely on $K = 200$ random splittings of the data into train (90%) and test (10%). Each data partitioning results from uniform sampling without replacement (a significance test for such a protocol is proposed in [NB03]) and we report average predictive performances over all resamplings.

Predictive performance is measured by the Balanced Classification Rate (BCR), which is the per class accuracy, averaged over the various classes. BCR is preferred to accuracy for classification problems with unequal class priors. BCR is also simpler than ROC analysis for multi-class problems. For binary classification problems, BCR simply reduces to the arithmetic average between specificity and sensitivity.

The stability of the class prediction measures to which extent each individual test example is assigned the same class label across various resamplings. For each example $x_i$, let $c^*$ denote the most commonly predicted class label (across all resamplings for which that example appears in a test fold); let $n_{c^*x_i}$ be the number of times such a prediction occurs out of the $n_{x_i}$ occurrences of $x_i$ in a test fold. The class prediction stability is given by:

$$1 \sum_{i=1}^{n} \frac{n_{c^*x_i}}{n_{x_i}},$$

where $n$ denotes the total number of examples, each of them appearing approximately $0.10 \times K = 20$ times in a test fold. Such a stability index falls in the interval $[\frac{1}{|C|}, 1]$ with $|C|$ classes. The stability is equal to 1 when each test example is always assigned the same, although not necessarily correct, class label.

The stability of the feature selection can be measured according to the Kuncheva Index [Kun07]. This index measures to which extent $K$ sets $S_i$ of $s$ selected features share common features:

$$K(\{S_1, ..., S_K\}) = \frac{2}{K(K - 1)} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \frac{|S_i \cap S_j| - \frac{s^2}{p}}{s - \frac{s^2}{p}},$$

where $p$ is the total number of features, and feature selection is performed on each of the $K$ training folds. The $s^2/p$ term corrects a bias due to the chance of selecting common features among two sets chosen.
A.4. Results

The experimental results are reported in figures A.1, A.2, A.3 and A.4. They show no significant BCR differences across the various ensemble classifiers, but for Arrhythmia (Figure A.4) on which stumps have sig-
significantly lower results ($p-$value $< 2.10^{-10}$ according to the corrected resampled t-test [NB03]). For all methods, the convergence of the predictive performance is typically reached after 10 or 20 trees for the four datasets. The same conclusions can be drawn when the predictive performance is estimated from the classification accuracy instead of the BCR (results not shown). The class predictions typically require an order of magnitude more trees (100...200 trees) to get stable. Feature selection only get stable, and yet not perfectly, from at least an order of magnitude more trees ($\geq 1,000$) on the 3 genomic datasets. An earlier convergence is obtained for Arrhythmia (Figure A.4) as a natural consequence of a smaller total number of features and more training samples. Bagging of decision stumps also tends to offer an earlier convergence of the feature selection stability. This makes sense as there is only one feature selected for each stump without any random sampling of the feature space.

A.5 Conclusion and perspectives

Our experimental study demonstrates, for a variety of ensemble tree classifiers, that stable class predictions and, most notably, stable feature selection require orders of magnitude more trees than those needed to reach the asymptotic predictive performance. Our future work includes a more formal analysis (similarly to [HLMMS11]) of such behaviors, and possibly ways to promote an earlier stability without losing the benefits of the ensemble.
A.5. Conclusion and perspectives

Figure A.1: Alon dataset: Predictive performances, class prediction and feature selection stabilities for an increasing number of trees
Figure A.2: DLBCL dataset: Predictive performances, class prediction and feature selection stabilities for an increasing number of trees
A.5. Conclusion and perspectives

Figure A.3: van’t Veer dataset: Predictive performances, class prediction and feature selection stabilities for an increasing number of trees
Figure A.4: Arrhythmia dataset: Predictive performances, class prediction and feature selection stabilities for an increasing number of trees
Chapter B

Inferring statistically significant features from Random Forests

Abstract

Embedded feature selection can be performed by analyzing the variables used in a Random Forest. Such a multivariate selection takes into account the interactions between variables but is not straightforward to interpret in a statistical sense. We propose a statistical procedure to measure variable importance that tests if variables are significantly useful in combination with others in a forest. We show experimentally that this new importance index correctly identifies relevant variables. The top of the variable ranking is largely correlated with Breiman’s importance index based on a permutation test. Our measure has the additional benefit to produce p-values from the forest voting process. Such p-values offer a very natural way to decide which features are significantly relevant while controlling the false discovery rate. Practical experiments are conducted on synthetic and real data including low and high-dimensional datasets for binary or multi-class problems. Results show that the proposed technique is effective and outperforms recent alternatives by reducing the computational complexity of the selection process by an order of magnitude while keeping similar performances.

B.1 Introduction

Feature selection aims at finding a subset of most relevant variables for a prediction task. To this end, univariate filters, such as a t-test, are commonly used because they are fast to compute and their associated $p$-values are easy to interpret. However such a univariate feature ranking does not take into account the possible interactions between variables. In contrast, a feature selection procedure embedded into the estimation of a multivariate predictive model typically captures those interactions.

A representative example of such an embedded variable importance measure has been proposed by Breiman with its Random Forest [Bre01] (RF) algorithm. While this importance index is effective to rank variables, it is difficult to decide how many such variables should eventually be kept. This question could be addressed through an additional validation protocol at the expense of an increased computational cost. In this work, we propose an alternative that avoids such additional cost and offers a statistical interpretation of the selected variables.

The proposed multivariate RF feature importance index uses out-of-bag (OOB) samples to measure changes in the distribution of class votes when permuting a particular variable. It produces $p$-values, corrected for multiple testing, measuring to which extent variables are useful in combination with other variables of the model. Such $p$-values offer a natural threshold for deciding which variables are statistically relevant.

The remainder of this document is organized as follows. Section B.2 sets up the context and introduces our proposed variable importance measure relying on a $\chi^2$ test. Section B.3 describes the metrics, experimental protocol and datasets used to assess the performances of feature selection indices. Comparative experiments with state-of-the-art methods are reported in Section B.4. Section B.5 summarizes our contribution and discusses some possible future work.

B.2 Material and methods

This section presents a novel feature selection index from tree ensembles, typically a Random Forest. Section B.2.1 introduces our notations and reviews Breiman’s RF feature importance measure. Our proposed feature importance index is presented in Section B.2.2, along with related work.
B.2. Material and methods

B.2.1 Context and notations

Let $X^{n \times p}$ be a data matrix consisting of $n$ observations in a $p$-dimensional space and $y$ a vector of size $n$ containing the corresponding class labels. A RF model [Bre01] is made of an ensemble of trees, each of which is grown from a bootstrap sample of the $n$ data points. For each tree, the selected samples form the bag (denoted by $B$), the remaining samples form the out-of-bag (OOB) denoted by $\overline{B}$. Let $\mathcal{B}$ stand for the set of bags over the ensemble and $\overline{\mathcal{B}}$ be the set of corresponding OOBs. We have $|\mathcal{B}| = |\overline{\mathcal{B}}| = T$, the number of trees in the forest.

In order to compute feature importances, Breiman [Bre01] proposes a permutation test procedure based on classification error. For each variable $x_j$, there is one permutation test per tree in the forest. For an OOB sample $\overline{B}_k$ corresponding to the $k$-th tree of the ensemble, one considers the original values of the variable $x_j$ and a random permutation $\tilde{x}_j$ of its values on $\overline{B}_k$. The difference in prediction error using the permuted and original variable is recorded and averaged over all the OOBs in the forest. The higher this index, the more important the variable is assumed because it corresponds to a stronger increase of the classification error when permuting it. The importance measure $J_a$ of the variable $x_j$ is precisely defined as:

$$J_a(x_j) = \frac{1}{T} \sum_{B_k \in \mathcal{B}} \frac{1}{|\overline{B}_k|} \left( \sum_{i \in \overline{B}_k} I(h_k^{\tilde{x}_j}(i) \neq y_i) - I(h_k(i) \neq y_i) \right)$$

(B.1)

where $y_i$ is the true class label of the OOB example $i$, $I$ is an indicator function, $h_k(i)$ is the class label of the example $i$ as predicted by the tree estimated on the bag $B_k$, $h_k^{\tilde{x}_j}(i)$ is the predicted class label from the same tree while the values of the variable $x_j$ have been permuted on $\overline{B}_k$. Such a permutation does not change the tree but potentially changes the prediction on the out-of-bag examples since its $j$-th dimension is modified after the permutation. Since the predictors with the original variable $h_k$ and the permuted variable $h_k^{\tilde{x}_j}$ are individual decision trees, the sum over the various trees where this variable is present represents the ensemble behavior, respectively from the original variable values and its various permutations. Whenever a specific variable does not appear in a tree, the prediction cannot be affected by permuting its value, which means that the specific term corresponding to this tree in equation (B.1) is null.
B.2.2 A statistical feature importance index from RF

While $J_a$ is able to capture individual variable importances conditioned to the other variables used in the forest, it is not easily interpretable. In particular, it does not define a clear threshold to highlight statistically relevant variables. In the following sections, we propose a statistical feature importance measure closely related to $J_a$, and compare it with existing approaches providing a statistical interpretation to feature importance scores.

Definition

In the present work, we combine the idea of Breiman’s $J_a$ to use a permutation test with an analysis of the tree class vote distribution of the forest. We propose to perform a statistical test that assesses whether permuting a variable significantly influences that distribution. The hypothesis is that removing an important variable signal by permuting it should change individual tree predictions, hence the class vote distribution.

One can estimate this distribution using the OOB data to simulate unseen examples. In a binary classification setting, for each data point in an OOB, the prediction of the corresponding tree can fall into one of the four following cases: correct prediction of class 1 (TP), correct prediction of class 0 (TN), incorrect prediction of class 1 (FP) and incorrect prediction of class 0 (FN). Summing the occurrences of those cases over all the OOBs gives an estimate of the class vote distribution of the whole forest on unseen examples. The same can be performed when permuting a particular feature $x_j$ to evaluate the effect on the class vote distribution after perturbing this variable. The various counts obtained can be arranged into a $4 \times 2$ contingency table defined as follows for each variable $x_j$ and its permuted version $\tilde{x}_j$:

<table>
<thead>
<tr>
<th></th>
<th>$x_j$</th>
<th>$\tilde{x}_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TN</td>
<td>$s(0, 0)$</td>
<td>$s^{\tilde{x}}(0, 0)$</td>
</tr>
<tr>
<td>FP</td>
<td>$s(0, 1)$</td>
<td>$s^{\tilde{x}}(0, 1)$</td>
</tr>
<tr>
<td>FN</td>
<td>$s(1, 0)$</td>
<td>$s^{\tilde{x}}(1, 0)$</td>
</tr>
<tr>
<td>TP</td>
<td>$s(1, 1)$</td>
<td>$s^{\tilde{x}}(1, 1)$</td>
</tr>
</tbody>
</table>

where

$$ s(l_1, l_2) = \sum_{\mathcal{B}_k \in \mathcal{R}} \sum_{i \in \mathcal{B}_k} I(y_i = l_1 \text{ and } h_k(i) = l_2) $$

and $s^{\tilde{x}}(l_1, l_2)$ is defined similarly with $h^{\tilde{x}}(i)$ instead of $h_k(i)$. 

$$ s(l_1, l_2) = \sum_{\mathcal{B}_k \in \mathcal{R}} \sum_{i \in \mathcal{B}_k} I(y_i = l_1 \text{ and } h_k(i) = l_2) $$

$$ s^{\tilde{x}}(l_1, l_2) = \sum_{\mathcal{B}_k \in \mathcal{R}} \sum_{i \in \mathcal{B}_k} I(y_i = l_1 \text{ and } h^{\tilde{x}}(i) = l_2) $$
A Pearson’s $\chi^2$ test is then used to assess whether the frequencies of those events significantly differ from the original $x_j$ and its permuted version $\tilde{x}_j$. Rejecting the null hypothesis with a low $p$-value $p_{\chi^2}(x_j)$ means that permuting variable $x_j$ significantly influences the class vote distribution and, therefore, that $x_j$ is important in the current predictive model. We note that, even on small datasets, there is no need to consider a Fisher’s exact test instead of Pearson’s $\chi^2$ since cell counts are generally sufficiently large: the sum of all counts is twice the sum of all OOB sizes.

Since the importance of several features is typically assessed through this test on the same data, $p$-values must be corrected for multiple testing. We use the popular Benjamini-Hochberg correction [BH95] to control the false discovery rate. Let $p_{\chi^2}^{FDR}(x_j)$ be the value of $p_{\chi^2}(x_j)$ after FDR correction, the new importance measure is defined as

$$J_{\chi^2}(x_j) = p_{\chi^2}^{FDR}(x_j)$$

(B.4)

The proposed index can easily be generalized to multi-class problems, as used in some of the experiments reported in Section B.3. In such cases, the contingency table (B.2) simply has $c^2 \times 2$ entries, where $c$ is the number of classes.

This statistical importance index is closely related to Breiman’s $J_a$. The two terms inside the innermost sum of Equation (B.1) correspond to counts of FP et FN for permuted and non permuted variable $x_j$. This is encoded by the second and third lines of the contingency table (B.2). There are some important differences between both approaches however. Firstly, $J_a$ aggregates both type of errors in a single measure, which might lose important information in case of unbalanced class priors. Secondly, the central term of $J_a$ (eq. (B.1)) is normalized by each OOB size while the contingency table of $J_{\chi^2}$ (eq. (B.2)) considers global counts. This follows from the fact that $J_a$ estimates an average increase in classification error on the OOB samples while $J_{\chi^2}$ measures a distribution shift on those samples. Finally, the very nature of those importance indices differ. $J_a$ is an average measure of differences between prediction performances whereas $J_{\chi^2}$ (eq. (B.4)) is a corrected $p$-value from a $\chi^2$ test. The higher $J_a$ the more important is the corresponding variable assumed. In contrast, the lower $J_{\chi^2}$ the stronger the evidence to reject the null hypothesis that permuting this variable does not affect the voting process of a RF. There is also a natural significance threshold for $J_{\chi^2}$ since any corrected $p$-value lower than 5% is commonly accepted as significant [Sti08].

The time complexity of computing $J_{\chi^2}$ for $p$ variables is exactly the same as with Breiman’s $J_a$. If we assume, to simplify the analysis, that
each tree node splits its instances into two sets of equal sizes until having one observation per leaf, then the depth of a tree is in $O(\log n)$ and the time complexity of classifying one example by a single tree is $O(\log n)$. The global time complexity of computing a ranking of $p$ variables from an ensemble of $T$ trees is in $O(T \cdot p \cdot n \cdot \log n)$. Algorithm B.1 describes the computation of $J_{\chi^2}$ and motivates its time complexity analysis.

| Algorithm B.1: Algorithm for computing the importance of all variables within a forest of $T = |\mathcal{F}|$ trees. |
|-----------------------------------------------|
| **init(res)**                                 |
| // Set to 0 a $p$-dimensional vector; $\Theta(p)$ |
| **for** $x_j \in \text{Variables do}$          |
| // $\Theta(p)$                                |
| **init(contTable)**                            |
| // Initialise contingency table with 0’s; $\Theta(1)$ |
| **for** $\mathcal{B}_k \in \mathcal{B}$ do   |
| // $\Theta(T)$                                |
| $\tilde{x}_j \leftarrow \text{perm}(x_j, \mathcal{B}_k)$ |
| // $\Theta(n)$                                |
| **for** $i \in \mathcal{B}_k$ do              |
| // $\Theta(n)$                                |
| $a \leftarrow h_k(i)$                         |
| // $\Theta(\text{depth})$                     |
| $b \leftarrow h_k^{\tilde{x}(i)}(i)$         |
| // $\Theta(\text{depth})$                     |
| contTable $\leftarrow \text{update}(\text{contTable}, a, b, y_i)$ |
| // $\Theta(1)$                                |
| **end**                                       |
| **end**                                       |
| $\text{res}[x_j] \leftarrow \chi^2(\text{contTable})$ |
| // $\Theta(1)$                                |
| **return** res                               |

**Related work**

In [HTSWG12], the authors compare several ways to obtain a statistically interpretable index from a feature relevance score. Their goal is to convert feature rankings to statistical measures such as the false discovery rate, the family wise error rate or $p$-values. Their proposed methods typically make use of an external permutation procedure to compute some null distribution from which those metrics are estimated. The external permutation tests repeatedly compute feature rankings on dataset variations, e.g. for which some features are randomly permuted. Similarly to our approach, this work can be applied to convert Breiman’s $J_a$ index to a statistically interpretable measure and to produce $p$-values on which a prescribed threshold can be easily defined. Yet, the methods proposed in [HTSWG12] are somewhat more complex since they rely on an additional resampling protocol. This external resampling encompasses the growing of many forests on top of the internal...
B.3. Experiments

This section describes various metrics to assess the performance of feature selection methods, our experimental protocol and the datasets on which we run our experiments.

B.3.1 Performance metrics

The Balanced Classification Rate (BCR) is used to assess the predictive performances of a classifier estimated on the selected features. It is defined as the mean of the classification accuracy in each class. BCR is preferred to the standard classification rate when dealing with unbalanced class priors. It also generalizes to multi-class problems more
Appendix B. \( J_\chi^2 \) @ NEUCOMP 2015

Easily than ROC analysis. For a two-class problem, BCR is defined as the average between sensitivity and specificity:

\[
BCR = \frac{1}{2} \left( \frac{TP}{P} + \frac{TN}{N} \right)
\]

(\( B.5 \))

Its multi-class generalization takes the following form:

\[
BCR = \frac{1}{c} \sum_{l=1}^{c} \frac{TC_l}{C_l}
\]

(\( B.6 \))

where \( c \) is the number of classes, \( TC_l \) is the number of correct predictions of class \( l \) and \( C_l \) is the total number of samples of class \( l \). This metric has been used, for instance, in the performance prediction challenge\(^1\) held at WCCI 2006 precisely to deal with possible class imbalance while considering the calibration of specific models [GADB06].

Stability of feature selection indices quantifies how selected sets of features vary after small perturbations of the datasets. The Kuncheva index (KI) [Kun07] specifically measures to which extent \( K \) sets (typically obtained from various resamplings) of \( s \) selected features share common elements.

\[
KI(\{S_1, \ldots, S_K\}) = \frac{2}{K(K-1)} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \frac{|S_i \cap S_j| - \frac{s^2}{p}}{s - \frac{s^2}{p}}
\]

(\( B.7 \))

where \( p \) is the total number of features and \( \frac{s^2}{p} \) is a term correcting the random chance, for 2 feature sets \( S_i \) and \( S_j \), to share common features. This index ranges within \((-1, 1]\). The larger its value, the larger the number of commonly selected features. A value of 0 is the expected stability for a selection performed uniformly at random.

**B.3.2 Experimental protocol**

In order to evaluate the predictive performances and the stability provided by a feature selection technique, an external resampling protocol is used. The goal is twofold. Firstly, resampling allows to assess how a particular classifier built on the selected features will predict the class of new data. Secondy, it mimics small perturbations in datasets to assess the stability of feature selection. The procedure consists in repeating \( N \) times the following steps:

\(^1\)The evaluation metric in this challenge actually relied on BER, the balanced error rate, which conveys the same information since \( BCR = 100\% - BER \).
B.3. Experiments

randomly select a training set $Tr$ made of 90% of the available data. The remaining 10% form the test set $Te$.

- train a forest of $T$ trees on $Tr$ and rank the features it uses
- for each number of selected features $s$
  * train a forest of 500 trees using only the first $s$ features on $Tr$
  * save the BCR computed on $Te$ and the set of $s$ features

The statistics recorded at each iteration are then aggregated to provide the mean BCR and the KI values. The above protocol considers several feature set sizes $s$. The results presented in section B.4.1 reports, on several datasets, how many out of $s$ are actually significant features.

B.3.3 Datasets

Artificial datasets allow to control by design the signal present in different features. Our first experiments are inspired from [HTSWG12] and conducted on artificial datasets with a linear decision boundary. Labels $y \in \{-1, 1\}^n$ are given by $y = \text{sign}(Xw)$ where $w \in \mathbb{R}^p$ and $X \in \mathbb{R}^{n \times p}$. Each dimension from the input data $X$ is repetitively drawn from a $\mathcal{N}(0,1)$ distribution. The number $p$ of variables is set to 110. The first 10 weights $w_i$ are randomly sampled from a uniform distribution $\mathcal{U}(0.5, 1)$. The other 100 weights are set to 0 such that only the first 10 variables are relevant. We draw $n = 500$ instances for a given run with a design matrix $X \in \mathbb{R}^{500 \times 110}$. Finally, 10% of the $y$ labels are randomly flipped to add some noise to the classification task.

Experiments are also performed on real-life datasets, briefly described in Table B.1 in terms of class priors and number of input features. We consider firstly four gene expression datasets from a microarray technology. The number of features $p$ in those datasets is typically much larger than the number $n$ of training examples. In such a challenging setting, feature selection is usually considered particularly important. The DLBCL [SRT+02] dataset aims at predicting the outcome of diffuse large b-cell lymphoma. The prediction task associated to the Lymphoma [AED+00] dataset concerns the identification of different subtypes of this pathology. Golub’s dataset [GST+99] aims at identifying different types of cancer. Finally, the Prostate [SFR+02] dataset focuses on the diagnostic of prostate cancer or healthy patients from their gene expression. Since the number of features in those datasets is orders of magnitude higher than the number of available samples, a non-specific filter (i.e. without considering the class labels) is applied first to remove 75% of the features with the lowest variance on the training set.
In addition, we consider six lower dimensional datasets with, proportionally, a larger number of training examples. Breast tissue [FA10] is a four classes dataset made of impedance measurements to predict the type of observed tissue. The Glass [FA10] dataset aims at classifying fragments of glass into seven different types using proportions of chemical elements that compose each fragment. The Wine [FA10] dataset consists of chemical measurements aiming at predicting from which of three domains comes a particular wine. The purpose of the Vehicle [FA10] dataset is to distinguish between four vehicle types given some geometrical features extracted from their silhouettes. The Musk1 [FA10] dataset describes two kinds of molecules (musk and non-musk) in terms of shape and conformation of the molecules. Finally, the Arrhythmia [FA10] dataset aims at predicting the presence of cardiac arrhythmia from ECG measurements.

## B.4 Results and discussion

The following sections present experiments that highlight properties of the $J_{\chi^2}$ importance measure. They show that $J_{\chi^2}$ actually provides an importance index from which a natural selection threshold can be chosen (Section B.4.1). Our results also illustrate that $J_{\chi^2}$ is closely related to $J_a$ (Section B.4.2), the original Brieman’s index, both in terms of variable rankings and predictive performances after building a classifier on the selected features. Further experiments described in Section B.4.3 present predictive performances obtained when restricting the classifier to be built only from variables which are deemed statistically significant.
Finally, Section B.4.4 details the relative performance of the $J_{\chi^2}$ and the two competing approaches mr-Test and 1Probe.

### B.4.1 Selecting statistically relevant features with $J_{\chi^2}$

The expected benefit of $J_{\chi^2}$ is to offer a principled way to select key variables from a tree ensemble. One aims at restricting the selection to those variables that are deemed significant for characterizing the class vote distribution in such an ensemble. We assess here to which extent this criterion matches the selection of relevant variables by design on an artificial dataset (cf. Section B.3.3).

A RF, built on the full dataset, is used to rank the variables according to their importance index. Similarly to [HTSWG12], a given variable is considered significantly important, whenever its p-value falls below 0.05 after correcting for multiple testing. Figures B.1 and B.2 reports importance indices obtained by forests of various sizes and $m$ values. This meta-parameter $m$ corresponds to the number of variables randomly sampled as possible candidates in each tree node while growing the forest. The specific ensemble sizes are chosen according to [PVD12] in which the stability of such ensembles is studied. This work shows in particular that a forest of 500 trees performs quite well in terms of predictive performances on such high dimensional datasets while a much larger ensemble of about 10,000 trees is required to reach a stable feature selection. In all plots of Figure B.1 and B.2, the 10 informative features appear at the top of the rankings of $J_a$ and $J_{\chi^2}$.

The results reported in Figure B.1 and B.2 illustrate that the original (decreasing) Breiman’s $J_a$ index does not offer a clear threshold to decide which variables are relevant. Our (increasing) $J_{\chi^2}$ index appears to distinguish more clearly between relevant and irrelevant variables. It however requires a relatively large number of trees to gain confidence that a feature is indeed relevant. When computed on small forests (left plots), $J_{\chi^2}$ may fail to identify variables as significantly important. Nevertheless those variables are still correctly ranked. Increasing the value of the $m$ meta-parameter also tends to positively impact the identification of those variables when the number of trees is low. This beneficial effect appears less strongly as the number of trees increases. In general, the larger the forests the better, in terms of the significance of the test. Beyond significance, the effect size could also be assessed as briefly discussed in section B.5.
Figure B.1: Importance indices computed on an artificial dataset with 10 informative features out of 110 features in total. Results are reported for various forest sizes ($T$) and $m$ values (see text). For the sake of visibility, $J_a$ has been rescaled between 0 and 1. The horizontal line is set at 0.05. $J_{\chi^2}(x_j)$ below this line are deemed statistically relevant. All 10 informative features appear at the top of each ranking in the four plots.
B.4. Results and discussion

Figure B.2: Importance indices computed on an artificial dataset with 10 informative features out of 110 features in total. Results are reported for various forest sizes \( T \) and \( m \) values (see text). For the sake of visibility, \( J_a \) has been rescaled between 0 and 1. The horizontal line is set at 0.05. \( J_{\chi^2}(x_j) \) below this line are deemed statistically relevant. All 10 informative features appear at the top of each ranking in the four plots.
B.4.2 Concordance with $J_a$

As discussed in Section B.2.2, $J_{\chi^2}$ and $J_a$ share some similarities and the same computational complexity to be evaluated. The top plot of Figure B.3 compares the rankings of those two importance measures on one particular resampling of the DLBCL dataset (cf. Section B.3.3). It shows that feature ranks in the top 500 are highly correlated. Spearman's rank correlation coefficient is 0.97 between both rankings.

The main differences are observed in the poorly ranked features, which are those very unlikely to be considered significant. While $J_a$ penalizes features whose permuted versions would increase the prediction accuracy, $J_{\chi^2}$ would favor such features since they affect the class vote distribution. In particular, after rank 1,250 on the horizontal axis, features have a negative $J_a$ value for they lower the prediction performance of the forest. Yet, since they influence the class vote distribution, they are considered more important by $J_{\chi^2}$.

This behavior of $J_{\chi^2}$ could be considered undesirable but the actual effect is negligible in practice because the large ranks of those variables indicate that they are very unlikely to be eventually selected. This is further confirmed by the bottom plot of Figure B.3 where the mean rank of each variable is computed over 200 resamplings. We can see that this effect totally disappears and is only due to random variations on those features. To sum up, this particular behavior of $J_{\chi^2}$ has virtually no practical impact since only top ranked features will typically be selected based on their low corrected $p$-values.

We further show that $J_a$ and $J_{\chi^2}$ are also similar in terms of stability of the feature selection and predictive performances of the final classifier built from the selected features. Figure B.4 presents the measurements made over 200-resamplings from the DLBCL dataset according to the number of features kept to train a RF as final classifier. It shows that the two indices behave very similarly when plotting predictive performance with respect to the number of selected features. Increasing the number of trees increases the feature selection stability in both cases, with a convergence of the stability curves observed from 5000 trees. Similar results have been reported for $J_a$ in [PVD12].

B.4.3 Prediction from significant features

The previous results show that $J_{\chi^2}$ ranks top features roughly the same way as $J_a$. Since $J_{\chi^2}$ is a corrected $p$-value, it is associated to a commonly accepted threshold equal to 0.05 to decide whether a variable should eventually be kept. We recall that this selection process is no longer univariate (as would be the case of a t-test) but is performed while
Figure B.3: Top: Rankings produced by $J_a$ and $J_{\chi^2}$ on one external resampling of the DLBCL dataset. Bottom: Mean rankings produced by $J_a$ and $J_{\chi^2}$, averaged over 200 resamplings of the DLBCL dataset. Approximately 1,800 features are ranked after pre-filtering 75% of features with the lowest variances.
Figure B.4: Average BCR and KI of $J_a$ and $J_{\chi^2}$ over 200 resamplings of the DLBCL dataset according to the number $s$ of selected features, for various numbers $T$ of trees.
considering each variable jointly with the others in the forest. One can wonder whether restricting the final classifier to be built strictly on the features which are deemed significant still offers good predictive results.

We follow here the protocol of Section B.3.2 and observe that the number of significant variables increases with the number of trees considered. This is consistent with the results already presented in Section B.4.1. Table B.2 specifically reports the results obtained from genomic data. The number of variables eventually considered significant largely varies across datasets. Almost no features are considered statistically significant on the DLBCL dataset (similar results are observed in [HTSWG12]). For the three other datasets and provided the number of trees is sufficiently large, the predictive performances of a RF built on significant features only (according to $J_{\chi^2}$) are similar to those of a RF built on the top 50 features according to $J_a$. While those predictive performances are close (especially with 10,000 trees) the differences are statistically significant in most cases. Yet, it is worth stressing that the final forest can be built on only a few key features with good predictive results.

Table B.3 reports similar results on the lower dimensional datasets, with different ratios between the numbers of learning examples and input features, including multi-class problems. As before, the average number of significant features increases with the number of trees. On the Breast tissue, Glass, Wine and Vehicle datasets, results show that nearly all features appear significant even with only 500 trees. Overall these results are fully consistent with those observed on the high dimensional genomic datasets but the number of trees needed to highlight relevant features that lead to good predictive performances tends to be lower.

B.4.4 Comparison of $J_{\chi^2}$ to 1Probe and mr-Test

In this section we compare $J_{\chi^2}$ to 1Probe and mr-Test, two methods proposed in [HTSWG12] and briefly reviewed in section B.2.2. Section B.4.4 compares the number of trees needed to highlight important variables from synthetic datasets. Section B.4.4 compares the predictive performances and signatures obtained using only significant variables.

Discovery rates evaluated on synthetic datasets

The performances of 1Probe and mr-Test on synthetic datasets are assessed in [HTSWG12] using $T = 1000$ trees and $N = 1000$ external resamplings. The total number of trees considered is therefore 1,000,000 in contrast to $J_{\chi^2}$ which show comparable results with only 10,000 trees (and no external resampling).
Appendix B. $J_{\chi^2}$ @ NEUCOMP 2015

<table>
<thead>
<tr>
<th>dataset</th>
<th>$T$</th>
<th>$\text{avg}(s^{rel})$</th>
<th>$\text{min}(s^{rel})$</th>
<th>$\text{max}(s^{rel})$</th>
<th>BCR</th>
<th>BCR$_{50}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLBCL</td>
<td>5000</td>
<td>0.04</td>
<td>0.00</td>
<td>1.00</td>
<td>0.50</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>0.99</td>
<td>0.00</td>
<td>5.00</td>
<td>0.61</td>
<td>0.88</td>
</tr>
<tr>
<td>golub</td>
<td>5000</td>
<td>5.96</td>
<td>3.00</td>
<td>10.00</td>
<td>0.93</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>10.82</td>
<td>8.00</td>
<td>14.00</td>
<td>0.96</td>
<td>0.97</td>
</tr>
<tr>
<td>lymphoma</td>
<td>5000</td>
<td>0.66</td>
<td>0.00</td>
<td>6.00</td>
<td>0.93</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>4.85</td>
<td>2.00</td>
<td>9.00</td>
<td>0.93</td>
<td>0.94</td>
</tr>
<tr>
<td>prostate</td>
<td>5000</td>
<td>4.95</td>
<td>2.00</td>
<td>8.00</td>
<td>0.93</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>7.92</td>
<td>6.00</td>
<td>11.00</td>
<td>0.93</td>
<td>0.94</td>
</tr>
</tbody>
</table>

Table B.2: Various statistics obtained over 200-resamplings when keeping only significant features. $T$ is the number of trees used to build the forest. $\text{avg}(s^{rel})$ (resp. max, min) is the average (resp. maximum, minimum) number of significant features according to $J_{\chi^2}$. BCR is the average predictive performance of a RF built from significant features only. BCR$_{50}$ is the average BCR obtained when using the 50 best ranked features according to $J_a$.

We aim here at comparing the 3 approaches with a similar computational budget and perform the same experiment on synthetic datasets as in Section B.4.1 with $T = 10,000$ for $J_{\chi^2}$ and $N \times T = 100 \times 100$ for 1Probe and mr-Test. This setting appears to be inadequate for the latter approaches. The number of trees $T$ in the forest for each resampling is too low to rank variables correctly. The two methods hardly find any of the important variables.

Better results are reported on Figure B.5 where the 10 informative features appear at the top of the ranking of each of the three methods. The number of trees is increased to $T = 1000$ (a total of 100,000 trees over all resamplings). $J_{\chi^2}$ (with 10 times fewer trees in total) and 1Probe are both able to highlight significant variables at the top of the ranking. In contrast, mr-Test does not consider them significant even though they are well ranked.

After repeating the above experiment on 10 runs for generating synthetic datasets, it appears that the number of true discoveries (i.e. $p$-value $\leq 0.05$ and actually informative feature) is on average 6.8 for $J_{\chi^2}$, 7.2 for 1Probe and 3.4 for mr-Test. A Friedman test [Dem06] shows a significant difference between the performances of the three approaches ($p$-value of $6 \times 10^{-3}$).

The Nemenyi post-hoc test [Dem06], illustrated by a critical difference diagram on Figure B.6, shows that the difference in performances of 1Probe and $J_{\chi^2}$ is not significant while the mr-Test performs significantly worse. All methods have a very high precision with an average of 0.1 false discoveries (i.e. $p$-value $\leq 0.05$ and not informative feature)
### B.4. Results and discussion

Table B.3: Various statistics obtained over 200-resamplings when keeping only significant features. $T$ is the number of trees used to build the forest. $\text{avg}(s^{rel})$ (resp. max, min) is the average (resp. maximum, minimum) number of significant features according to $J_{\chi^2}$. BCR is the average predictive performance of a RF built from significant features only. BCR* is the average BCR obtained when using the 50 best ranked features according to $J_\alpha$ or all the available features if the number of variables is less than 50.

<table>
<thead>
<tr>
<th>dataset</th>
<th>$T$</th>
<th>$\text{avg}(s^{rel})$</th>
<th>$\text{min}(s^{rel})$</th>
<th>$\text{max}(s^{rel})$</th>
<th>BCR</th>
<th>BCR*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast tissue</td>
<td>50</td>
<td>6.64</td>
<td>3.00</td>
<td>8.00</td>
<td>0.86</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>7.75</td>
<td>5.00</td>
<td>8.00</td>
<td>0.85</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>7.99</td>
<td>7.00</td>
<td>8.00</td>
<td>0.85</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>8.00</td>
<td>8.00</td>
<td>8.00</td>
<td>0.85</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>8.00</td>
<td>8.00</td>
<td>8.00</td>
<td>0.85</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>2500</td>
<td>8.01</td>
<td>8.00</td>
<td>9.00</td>
<td>0.85</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>8.21</td>
<td>8.00</td>
<td>9.00</td>
<td>0.85</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>8.90</td>
<td>8.00</td>
<td>9.00</td>
<td>0.86</td>
<td>0.86</td>
</tr>
<tr>
<td>Glass</td>
<td>50</td>
<td>4.74</td>
<td>1.00</td>
<td>7.00</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>6.66</td>
<td>3.00</td>
<td>8.00</td>
<td>0.73</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>7.96</td>
<td>5.00</td>
<td>8.00</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>8.00</td>
<td>8.00</td>
<td>8.00</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>8.01</td>
<td>8.00</td>
<td>9.00</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>2500</td>
<td>8.49</td>
<td>8.00</td>
<td>9.00</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>8.94</td>
<td>8.00</td>
<td>9.00</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>9.00</td>
<td>9.00</td>
<td>9.00</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td>Wine</td>
<td>50</td>
<td>7.12</td>
<td>6.00</td>
<td>9.00</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>7.85</td>
<td>7.00</td>
<td>10.00</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>9.86</td>
<td>8.00</td>
<td>12.00</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>11.04</td>
<td>10.00</td>
<td>12.00</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>11.49</td>
<td>11.00</td>
<td>13.00</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>2500</td>
<td>12.76</td>
<td>11.00</td>
<td>13.00</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>12.99</td>
<td>12.00</td>
<td>13.00</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>13.00</td>
<td>13.00</td>
<td>13.00</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>Vehicle</td>
<td>50</td>
<td>15.89</td>
<td>14.00</td>
<td>18.00</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>17.32</td>
<td>16.00</td>
<td>18.00</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>18.00</td>
<td>18.00</td>
<td>18.00</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>18.00</td>
<td>18.00</td>
<td>18.00</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>18.00</td>
<td>18.00</td>
<td>18.00</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>2500</td>
<td>18.00</td>
<td>18.00</td>
<td>18.00</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>18.00</td>
<td>18.00</td>
<td>18.00</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>18.00</td>
<td>18.00</td>
<td>18.00</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td>Musk1</td>
<td>50</td>
<td>0.02</td>
<td>0.00</td>
<td>1.00</td>
<td>0.50</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.25</td>
<td>0.00</td>
<td>2.00</td>
<td>0.53</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>1.91</td>
<td>0.00</td>
<td>5.00</td>
<td>0.67</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>4.89</td>
<td>3.00</td>
<td>9.00</td>
<td>0.73</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>12.15</td>
<td>7.00</td>
<td>18.00</td>
<td>0.81</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>2500</td>
<td>31.62</td>
<td>21.00</td>
<td>38.00</td>
<td>0.87</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>63.23</td>
<td>52.00</td>
<td>77.00</td>
<td>0.90</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>102.08</td>
<td>89.00</td>
<td>118.00</td>
<td>0.90</td>
<td>0.89</td>
</tr>
<tr>
<td>Arrhythmia</td>
<td>50</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.10</td>
<td>0.00</td>
<td>1.00</td>
<td>0.50</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>1.09</td>
<td>0.00</td>
<td>2.00</td>
<td>0.60</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>2.29</td>
<td>1.00</td>
<td>5.00</td>
<td>0.68</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>6.17</td>
<td>2.00</td>
<td>11.00</td>
<td>0.78</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>2500</td>
<td>15.42</td>
<td>10.00</td>
<td>21.00</td>
<td>0.82</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>24.19</td>
<td>18.00</td>
<td>31.00</td>
<td>0.84</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>36.13</td>
<td>29.00</td>
<td>42.00</td>
<td>0.85</td>
<td>0.85</td>
</tr>
</tbody>
</table>
Figure B.5: Importance indices computed on an artificial dataset with 10 informative features out of 110 features in total. The horizontal line is set to 0.05. $p$-values below this line are deemed statistically relevant. The 10 informative features are ranked at the top of those rankings.

for $J_{\chi^2}$, 0.3 for 1Probe and 0 for mr-Test. None of those differences are statistically significant according to a Friedman test.

In summary, the ability of $J_{\chi^2}$ and 1Probe to discover informative variables and discard non-informative variables is essentially the same but 1Probe requires an order of magnitude more trees. In contrast, mr-Test is too conservative as it typically misses informative variables which are wrongly considered not significant.

**Prediction from significant features**

Similar experiments as those described in Section B.4.3 are conducted on the datasets presented in Section B.3.3. We report here comparative predictive performances between the 3 methods when estimating a final RF only on the features that are considered significant.

Table B.4 reports the results obtained with 100 resamplings to evaluate those performances on the genomic datasets. Table B.6 and Table B.7 report results on the lower dimensional datasets. Whenever a specific approach does not select any feature as being significant in a particular resampling, the BCR is fixed to 0.5 since no classifier can be built from zero features and one has to resort on random guessing. The
null distribution is also difficult to define with mr-Test when none of the $p^2$ worst features actually appears in the tree ensembles. In such cases, occurring in particular with few trees, a default BCR=0.5 is also considered.

As observed in [HTSWG12], increasing the number of trees promotes the selection of larger subsets of features. On genomic datasets, considering $100 \times 1000$ trees with 1Probe provides very good predictive performances. mr-Test appears a lot more conservative and tends to select very few or no features. Whenever at least a few genes are considered significant with mr-Test (only for the Golub and Prostate datasets), good predictive performances are observed. The number of $J_{\chi^2}$’s significant features also increases with the number of trees (cf. section B.4.1), providing the best predictive performances with 100,000 trees. The average number of selected features of $J_{\chi^2}$ typically falls between the results of mr-Test and 1Probe. A finer analysis of the gene signatures is presented in Table B.5. It shows that nearly all features estimated to be significant by $J_{\chi^2}$ with 100,000 trees belong to the feature sets selected with 1Probe. In summary, $J_{\chi^2}$ with 10,000 trees already offers good predictive performances on genomic datasets except on DLBCL where more trees are required. 1Probe typically requires an order of magnitude more trees to offer competitive results while mr-Test is often too conservative and selects too few important features.

Experiments on lower dimensional datasets are performed with a smaller number of trees. Table B.6 shows that $J_{\chi^2}$ and 1Probe perform similarly in terms of predictive performances on the Breast tissue, Glass, Wine and Vehicle datasets. However, 1Probe has to grow 5000 trees to be able to select as many features as $J_{\chi^2}$ with only 1000 trees. On the Musk1 and Arrhythmia datasets, which contain many more features, 1Probe needs 100x100 (= 10,000) trees to find out relevant features that lead to predictive performances similar to those of $J_{\chi^2}$ with 2,500
### Table B.4: Various statistics obtained over 100-resamplings when keeping only significant features.

<table>
<thead>
<tr>
<th>dataset</th>
<th>method</th>
<th>$T$ or $N\times T$</th>
<th>avg($s^{ref}$)</th>
<th>min($s^{ref}$)</th>
<th>max($s^{ref}$)</th>
<th>BCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLBCL</td>
<td>$\chi^2$</td>
<td>10,000</td>
<td>0.88</td>
<td>0</td>
<td>4</td>
<td>0.60</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x100</td>
<td>0.02</td>
<td>0</td>
<td>1</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>$\chi^2$</td>
<td>100,000</td>
<td>27.53</td>
<td>22</td>
<td>37</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x1000</td>
<td>45.38</td>
<td>33</td>
<td>60</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x1000</td>
<td>0.03</td>
<td>0</td>
<td>2</td>
<td>0.50</td>
</tr>
<tr>
<td>golub</td>
<td>$\chi^2$</td>
<td>10,000</td>
<td>10.80</td>
<td>8</td>
<td>13</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x100</td>
<td>0.42</td>
<td>0</td>
<td>2</td>
<td>0.64</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>$\chi^2$</td>
<td>100,000</td>
<td>40.85</td>
<td>35</td>
<td>50</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x1000</td>
<td>67.00</td>
<td>53</td>
<td>78</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x1000</td>
<td>9.87</td>
<td>3</td>
<td>15</td>
<td>0.95</td>
</tr>
<tr>
<td>lymphoma</td>
<td>$\chi^2$</td>
<td>10,000</td>
<td>4.83</td>
<td>2</td>
<td>8</td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x100</td>
<td>0.18</td>
<td>0</td>
<td>2</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>$\chi^2$</td>
<td>100,000</td>
<td>27.72</td>
<td>22</td>
<td>34</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x1000</td>
<td>36.45</td>
<td>24</td>
<td>51</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x1000</td>
<td>0.13</td>
<td>0</td>
<td>5</td>
<td>0.50</td>
</tr>
<tr>
<td>prostate</td>
<td>$\chi^2$</td>
<td>10,000</td>
<td>7.89</td>
<td>6</td>
<td>11</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x100</td>
<td>2.98</td>
<td>1</td>
<td>6</td>
<td>0.92</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>$\chi^2$</td>
<td>100,000</td>
<td>41.52</td>
<td>34</td>
<td>53</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x1000</td>
<td>50.20</td>
<td>21</td>
<td>71</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x1000</td>
<td>7.70</td>
<td>5</td>
<td>11</td>
<td>0.93</td>
</tr>
</tbody>
</table>

$T$ is the number of trees used to build the forest. For 1Probe and mr-Test, $N$ indicates the number of external resamplings to compute the null distribution. avg($s^{ref}$) (resp. max, min) is the average (resp. maximum, minimum) number of significant features. BCR is the average predictive performance of a RF built only from features which are estimated significant.
### Table B.5: Average proportion of common genes between various feature sets.

The entry at row $i$ and column $j$ in this matrix represents the average proportion, over 100-resamplings, of significant features selected by the index $i$ that also belong to the feature set selected by method $j$ for the same sampling.

<table>
<thead>
<tr>
<th>dataset</th>
<th>methods</th>
<th>$J_\chi^2$ 10,000</th>
<th>$J_\chi^2$ 100,000</th>
<th>1Probe 100x1000</th>
<th>mrTest 100x1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLBCL</td>
<td>$J_\chi^2$ 10,000</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>$J_\chi^2$ 100,000</td>
<td>0.03</td>
<td>1.00</td>
<td>0.98</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1Probe 100x1000</td>
<td>0.02</td>
<td>0.60</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>mrTest 100x1000</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>golub</td>
<td>$J_\chi^2$ 10,000</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>$J_\chi^2$ 100,000</td>
<td>0.27</td>
<td>1.00</td>
<td>0.99</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td>1Probe 100x1000</td>
<td>0.16</td>
<td>0.61</td>
<td>1.00</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>mrTest 100x1000</td>
<td>0.95</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>lymphoma</td>
<td>$J_\chi^2$ 10,000</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>$J_\chi^2$ 100,000</td>
<td>0.18</td>
<td>1.00</td>
<td>0.98</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>1Probe 100x1000</td>
<td>0.13</td>
<td>0.75</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>mrTest 100x1000</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>prostate</td>
<td>$J_\chi^2$ 10,000</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>$J_\chi^2$ 100,000</td>
<td>0.19</td>
<td>1.00</td>
<td>0.93</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>1Probe 100x1000</td>
<td>0.16</td>
<td>0.78</td>
<td>1.00</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>mrTest 100x1000</td>
<td>0.92</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>
trees, as shown in Table B.7. For a fixed number of trees, $J_{\chi^2}$ appears to highlight more important features than 1Probe. In addition, results presented in Table B.8 show that the majority of the features selected by 1Probe with 100x100 trees are also selected by $J_{\chi^2}$ with 2,500 trees and that they are all selected by $J_{\chi^2}$ with 10,000 trees.

The mr-Test approach fails to select a sufficient amount of features to perform a good prediction on all datasets but Wine. Its assumption that $p^2$ variables are irrelevant seems clearly violated on the Breast tissue, Glass and Vehicle datasets, as attested by the number of variables selected by the two other methods on the same datasets.

To sum up, significant variables selected by $J_{\chi^2}$ or 1Probe also lead to good predictive performances on all datasets. However, 1Probe requires a much larger number of trees to reach those performances. The underlying assumption behind mr-Test is hardly met in practice, which probably explains its poor performances.

B.5 Conclusion and perspectives

We propose in this work a novel feature importance index from random forests. This index $J_{\chi^2}$ produces a feature ranking similar to Breiman’s importance index, especially for top ranked features. It has the additional benefit of being a (corrected) $p$-value from a $\chi^2$ test. Such approach defines a natural threshold to decide which features are estimated statistically important. Unlike a standard t-test, the proposed index is also multivariate as it evaluates the importance of each variable conditioned to the other variables present in the tree ensemble.

Experiments were conducted both on synthetic and real datasets, including low and high-dimensional datasets for binary or multi-class problems. They show that $J_{\chi^2}$ allows us to highlight informative features and discard non-informative ones. Computing $J_{\chi^2}$ has the same computational complexity as Breiman’s index, which is a linear function of the number of trees and the total number of features to be evaluated. $J_{\chi^2}$ is also shown to outperform two recently proposed alternatives, known as 1Probe and mr-Test [HTSWG12].

The selected features with $J_{\chi^2}$ offer similar predictive performances when included in a final classifier as compared to a selection by 1Probe. However, the total number of trees required to reach such performances is typically one order of magnitude smaller with $J_{\chi^2}$, especially on high dimensional data. This computational benefit comes from the fact that $J_{\chi^2}$ is estimated on the out-of-bag samples which have been defined while growing the forest. In contrast, the existing alternatives include
### Table B.6: Various statistics obtained over 200-resamplings when keeping only significant features.

<table>
<thead>
<tr>
<th>dataset</th>
<th>method</th>
<th>T or NxT</th>
<th>$\text{avg}(s^{rel})$</th>
<th>$\text{min}(s^{rel})$</th>
<th>$\text{max}(s^{rel})$</th>
<th>BCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast tissue</td>
<td>$J_{\chi^2}$</td>
<td>1000</td>
<td>8.00</td>
<td>8.00</td>
<td>8.00</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x10</td>
<td>6.09</td>
<td>1.00</td>
<td>8.00</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x10</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>2500</td>
<td>8.01</td>
<td>8.00</td>
<td>9.00</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x25</td>
<td>7.99</td>
<td>7.00</td>
<td>8.00</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x25</td>
<td>0.04</td>
<td>0.00</td>
<td>2.00</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>5000</td>
<td>8.21</td>
<td>8.00</td>
<td>9.00</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x50</td>
<td>8.00</td>
<td>8.00</td>
<td>8.00</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x50</td>
<td>0.43</td>
<td>0.00</td>
<td>2.00</td>
<td>0.59</td>
</tr>
<tr>
<td>Glass</td>
<td>$J_{\chi^2}$</td>
<td>1000</td>
<td>8.01</td>
<td>8.00</td>
<td>9.00</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x10</td>
<td>7.38</td>
<td>4.00</td>
<td>8.00</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x10</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>2500</td>
<td>8.39</td>
<td>8.00</td>
<td>9.00</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x25</td>
<td>8.00</td>
<td>8.00</td>
<td>8.00</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x25</td>
<td>1.10</td>
<td>0.00</td>
<td>4.00</td>
<td>0.48</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>5000</td>
<td>8.94</td>
<td>8.00</td>
<td>9.00</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x50</td>
<td>8.14</td>
<td>8.00</td>
<td>9.00</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x50</td>
<td>2.88</td>
<td>0.00</td>
<td>5.00</td>
<td>0.53</td>
</tr>
<tr>
<td>Wine</td>
<td>$J_{\chi^2}$</td>
<td>1000</td>
<td>11.49</td>
<td>11.00</td>
<td>13.00</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x10</td>
<td>6.04</td>
<td>4.00</td>
<td>8.00</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x10</td>
<td>4.42</td>
<td>0.00</td>
<td>6.00</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>2500</td>
<td>12.76</td>
<td>11.00</td>
<td>13.00</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x25</td>
<td>9.88</td>
<td>7.00</td>
<td>12.00</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x25</td>
<td>5.88</td>
<td>5.00</td>
<td>7.00</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>5000</td>
<td>12.97</td>
<td>12.00</td>
<td>14.00</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x50</td>
<td>11.31</td>
<td>10.00</td>
<td>13.00</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x50</td>
<td>6.05</td>
<td>6.00</td>
<td>7.00</td>
<td>0.98</td>
</tr>
<tr>
<td>Vehicle</td>
<td>$J_{\chi^2}$</td>
<td>1000</td>
<td>18.00</td>
<td>18.00</td>
<td>18.00</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x10</td>
<td>14.21</td>
<td>0.00</td>
<td>18.00</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x10</td>
<td>0.34</td>
<td>0.00</td>
<td>1.00</td>
<td>0.47</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>2500</td>
<td>18.00</td>
<td>18.00</td>
<td>18.00</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x25</td>
<td>17.72</td>
<td>16.00</td>
<td>18.00</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x25</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>5000</td>
<td>18.00</td>
<td>18.00</td>
<td>18.00</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x50</td>
<td>18.00</td>
<td>18.00</td>
<td>18.00</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x50</td>
<td>1.24</td>
<td>1.00</td>
<td>4.00</td>
<td>0.44</td>
</tr>
</tbody>
</table>

*Table B.6: Various statistics obtained over 200-resamplings when keeping only significant features. $T$ is the number of trees used to build the forest. For 1Probe and mr-Test, $N$ indicates the number of external resamplings to compute the null distribution. $\text{avg}(s^{rel})$ (resp. max, min) is the average (resp. maximum, minimum) number of significant features. BCR is the average predictive performance of a RF built only from features which are estimated significant.*
<table>
<thead>
<tr>
<th>dataset</th>
<th>method</th>
<th>$T$ or $N \times T$</th>
<th>$\text{avg}(s_{rel}^{cl})$</th>
<th>$\text{min}(s_{rel}^{cl})$</th>
<th>$\text{max}(s_{rel}^{cl})$</th>
<th>BCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musk1</td>
<td>$J_{\chi^2}$</td>
<td>1000</td>
<td>12.15</td>
<td>7.00</td>
<td>18.00</td>
<td>0.81</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x10</td>
<td>0.01</td>
<td>0.00</td>
<td>1.00</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x10</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>2500</td>
<td>-31.62</td>
<td>-21.00</td>
<td>-38.00</td>
<td>0.37</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x25</td>
<td>0.92</td>
<td>0.00</td>
<td>3.00</td>
<td>0.61</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x25</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>5000</td>
<td>-63.23</td>
<td>-52.00</td>
<td>-73.00</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x50</td>
<td>6.51</td>
<td>0.00</td>
<td>15.00</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x50</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>10000</td>
<td>-102.08</td>
<td>-89.00</td>
<td>-118.00</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x100</td>
<td>49.01</td>
<td>7.00</td>
<td>121.00</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x100</td>
<td>0.69</td>
<td>0.00</td>
<td>2.00</td>
<td>0.60</td>
</tr>
<tr>
<td>Arrhythmia</td>
<td>$J_{\chi^2}$</td>
<td>1000</td>
<td>6.17</td>
<td>2.00</td>
<td>11.00</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x10</td>
<td>0.26</td>
<td>0.00</td>
<td>1.00</td>
<td>0.52</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x10</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>2500</td>
<td>-15.32</td>
<td>-10.00</td>
<td>-21.00</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x25</td>
<td>1.80</td>
<td>1.00</td>
<td>4.00</td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x25</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>5000</td>
<td>-21.19</td>
<td>-18.00</td>
<td>-31.00</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x50</td>
<td>4.92</td>
<td>2.00</td>
<td>11.00</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x50</td>
<td>0.17</td>
<td>0.00</td>
<td>2.00</td>
<td>0.51</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$</td>
<td>10000</td>
<td>-36.15</td>
<td>-29.00</td>
<td>-42.00</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>1Probe</td>
<td>100x100</td>
<td>13.44</td>
<td>6.00</td>
<td>21.00</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td>mrTest</td>
<td>100x100</td>
<td>2.18</td>
<td>0.00</td>
<td>5.00</td>
<td>0.66</td>
</tr>
</tbody>
</table>

Table B.7: Various statistics obtained over 200-resamplings when keeping only significant features. $T$ is the number of trees used to build the forest. For 1Probe and mr-Test, $N$ indicates the number of external resamplings to compute the null distribution. $\text{avg}(s_{rel}^{cl})$ (resp. $\text{max}$, $\text{min}$) is the average (resp. maximum, minimum) number of significant features. BCR is the average predictive performance of a RF built only from features which are estimated significant.
B.5. Conclusion and perspectives

Table B.8: Average proportion of common variables between various feature sets. The entry at row $i$ and column $j$ in this matrix represents the average proportion, over 200-resamplings, of significant features selected by the index $i$ that also belong to the feature set selected by method $j$ for the same sampling.

<table>
<thead>
<tr>
<th>dataset</th>
<th>methods</th>
<th>$J_{\chi^2}$ 2500</th>
<th>$J_{\chi^2}$ 10,000</th>
<th>1Probe 100x100</th>
<th>mrTest 100x100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musk1</td>
<td>$J_{\chi^2}$ 2500</td>
<td>1.00</td>
<td>1.00</td>
<td>0.89</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$ 10,000</td>
<td>0.31</td>
<td>1.00</td>
<td>0.48</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>1Probe 100x100</td>
<td>0.62</td>
<td>0.99</td>
<td>1.00</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>mrTest 100x100</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Arrhythmia</td>
<td>$J_{\chi^2}$ 2500</td>
<td>1.00</td>
<td>1.00</td>
<td>0.69</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>$J_{\chi^2}$ 10,000</td>
<td>0.43</td>
<td>1.00</td>
<td>0.37</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>1Probe 100x100</td>
<td>0.82</td>
<td>1.00</td>
<td>1.00</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td>mrTest 100x100</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

the cost of an additional resampling procedure. The second alternative, mr-Test, is also shown to be too conservative, or even inadequate, and consequently may miss important features which are not estimated to be significant.

We consider here tree ensembles in the specific form of Random Forests. This was originally motivated by the link to be drawn between $J_{\chi^2}$ and the original Breiman’s index. Yet, $J_{\chi^2}$ can in principle be computed from any tree ensemble techniques leaving aside some out-of-bag samples while growing the ensemble. Those include at least bagging of trees, extremely randomized trees [GEW06] and c-Forests [SBZH07]. The impact of considering $J_{\chi^2}$ jointly with these techniques is considered as future work.

On high dimensional data, increasing the number of trees (typically up to 10,000) is shown to be beneficial to correctly discover the informative variables and to discard irrelevant ones. This result is consistent with the study of feature selection stability from RF proposed in [PVD12]. In general, enlarging the ensemble size naturally leads to increase the number of features that are deemed statistically significant. Beyond significance itself, it would also be interesting to study the effect size evaluated by such a statistical procedure. The Cramer’s V measure [Cra46] looks interesting in this regard.

Finally, this work show that measuring the distribution shift of class votes before and after permuting a feature in a tree ensemble conveys some useful information. The specific test to characterize such distribution shift is a bit less central. The $J_{\chi^2}$ test is convenient and appears to
be effective in practice, yet one could certainly design other procedures.

For instance, a Kolmogorov-Smirnov (KS) test offers a particular non-parametric alternative. The test statistic here relies on the distribution of the out-of-bag classification accuracies (or balanced classification rates averaging specificity and sensitivity) across the various trees in the ensemble. The effect on such distributions after permuting a specific variable is assessed. Our preliminary results along those lines show that the KS procedure offers very similar results to those of $J_{\chi^2}$, but at a higher computational cost.

In the same spirit, one could easily design further variants to focus on some specific function of the class confusion matrix and, for instance, to promote the selection of features that play a more critical role in the sensitivity of the classifier while putting less emphasis on the specificity as well.

Acknowledgements

We thank the anonymous reviewers for their fruitful comments. Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Equipements de Calcul Intensif en Fédération Wallonie Bruxelles (CECI) funded by the Fonds de la Recherche Scientifique de Belgique (FRS-FNRS).
Chapter C

Kernel methods for heterogeneous feature selection


Abstract

This paper introduces two feature selection methods to deal with heterogeneous data that include continuous and categorical variables. We propose to plug a dedicated kernel that handles both kind of variables into a Recursive Feature Elimination procedure using either a non-linear SVM or Multiple Kernel Learning. These methods are shown to offer state-of-the-art performances on a variety of high-dimensional classification tasks.

C.1 Introduction

Feature selection is an important preprocessing step in machine learning and data mining as increasingly more data are available and problems with hundreds or thousands of features have become common. Those high dimensional data appear in many areas such as gene expression array analysis, text processing of internet documents, economic forecasting, etc. Feature selection allows domain experts to interpret a decision model by reducing the number of variables to analyze. It also reduces
Appendix C. Kernel methods @ NEUCOMP 2015

training and classification times as well as measurement and storage requirements.

To the best of our knowledge, little effort has been dedicated to develop feature selection methods tailored for datasets with both categorical and numerical values. Such heterogeneous data are found in several applications. For instance, in the medical domain, high dimensional continuous feature sets (e.g., gene expression data) are typically considered along with a few clinical features. These features can be continuous (e.g., blood pressure) or categorical (e.g., sex, smoker vs non-smoker). To highlight important variables, a naive approach would transform heterogeneous data into either fully continuous or categorical variables before applying any standard feature selection algorithm. To get a continuous dataset, categorical variables can be encoded as numerical values. The specific choice of such numerical values is however arbitrary. It introduces an artificial order between the feature values and can lead to largely different distance measures between instances [DV11].

A standard approach relies on a multivariate numerical encoding, such as the disjunctive encoding, to represent categorical variables. For instance, a feature having 3 categories as possible values could be encoded by considering 3 new features instead: (1, 0, 0), (0, 1, 0) and (0, 0, 1). However, they need specific approaches, such as group lasso [YL06], to correctly handle feature selection at the granularity of the original features.

The discretization of continuous features is a common alternative to represent categorical and numerical features in a similar space. Such approach comes at the price of making the selection highly sensitive to the specific discretization [DV11].

A natural alternative would consider tree ensemble methods such as Random Forests (RF), since they can be grown from both types of variables and these methods perform an embedded selection. RF were however shown to bias the selection towards variables with many values [SBZH07]. The cForest method has been introduced to correct this bias [SBZH07] but its computational time is drastically increased and becomes prohibitive when dealing with thousands of features\footnote{In each node of each tree of the forest, a conditional independence permutation test needs to be performed to select the best variable instead of a simple Gini evaluation.}.

In this paper we propose two kernel based methods for feature selection. They are conceptually similar to disjunctive encoding while keeping original features throughout the whole selection process. In both approaches, the selection is performed by the Recursive Feature Elimination (RFE) [GWBV02] mechanism that iteratively ranks variables
C.2. Material and methods

This section presents the different building blocks that compose our two heterogeneous feature selection methods. Recursive Feature Elimination (RFE), the main feature selection mechanism, is presented in Section C.2.1. It internally uses a global variable ranking for both continuous and categorical features. This ranking is extracted from two kernel methods (Support Vector Machine and Multiple Kernel Learning) that use a dedicated heterogeneous kernel called the clinical kernel (Section C.2.2). Section C.2.3 details how to obtain a feature ranking from a non-linear SVM. Finally, Section C.2.4 sketches Multiple Kernel Learning, which offers an alternative way to rank variables with the clinical kernel.

C.2.1 Recursive feature elimination

RFE [GWBV02] is an embedded backward elimination strategy that iteratively builds a feature ranking by removing the least important features in a classification model at each step. Following [AHvdP+10], a fixed proportion of 20% of features is dropped at each iteration. The benefit of such a fixed proportion is that the actual number of features removed at each step gradually decreases till be rounded to 1, allowing a finer ranking for the most important features. This iterative process is pursued till all variables are ranked. The number of iterations automatically depends on the total number $p$ of features to be ranked while following this strategy. RFE is most commonly used in combination with a linear SVM from which feature weights are extracted. However, it can be used with any classification model from which individual feature importance can be deduced. A general pseudo-code for RFE is given in Algorithm C.1.
Algorithm C.1: Recursive Feature Elimination

\[
\begin{align*}
R &\leftarrow \text{empty ranking} \\
F &\leftarrow \text{set of all features} \\
\text{while } F \text{ is not empty do} &\quad \text{train a classifier } m \text{ using } F \\
&\quad \text{extract variable importances from } m \\
&\quad \text{remove the 20% least important features from } F \\
&\quad \text{put those features on top of } R \\
\text{end} \\
\text{return } R
\end{align*}
\]

C.2.2 Clinical kernel

The so-called clinical kernel proposed in [DDM09] was shown to outperform a linear kernel for classifying heterogeneous data. It averages univariate subkernels [STC04] defined for each feature.

\[
k(x_i, x_j) = \frac{1}{p} \sum_{f=1}^{p} k_f(x_{if}, x_{jf})
\]

\[
k_f(a, b) = \begin{cases} 
I(a = b) & \text{if } f \text{ is categorical} \\
\frac{(max_f - min_f) - |a - b|}{max_f - min_f} & \text{if } f \text{ is continuous}
\end{cases}
\]

where \(x_i\) is a data point in \(p\) dimensions, \(x_{if}\) is the value of \(x_i\) for feature \(f\), \(I\) is the indicator function, \(a\) and \(b\) are scalars and \(max_f\) and \(min_f\) are the maximum and minimum values observed for feature \(f\). One can note that summing kernels simply amounts to concatenating variables in the kernel induced space.

Given two data points, the subkernel values lie between 0, when the feature values are farthest apart, and 1 when they are identical, similarly to the gaussian kernel. The clinical kernel is basically an unweighted average of overlap kernels [VM05] for categorical features and triangular kernels [Gen02, BCR84] for continuous features. The overlap kernel can also be seen as a rescaled \(l_1\)-norm on a disjunctive encoding of the categorical variables. The clinical kernel assumes the same importance to each original variable. We show here the benefit of adapting this kernel for heterogeneous feature selection.

C.2.3 Feature importance from non-linear Support Vector Machines

The Support Vector Machine (SVM) [BGV92] is a well-known algorithm that is widely used to solve classification problems. It looks for
the largest margin hyperplane that distinguishes between samples of dif-
ferent classes. In the case of a linear SVM, one can measure the feature
importances by looking at their respective weights in the hyperplane.
When dealing with a non-linear SVM, we can instead look at the vari-
ation in margin size $\frac{1}{||w||}$. Since the larger the margin, the lower the
generalization error (at least in terms of bound), a feature that does
not decrease much the margin size is not deemed important for general-
ization purposes. So, in order to measure feature importances with a
non-linear SVM, one can look at the in
fluence on the margin of removing
a particular feature [Guy06].

The margin is inversely proportional to

$$W^2(\alpha) = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) = \|w\|^2 \quad (C.3)$$

where $\alpha_i$ and $\alpha_j$ are the dual variables of a SVM, $y_i$ and $y_j$ the labels
of $x_i$ and $x_j$, out of $n$ training examples, and $k$ a kernel. Therefore,
the importance of a particular feature $f$ can be approximated without
re-estimating $\alpha$ by the following formula:

$$J_{SV M}(f) = |W^2(\alpha) - W^2(-f)(\alpha)| \quad (C.4)$$

where $x_i^{-f}$ is the $i^{th}$ training example without considering the feature $f$.
In Equation (C.5), the $\alpha$’s are kept identical to those in Equation (C.3).
This is a computationally efficient approximation originally proposed
in [Guy06]. The feature importance is thus evaluated with respect to
the separating hyperplane in the current feature space and hence the
current decision function.

Updating $k(x_i, x_j)$ to $k(x_i^{-f}, x_j^{-f})$ is pretty efficient and straight-
forward with the clinical kernel (Section C.2.2). There is no need to
recompute the sum of all subkernels but one only has to remove $k_f$
(Equation (C.2)) and normalize accordingly. Removing one such sub-
kernel is equivalent to removing features in the projected space, which
is similar to what is done with a linear kernel.

In this work, we propose to combine the $J_{SV M}$ feature importance
(Equation (C.4)) with the RFE mechanism in order to provide a full
ranking of the features. This method will be referred to as $RFE^{SV M}$. 

C.2. Material and methods
C.2.4 Feature importance from Multiple Kernel Learning

MKL [LDBC+04] learns an appropriate linear combination of \( M \) basis kernels, each one possibly associated to a specific input variable, as well as a discriminant function. The resulting kernel is a weighted combination of different input kernels.

\[
k(x_i, x_j) = \sum_{m=1}^{M} \mu_m k_m(x_i, x_j) \quad \text{s.t. } \mu_m \geq 0 \quad (C.6)
\]

Summing kernels is equivalent to concatenating the respective feature maps \( \psi_1, \ldots, \psi_m \) induced by those kernels. The associated decision function \( f(x) \) is a generalized linear model in the induced space:

\[
f(x) = \sum_{m=1}^{M} \sqrt{\mu_m} w_m^T \psi_m(x) + b \quad (C.7)
\]

where \( \mu_m, w_m \) and \( \psi_m \) are respectively the kernel weight, feature weight and explicit feature map corresponding to the \( m \)-th kernel, and \( b \) a bias term. Those parameters are estimated by minimizing the following objective

\[
\arg\min_{w, b, \mu \geq 0} C \sum_{i=1}^{n} \ell(f(x_i), y_i) + \frac{1}{2} \sum_{m=1}^{M} \|w_m\|_2^2 \text{ such that } \|\mu\|_2^2 \leq 1 \quad (C.8)
\]

where \( C > 0 \) and \( \ell \) denotes the hinge loss \( \ell(f(x), y) = \max\{0, 1 - yf(x)\} \). We note that the kernel weight vector \( \mu \) is \( l_2 \)-regularized in contrast to MKL approaches using sparsity inducing norms [BLJ04]. Indeed, non-sparse MKL has been shown to be more effective on various computational biology problems [KBL+09]. It is also more convenient in our context since we interpret \( |\mu_m| \) as a feature importance measure and look for a full ranking of all features.

In this work, we adapt the clinical kernel (Equation (C.2)) with MKL to learn a non-uniform combination of the basis kernels, each one associated to a single feature. As we can see in Equation (C.7), \( \mu_f \) reflects the influence of kernel \( k_f \) in the decision function [LDBC+04]. \( \mu_f \) can thus be seen as the importance \( J_{MKL}(f) \) of feature \( f \).

The combination of RFE with this feature importance extracted from MKL will be referred to as \( RFE^{MKL} \). It specifically uses the kernel weights \( |\mu_f| \) as feature importance value to eliminate at each iteration a prescribed fraction of the least relevant features.
C.3 Competing approaches

This section presents the three competing methods we compare to in the experiments: Random Forest [Bre01] and two variants of Hybrid Feature Selection [DV11].

The Random Forest (RF) algorithm builds an ensemble of $T$ decision trees. Each one is grown on a bootstrap sample of the dataset. The subset of data points that are used to build a particular tree forms its bag. The remaining set of points is its out-of-bag. To compute variable importances, Breiman [Bre01] proposes a permutation test. It uses the out-of-bag samples to estimate how much the predictive performances of the RF decrease when permuting a particular variable. The bigger the drop in accuracy, the higher the variable importance. In order to obtain good and stable feature selection from RF, a large ensemble of 10,000 trees ($RF^{10000}$) is considered according to the analysis in [PVD12].

An alternative method performs a greedy forward selection aggregating separate rankings for each type of variables into a global ranking [DV11]. The authors report improved results over those of the method proposed in [HLY08], which is based on neighborhood relationships between heterogeneous samples. Out of a total of $p$ variables, categorical and continuous features are first ranked independently. Mutual information (MI) was originally proposed for those rankings but a reliable estimate of MI is difficult to obtain whenever fewer samples than dimensions are available. Instead we use the p-values of a t-test to rank continuous features and of a Fisher exact test for categorical ones. The two feature rankings are then combined into a global ranking by iteratively adding the first categorical or continuous variable that maximizes the predictive performance of a Naive Bayes or a 5-NN classifier (consistently with the choices made in [DV11]). The NN classifier uses the Heterogeneous Euclidian-Overlap Metric [WM97] between pairs of instances as follows:

\[
d(x_i, x_j) = \sqrt{\sum_{f=1}^{p} d_f(x_{if}, x_{jf})^2}
\]

(C.9)

\[
d_f(a, b) = \begin{cases} 
I(a \neq b) & \text{if } f \text{ is categorical} \\
\frac{|a-b|}{\text{max}_f-\text{min}_f} & \text{if } f \text{ is continuous}
\end{cases}
\]

(C.10)

\[
d_f(a, b) = 1 - k_f(a, b)
\]

(C.11)

This metric is closely related to the clinical kernel (Equation (C.2)). For each feature, $d_f$ takes value 0 for identical points and value 1 for points that are farthest apart in that dimension. We refer to these approaches
as $HFS^{NB}$ and $HFS^{5NN}$ in the sequel.

C.4 Experiments

In order to compare the five feature selection methods, we report predictive performances of classifiers built on selected variables as well as quality measures on those feature sets. A statistical analysis is also performed to assess if there are significant differences between the performances of the various methods. This section presents the experimental protocol, the various evaluation metrics and the datasets that we use in our experiments.

C.4.1 Experimental protocol

When a sufficient amount of data is available, 10-fold cross validation (10-CV) provides a reliable estimate of model performances [Koh95]. However, it may lead to inaccurate estimates on small-sized datasets, due to a higher variability in the different folds. We thus make use of a resampling strategy consisting of 200 random splits of the data into training (90%) and test (10%). Such a protocol has the same training/test proportions as 10-CV but benefits from a larger number of tests. It also keeps the training size sufficiently large so as to report performances close enough to those of a model estimated on the whole available data.

For each data partition, the training set is used to rank features and build predictive models using different numbers of features. The ranking is recorded and predictive performances are measured while classifying the test set. Average predictive performances are reported over all test folds and the stability of various signature sizes is computed from the 200 feature rankings. The average number of selected categorical features is also computed for each signature size. This number does not reflect a specific performance value of the feature selection methods but rather gives some insight into how they deal with the selection of heterogeneous variables.

Whenever a SVM is trained with the clinical kernel, the regularization parameter is fixed to a predefined value estimated from preliminary experiments on independent datasets. Such a value is set to 0.1 for the feature selection itself and to 10 when learning a final classifier on the selected features.
C.4. Experiments

C.4.2 Performance metrics

Predictive performances are reported here in terms of balanced classification rate (BCR), which is the average between sensitivity and specificity. These metrics are particularly popular in the medical domain and BCR, unlike AUC, easily generalizes to multi-class with unbalanced priors. For binary classification, it is defined as follows :

\[
BCR = \frac{1}{2} \left( \frac{TP}{P} + \frac{TN}{N} \right)
\]  
(C.12)

where \( TP \) (resp. \( TN \)) is the number of true positives (resp. negatives) and \( P \) (resp. \( N \)) the number of positive (resp. negative) samples in the dataset.

Selection stability is assessed here through the Kuncheva’s index (KI) [Kun07] which measures to which extent \( K \) sets of \( s \) selected features share common elements.

\[
KI(\{S_1, ..., S_K\}) = \frac{2}{K(K-1)} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \frac{|S_i \cap S_j| - s^2_p}{s - s^2_p}
\]  
(C.13)

where \( p \) is the total number of features and \( s^2_p \) is a correction for the random chance that 2 feature sets \( S_i \) and \( S_j \) share common features. KI takes values in \((-1, 1]\). A value of 0 indicates random selection. The larger KI, the larger the number of commonly selected features.

In order to globally compare the five feature selection methods, a Friedman statistical test [Dem06] is performed across all datasets and all feature set sizes. A low \( p \)-value indicates that there is indeed a difference between the various algorithm performances. In that case, a Nemenyi post-hoc test [Dem06] is performed to find out which methods perform significantly differently than others.

C.4.3 Datasets

We report results on 7 binary classification datasets briefly described in Table C.1 in terms of number of features and class priors. The Arrhythmia [FA10] dataset aims at distinguishing between the presence or absence of cardiac arrhythmia from features extracted from electrocardiograms. The Bands [FA10] dataset tackles the problem of band (grooves) detection on cylinders engraved by rotogravure printing. It consists of physical measurements and technical printing specifications. The task associated to the Heart [FA10] dataset is to detect the presence of a heart disease in the patient. Variables come from clinical measurements. The Hepatitis [FA10] dataset is about predicting survival
to hepatitis from clinical variables. The goal of the Housing [LD10] dataset is to evaluate the median value of owner-occupied homes from local statistics. The two classes are defined by a cutoff at $20,000. The Rheumagene [FHLD+11] dataset aims at diagnosing arthritis at a very early stage of the disease. Genomic variables are provided along with 3 clinical variables. Finally, the van’t Veer [vtVDvdV+02] dataset tackles a breast cancer prognosis problem. This very high dimensional dataset consists of genomic features from microarray analysis and seven clinical variables, two of them being categorical.

Table C.1: Datasets overview

<table>
<thead>
<tr>
<th>Name</th>
<th>Continuous features</th>
<th>Categorical features</th>
<th>Class priors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrhythmia [FA10]</td>
<td>198</td>
<td>64</td>
<td>245/185</td>
</tr>
<tr>
<td>Bands [FA10]</td>
<td>20</td>
<td>14</td>
<td>312/228</td>
</tr>
<tr>
<td>Heart [FA10]</td>
<td>6</td>
<td>7</td>
<td>164/139</td>
</tr>
<tr>
<td>Hepatitis [FA10]</td>
<td>6</td>
<td>13</td>
<td>32/123</td>
</tr>
<tr>
<td>Housing [LD10]</td>
<td>15</td>
<td>2</td>
<td>215/291</td>
</tr>
<tr>
<td>Rheumagene [FHLD+11]</td>
<td>100</td>
<td>3</td>
<td>28/21</td>
</tr>
<tr>
<td>van’t Veer [vtVDvdV+02]</td>
<td>4353</td>
<td>2</td>
<td>44/33</td>
</tr>
</tbody>
</table>

C.5 Results and discussion

We compare here $RFE^{MKL}$ and $RFE^{SVM}$ to $HFS^{NB}$, $HFS^{5NN}$ and $RF$ of 10,000 trees on 7 real-life datasets resulting in more than 7,000 experiments. These methods essentially provide a ranking of the features, without defining specific feature weights\(^2\). Predictive performances can then be assessed on a common basis for all techniques by selecting all features up to a prescribed rank and estimating a classifier restricted to those features. We use here a non-linear SVM with the clinical kernel reduced to the selected features as final classifier. Other final classifiers such as RF, Naive Bayes or 5-NN offer similar predictive performances and are not reported here.

We compare first all selection techniques across all feature set sizes and datasets to give a general view of the performances. Choosing a specific number of features is indeed often left to the final user who, for instance, might favor the greater interpretability of a reduced feature set at the price of some predictive performance decrease. Our second

\(^2\)Feature weights are used at each RFE iteration but those weights need not be comparable globally across iterations.
C.5. Results and discussion

analysis focuses on a fixed number of features offering a good trade-off between predictive performances and sparsity.

Figure C.1 reports the statistical analysis across all datasets and all feature set sizes using a Friedman test, followed by a Nemenyi post-hoc test. Figures C.2 – C.8 report more detailed results. They show the predictive performance, the stability of feature selection and the average number of selected categorical features on each signature size of each dataset.

The Friedman test [Dem06] can be seen as a non-parametric equivalent to the repeated-measures ANOVA. It tests whether the methods significantly differ based on their average ranks. In our experiments, it shows significant differences of the predictive performances of the 5 feature selection methods across all datasets and all feature set sizes ($p$-value < $10^{-6}$). According to the Nemenyi post-hoc test, (see Figure C.1, top), $RFE^{MKL}$ is best ranked (i.e. it has the lowest mean rank) and performs significantly better than $HFS^{SNN}$ and $RFE^{SVM}$ which appear at the end of the ranking. Our data does not show significant differences between the predictive performances of $RFE^{MKL}$, $RF10000$ and $HFS^{NB}$. A Friedman test on the feature selection stability also shows highly significant differences ($p$-value < $10^{-29}$) between the 5 feature selection approaches. According to a Nemenyi post-hoc test (see Figure C.1, bottom), our RFE approaches are at the bottom of the ranking. $RFE^{MKL}$ is however not significantly less stable than $HFS^{NB}$ and $RF10000$. In addition, the two $HFS$ approaches may have the natural advantage that they are based on filter methods that are more stable than embedded methods [HGV11]. Moreover, the RFs had to be run with a very large number of trees (10,000) to provide a stable feature selection [PVD12]. This leads to increased computational times and heavier models, especially on datasets with a higher number of instances. On the Arrhythmia and Bands datasets, the 200 resamplings require 1.5 more CPU time with RF10000 (single-core implementation in the randomForest R-package [LW02]) than with the RFE methods (in the Shogun [SRH+10] implementation of MKL and SVM). On the Housing dataset, the RF implementation is 5 times slower than the RFE methods.

The top graph of Figure C.2 shows predictive performances of the five methods on the Arrhythmia dataset. We can see that $RFE^{MKL}$ and $RF10000$ perform best since they avoid to select categorical features which happen to be noisy on this dataset. The bottom plot of

3Specifically, CPU times were measured on a 2.60 Ghz machine with 8GB Ram memory. On this dataset, $RFE^{MKL}$, $RFE^{SVM}$, and $RF10000$ took respectively 23 min, 26 min and 114 min to be run.
Figure C.1: Nemenyi critical difference diagrams [Dem06]: comparison of the predictive performances (BCR) and stability (KI) of the five algorithms over all signature sizes of all datasets. Horizontal black lines group together methods whose mean ranks do not differ significantly. \(\text{CD}\) represents the rank difference needed to have a 95% confidence that methods performances are significantly different.
Figure C.7 reports the average number of categorical features among selected features for the Rheumagene dataset. It shows that all but $RFE^{SVM}$ and $HFS^{5NN}$ select two categorical variables first, leading to already good predictive performances with very few selected variables (top graph of Figure C.7). The third categorical variable is actually never selected since it happens to convey very few information to predict the class label4. On the van’t Veer dataset, the $HFS$ approaches tend to keep selecting the two categorical variables even when the feature selection is very aggressive (Figure C.8, bottom). They show a peak in predictive performances when 5 features are kept (Figure C.8, top). However, the best predictive performance (Figure C.8, top) is obtained with $RFE^{MKL}$ which selects one of the two categorical variables. It also corresponds to a very good feature selection stability, as shown in the graph in the middle of Figure C.8. Finally, on the three high dimensional datasets (Arrhythmia, Rheumagene and van’t Veer), $RFE^{SV, M}$ is significantly less stable.

We further analyze below the various feature selection methods for a fixed number of selected features. One could indeed be interested in selecting a feature set as small as possible with only a marginal decrease in predictive performances. For each dataset, we choose the smaller feature set size such that the BCR of $RFE^{MKL}$ lies in the 95% confidence interval of the best $RFE^{MKL}$ predictive performance. Those signature sizes are highlighted in Figures C.2 – C.8 by vertical dashed lines. A Friedman test on those predictive performances finds significant differences ($p$-value of 0.008). A Nemenyi post-hoc test (Figure C.9, top) shows that the two best ranked methods, RF10000 and $RFE^{MKL}$, perform significantly better than $RFE^{SV, M}$ in terms of BCR. Feature selection stabilities also significantly differ according to a Friedman test ($p$-value of 0.02). Figure C.9 illustrates that the ranking among the five methods is the same for stability and BCR. Those results on a fixed number of features show that the $RFE^{MKL}$ and RF10000 are the two best performing methods without significant differences between them, but at a larger computational cost for the latter.

### C.6 Conclusion and perspectives

We introduce two heterogeneous feature selection techniques that can deal with continuous and categorical features. They combine Recursive Feature Elimination with variable importances extracted from MKL.

---

4Out of 49 samples (28 negative, 21 positive), this variable takes value ‘0’ 46 times and ‘1’ only 3 times.
Figure C.2: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the Arrhythmia dataset. The dashline defines the minimal number of features to select without losing much in predictive performances (see text).
Figure C.3: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the Bands dataset. The dashline defines the minimal number of features to select without loosing much in predictive performances (see text).
Figure C.4: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the Heart dataset. The dashline defines the minimal number of features to select without loosing much in predictive performances (see text).
Figure C.5: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the Hepatitis dataset. The dashline defines the minimal number of features to select without losing much in predictive performances (see text).
Figure C.6: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the Housing dataset. The dashline defines the minimal number of features to select without losing much in predictive performances (see text).
C.6. Conclusion and perspectives

Figure C.7: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the Rheumagene dataset. The dashline defines the minimal number of features to select without losing much in predictive performances (see text).
Figure C.8: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the van’t Veer dataset. The dashline defines the minimal number of features to select without losing much in predictive performances (see text).
Figure C.9: Nemenyi critical difference diagrams [Dem06]: comparison of the predictive performances (BCR) and stability (KI) of the five algorithms for one small signature size in each dataset. Horizontal black lines group together methods whose mean ranks do not differ significantly. CD represents the rank difference needed to have a 95% confidence that methods performances are significantly different.
(RFE_{MKL}) or a non-linear SVM (RFE_{SVM}). These methods use a dedicated kernel combining continuous and categorical variables. Experiments show that RFE_{MKL} produces state-of-the-art predictive performances and is as good as competing methods in terms of feature selection stability. It offers results similar to Random Forests with smaller computational times. RFE_{SVM} performs worse than RFE_{MKL}. It also seems less efficient in terms of prediction and stability than competing approaches, even though not significantly different from all competitors.

The two kernel based methods proposed here are among the few existing selection methods that specifically tackle heterogeneous features. Yet, we plan in our future work to improve their stability possibly by resorting to an ensemble procedure [AHVdP+10].

We observed that the proposed methods run faster than the competing approaches on various datasets. Those differences would be worth to reassess in a further study relying on parallel implementations.

Acknowledgements

Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Equipements de Calcul Intensif en Fédération Wallonie Bruxelles (CECI) funded by the Fonds de la Recherche Scientifique de Belgique (FRS-FNRS).
Part IV

Side notes
Chapter D

About stability

In the experiments and papers published during this thesis, we measure feature selection stability with Kuncheva’s index (KI) [Kun07] (see Section 4.2). We chose KI for two reasons. Firstly, we always compare feature sets of the same size, which is required to use KI. Secondly, KI has the great advantage of being calibrated with respect to the chance of randomly selecting common features (in this case KI = 0).

There are, of course, other stability indices. In this chapter, we present two additional such indices and highlight some limitations of measuring stability in situations where there are redundant feature sets in the data.

D.1 Two alternative stability indices

Jaccard’s index (JI) [Jac12] is similar to KI. It measures how much $K$ feature sets (or signatures) share common elements. Compared to KI, JI has the advantage of being defined for sets of different sizes. However, it does not include any correction for the chance of randomly selecting common features. This index is defined as follows:

$$JI({F_1, \ldots ,F_K}) = \frac{2}{{K(K - 1)}} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \frac{{|F_i \cap F_j|}}{{|F_i \cup F_j|}}.$$  \hspace{1cm} (D.1)

When all the feature sets $F_i$ have the same size $s$, we can rewrite this index as

$$JI({F_1, \ldots ,F_K}) = \frac{2}{{K(K - 1)}} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \frac{{|F_i \cap F_j|}}{{2s - |F_i \cap F_j|}}.$$  \hspace{1cm} (D.2)
Since $|F_i \cap F_j| \in [0, s]$, this function is monotonically increasing with the sizes of the intersections $|F_i \cap F_j|$. It is therefore expected to behave similarly to $K_I$ (see Equation 4.3, page 60) except that it is not corrected with respect to the random chance of the $F_i$’s to share common features. Indeed, when one selects all $s = p$ features, $J_I$ reaches its maximal value of 1, while $K_I$ would have a value of 0 which corresponds to random selection.

Haibe-Kains’ index ($HK_I$) [HK09] adopts a relatively different approach to compute feature selection stability. It compares $K$ sets of $s$ selected features and focuses on the frequency of the top $s$ best variables. $HK_I$ measures the average frequency of the $s$ features that appear the most often in the $K$ signatures. The mathematical definition of this index is given here:

$$HKI_{uncor}(\{F_1, \ldots, F_K\}) = \frac{\sum_{j=1}^{s} \text{frequency}(x_{(j)})}{sK}, \quad (D.3)$$

where $\text{frequency}(x_{(j)})$ computes the number of signatures in which variable $x_j$ appears and $x_{(1)}, \ldots, x_{(p)}$ are the features $x_1, \ldots, x_p$ sorted by decreasing order of frequency. This index is also expected to be coherent with $K_I$. Indeed the number of common features among the pairs of sets $F_i, F_j$ is proportional to the selection frequencies of the variables. Since $HKI_{uncor}$ can be made arbitrarily large by increasing $s$, a penalty proportional to the signature size is included in the following corrected version:

$$HKI(\{F_1, \ldots, F_K\}) = \max \left( 0, HKI_{uncor}(\{F_1, \ldots, F_K\}) - \frac{s}{p} \right), \quad (D.4)$$

where $p$ is the total number of variables. Similarly to $K_I$, the stability of always selecting all features is 0.

### D.2 Stability with redundant feature sets

In our contributions about kernel methods for heterogeneous feature selection (see Chapter C), we formulate the hypothesis that recursive feature elimination procedures (RFE) (cf. Section 3.3.5), which are multivariate feature selection approaches, may inherently suffer from lower stability compared to univariate methods, when there are redundant feature sets in a dataset. Indeed, if the same signal is contained in different sets of variables, a multivariate approach would sometimes favour one signature over the other. In the other hand, univariate methods that do not take feature interactions into account would tend to produce more stable rankings.
In order to further investigate this hypothesis, we use artificial datasets to compare the stability of a Welch’s t-test filter with the stability of a standard RFE combined with a linear SVM. The first method is univariate while the second one considers variable interactions.

Artificial datasets of \( n = 500 \) samples and \( p = 50 \) variables are designed to contain different groups of features that encode the exact same class information. To do so, a first group of five features is generated in the following way. A matrix \( X \in \mathbb{R}^{500 \times 5} \) is sampled from a \( N(0, 1) \) and a weight vector \( w \) of length 5 is sampled uniformly in \([-1, -0.5] \cup [0.5, 1]\]. This first group of five features defines the labels by \( y = \text{sign}(Xw) \). For each other group of variables, a new matrix \( X' \in \mathbb{R}^{500 \times 5} \) and a new weight vector \( w' \) are sampled in the same way. Then, for each \( x'_i \) where \( \text{sign}(\langle x'_i, w' \rangle) \neq y_i \), we replace \( x'_i \) by its opposite \(-x'_i\). This ensures that the different groups of features generate the same labels with their respective linear predictor:

\[
\text{sign}(Xw) = \text{sign}(X'w') = y. \tag{D.5}
\]

The final dataset is then defined as the concatenation of the different sets of 5 variables and \( y \) is the vector of class labels.

We report here results obtained from 200-resamplings on two datasets. The first one contains signal in only one group of 5 features. The second one has 3 informative groups of 5 variables. The remaining 45 and 35 features contain random noise sampled from \( N(0, 1) \). The predictions are produced by a random forest (RF) of 500 trees trained on the selected features.

First of all, as we can see on Figure D.1, the predictive performances of a RF built on the features selected from the \( t \)-test and the RFE procedure are very similar. This indicates that the selected features are equally good for prediction purposes whatever the selection method. However, the two feature selection methods have a very different behaviour from a stability point of view. As we can see on top of Figures D.2, D.3 and D.4 where there is only one group of 5 relevant features, the two approaches perform similarly for small feature sets (with a slight advantage for the multivariate one) till they reach the optimal signature size of 5. But, when there are 3 groups of 5 relevant features, there is a flagrant difference between the stability of the \( t \)-test and RFE. The first one remains very stable even for a signature size of 5, while the multivariate approach drops considerably. As the predictive performances are very comparable, this indicates that RFE indeed oscillate between different groups of relevant features which yields a smaller stability.
Appendix D. About stability

Figure D.1: Average predictive performances w.r.t. the number of selected variables on 200 resamplings. Top: only one informative group of 5 features (vertical line at $s = 5$). Bottom: three informative groups of 5 features each (vertical line at $s = 15$).
D.2. Stability with redundant feature sets

Figure D.2: Stability measured with Kuncheva’s index w.r.t. the number of selected variables on 200 resamplings. Top: only one informative group of 5 features (vertical line at $s = 5$). Bottom: three informative groups of 5 features each (vertical line at $s = 15$).
Figure D.3: Stability measured with Jaccard’s index w.r.t. the number of selected variables on 200 resamplings. Top: only one informative group of 5 features (vertical line at $s = 5$). Bottom: three informative groups of 5 features each (vertical line at $s = 15$).
D.2. Stability with redundant feature sets

Figure D.4: Stability measured with Haibe-Kains’ index w.r.t. the number of selected variables on 200 resamplings. Top: only one informative group of 5 features (vertical line at $s = 5$). Bottom: three informative groups of 5 features each (vertical line at $s = 15$).
As expected, the three stability indices perform similarly. The only very important difference is that JI rises up to one when all features are selected.\footnote{This is the case when all 50 features are selected. The maximal value is a bit lower on the graphs because the biggest tested signature size is \( s = 48 \).} This is not very convenient since it does not reflect the quality of such selection. One must be very careful when interpreting JI on big signature sizes.
Chapter E

RFE with the disjunctive encoding

In [PDD15], we introduce two feature selection methods, RFE\textsuperscript{MKL} and RFE\textsuperscript{SVM}, based on recursive feature elimination (RFE) with a non-linear kernel (see Chapter 7). Yet, the standard RFE [GWBV02] makes use of a linear SVM (see Section 3.3.5). Whereas it does not naturally handle categorical variables, we can nevertheless use the linear SVM on such data, provided that we recode categorical features into continuous ones. This chapter expands the analysis of [PDD15] (see Chapter C) by comparing the proposed methods to an additional baseline: the standard RFE with a data recoding.

In order to obtain a fully continuous dataset, we represent categorical variables with a disjunctive encoding (see Section 3.3.1). Each binary feature is encoded as a numerical variable and the two values are mapped to 0 and 1. Features with \( C > 2 \) categories are encoded as \( C \) numerical features that represent the membership to each category. For instance a feature with 3 categories is encoded as 3 fresh variables. Value (1, 0, 0) represents the first category, (0, 1, 0) the second one and (0, 0, 1) the last one.

The following section presents an update of the results of [PDD15]. They additionally include the performances of a linear RFE to rank features and a linear SVM\textsuperscript{1} to perform classification. So we compare RFE\textsuperscript{MKL} and RFE\textsuperscript{SVM} with RF of 10,000 trees (see Section 3.3.3), HFS\textsuperscript{NB}, HFS\textsuperscript{SNN} (see Section 3.3.4) and linear RFE on 7 real-life datasets. The experimental setting is exactly the same as in [PDD15].

\textsuperscript{1}The cost meta-parameter is set to the default value of 1, both for the prediction and in the RFE procedure.
However, because of the disjunctive encoding, the number of dimensions of 3 datasets is increased. Indeed, Bands [FA10], Housing [LD10] and Heart [FA10] (see Chapter F) contain categorical features with more than two levels. This change of dimensionality is particularly spectacular with the Housing dataset because one of its categorical variables is expanded into 92 continuous attributes, making its number of features increase from 17 to 108.

E.1 Results

We compare the balanced classification rate and feature selection stability across all feature set sizes and all datasets to give a global view of the performances. We also report the average number of selected categorical variables for each signature size of each dataset. Whenever a disjunctive encoding is applied, we consider all the new continuous features as categorical ones in this computation.

A Friedman test [Dem06] highlights significant differences (p-value < $10^{-15}$) between the predictive performances of the 6 methods across all feature set sizes of all datasets. On top of Figure E.1, the Nemenyi post-hoc test shows that RFE with a disjunctive encoding (referred to as $RFE + recoding$) performs worse in terms of predictive performances although it does not differ significantly from $HFS^{SNN}$ and $RFE^{SVM}$.

All the other methods are ranked in the same way as in the results presented in the paper, $RFE^{MKL}$ being the best performing one. Figures E.2 – E.9 present detailed results for each dataset. They show that for all datasets but Hepatitis, $RFE^{MKL}$ has nearly always better predictive performances than the standard RFE with the disjunctive encoding. This is particularly striking on the high-dimensional datasets.

A Friedman test also points out some significant differences (p-value < $10^{-50}$) regarding the stability of feature selection. The bottom plot of Figure E.1 shows the outcome of the Nemenyi post-hoc test. Similarly to the published results, RF, the $HFS$ methods and $RFE^{MKL}$ appear at the top of the ranking and their performances are not significantly different. $RFE$ with a linear SVM using a disjunctive encoding presents a stability that is significantly worse than all the other approaches. Looking at the plots in the middle of Figures E.2 – E.9, we can see that $RFE + recoding$ is particularly less stable than the other approaches on the high-dimensional datasets.

We now turn our attention to the number of selected categorical features for each signature size (bottom plots of Figures E.2 – E.9). We can observe on the Arrhythmia, Housing, Rheumagene and van’t Veer
datasets that \( RFE + \text{recoding} \) tends to favour truly continuous features more than most of the other methods. It selects the disjunctive encoded variables later, for larger feature sets.

Finally, we get back on the initial motivation for not using the disjunctive encoding in our previous analyses. As explained in Section 3.3.1, this encoding changes the granularity of feature selection. Indeed, one categorical variable is transformed into several continuous features. Therefore, the same original variable can be assigned very distinct ranks for each of its continuous surrogates. The question of whether it is actually a problem is arguable. Nonetheless, we confirm that this occurs in the reported experiments. For instance, in a particular resampling on the Bands dataset — which is about the quality control of a mechanical process (see Chapter F) —, the categorical feature that represents distinct production units has ranks 3, 15, 20, 31, 47 and 52 out of a total of 62 variables after recoding. Selecting the original variable based on its ranks in the disjunctive encoding would then be quite difficult due to the large scattering of those ranks.

In a nutshell, the standard re-encoding of categorical features into continuous variables performs both worse in terms of prediction performance and stability as compared to existing alternatives including the novel ones proposed in this thesis.
Figure E.1: Nemenyi critical difference diagrams [Dem06]: comparison of the predictive performances (BCR) and stability (KI) of the five algorithms over all signature sizes of all datasets. Horizontal black lines group together methods whose mean ranks do not differ significantly. CD represents the rank difference needed to have a 95% confidence that methods performances are significantly different.
E.1. Results

Figure E.2: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the Arrhythmia dataset.
Appendix E. RFE with the disjunctive encoding

Figure E.3: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the Bands dataset.
Figure E.4: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the Heart dataset.
Appendix E. RFE with the disjunctive encoding

Figure E.5: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the Hepatitis dataset.
Figure E.6: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the Housing dataset. To enhance readability, a zoom on the vertical axis is provided in Figure E.7 for the bottom plot.
Figure E.7: Zoom on the number of selected categorical features for each signature size of the Housing dataset
E.1. Results

Figure E.8: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the Rheumagene dataset.
Figure E.9: Predictive performances (BCR), feature selection stability (KI) and number of selected categorical features for each signature size of the van’t Veer dataset.
Chapter F

Datasets

This chapter gives a brief overview of the various datasets used in the experiments. Table F.1 summarises the datasets in terms of dimensionality and class prior. The remainder of the chapter gives some information about the purpose of each dataset.

Genomic datasets

- Alon [ABN+99] : discriminate between normal and colon tumour tissues.
- DLBCL [SRT+02] : predict the outcome of diffuse large b-cell lymphoma.
- Golub [GST+99] : identify different types of cancer.
- Lymphoma [AED+00] : identify different lymphoma subtypes.
- Prostate [SFR+02] : diagnostic of prostate cancer.

The Alon, DLBCL, Golub and Prostate datasets are available as pre-processed RData files at

https://github.com/ramhiser/datamicroarray

Lymphoma is available at

http://eps.upo.es/aguilar/datasets.html

Genomic data + clinical variables

Appendix F. Datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>cont. feat.</th>
<th>cat. feat.</th>
<th>class priors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alon [ABN+99]</td>
<td>2000</td>
<td>0</td>
<td>40/22</td>
</tr>
<tr>
<td>DLBCL [SRT+02]</td>
<td>7129</td>
<td>0</td>
<td>58/19</td>
</tr>
<tr>
<td>Golub [GST+99]</td>
<td>7129</td>
<td>0</td>
<td>25/47</td>
</tr>
<tr>
<td>Lymphoma [AED+00]</td>
<td>4026</td>
<td>0</td>
<td>22/23</td>
</tr>
<tr>
<td>Prostate [SFR+02]</td>
<td>6033</td>
<td>0</td>
<td>52/50</td>
</tr>
<tr>
<td>Rheumagene [LHLG+15]</td>
<td>100</td>
<td>3</td>
<td>28/21</td>
</tr>
<tr>
<td>van’t Veer [vtVDvdV+02]</td>
<td>4353</td>
<td>2</td>
<td>44/33</td>
</tr>
<tr>
<td>Breast tissue [FA10]</td>
<td>9</td>
<td>0</td>
<td>22/21/14/49</td>
</tr>
<tr>
<td>Glass [FA10]</td>
<td>9</td>
<td>0</td>
<td>70/76/17/13/9/29</td>
</tr>
<tr>
<td>Musk1 [FA10]</td>
<td>166</td>
<td>0</td>
<td>269/207</td>
</tr>
<tr>
<td>Vehicle [FA10]</td>
<td>18</td>
<td>0</td>
<td>218/212/217/199</td>
</tr>
<tr>
<td>Wine [FA10]</td>
<td>13</td>
<td>0</td>
<td>59/71/48</td>
</tr>
<tr>
<td>Arrhythmia [FA10]</td>
<td>198</td>
<td>64</td>
<td>245/185</td>
</tr>
<tr>
<td>Bands [FA10]</td>
<td>20</td>
<td>14</td>
<td>312/228</td>
</tr>
<tr>
<td>Heart [FA10]</td>
<td>6</td>
<td>7</td>
<td>164/139</td>
</tr>
<tr>
<td>Hepatitis [FA10]</td>
<td>6</td>
<td>13</td>
<td>32/123</td>
</tr>
<tr>
<td>Housing [LD10]</td>
<td>15</td>
<td>2</td>
<td>215/291</td>
</tr>
</tbody>
</table>

Table F.1: Datasets overview

- van’t Veer [vtVDvdV+02]: breast cancer prognosis.

The Rheumagene dataset is not publicly available. It is part of a project in which the machine learning group of the UCL took part. The van’t Veer dataset is available here:

http://ccb.nki.nl/data/

Other datasets with continuous variables only

- Breast tissue [FA10]: prediction of tissue type from impedance measurements.
- Glass [FA10]: classification of glass fragments into seven different types using proportions of chemical elements that compose each fragment.
- Musk1 [FA10]: identification of musk molecules from shape and conformation measurements.
- Vehicle [FA10]: distinction between four vehicle types given some geometrical features extracted from their silhouettes.
• The Wine [FA10] : identification of the production domain from chemical measurements of different wines.

Those datasets are available on the UCI repository

http://archive.ics.uci.edu/ml

Other heterogeneous datasets

• Arrhythmia [FA10] : distinction between the presence or absence of cardiac arrhythmia from electrocardiograms.

• Bands [FA10] : band (grooves) detection on cylinders engraved by rotogravure printing. The features are physical measurements and technical printing specifications.

• Heart [FA10] : heart disease detection from clinical measurements.

• Hepatitis [FA10] : prediction of survival to hepatitis from clinical variables.

• Housing [LD10] : evaluation of the median value of owner-occupied homes from local statistics. The two classes are defined by a cutoff at $20,000.

Those datasets are available on the UCI repository

http://archive.ics.uci.edu/ml

and in the mlbench R package

http://cran.r-project.org/web/packages/mlbench/
References


[AM02] Christophe Ambroise and Geoffrey J. McLachlan. Selection bias in gene extraction on the basis of microar-


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


[SSDB95] Mark Schena, Dari Shalon, Ronald W. Davis, and Patrick O. Brown. Quantitative monitoring of gene ex-


REFERENCES

