University of Cagliari

Doctoral Thesis

Dataset Analysis for Classifier Ensemble Enhancement

Author:
Emanuele Tamponi

Supervisor:
Prof. Giuliano Armano

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in

Electronic and Computer Engineering
ING-INF/05

April 2015

All that is gold does not glitter,
Not all those who wander are lost;
The old that is strong does not wither,
Deep roots are not reached by the frost.

From the ashes a fire shall be woken,
A light from the shadows shall spring;
Renewed shall be blade that was broken,
The crownless again shall be king.

J.R.R. Tolkien

Jesus knew that his hour was come
that he should depart out of this world
unto the Father, having loved his own
which were in the world,
he loved them unto the end.

John 13,1

UNIVERSITY OF CAGLIARI

Abstract

Faculty of Engineering and Architecture
Department of Electrical and Electronic Engineering

Doctor of Philosophy

Dataset Analysis for
Classifier Ensemble Enhancement

by Emanuele TAMPONI

We developed three different methods for dataset analysis and ensemble enhancement. They share the underlying idea that an accurate preprocessing and adaptation of the data can improve the system performance, without changing the classification model. *Correlation Score* is a generic framework for assessing encoding techniques by measuring the correlation between the encoded feature vectors and the corresponding class labels; experiments show its effectiveness in discovering the best encoding configurations between those tested, on a wide range of classification domains. *Multi-Resolution Complexity Analysis* is a method for assessing the *local complexity* inside a given domain. It is able to split a domain into regions of different classification complexity, giving insights on the inner structure of the populations inside the domain. Finally, *Forests of Local Trees* are a novel training algorithm for ensemble classifiers. They are based on the concept of *local trees*: classifiers trained with a bias toward a certain region of the domain. This bias enhances the diversity inside the ensemble, leading to improved performance.

These three topics are meant as a foundation for a more complex framework, that will eventually utilize them organically.

Emanuele Tamponi gratefully acknowledges Sardinia Regional Government for the financial support of her PhD scholarship (P.O.R. Sardegna F.S.E. Operational Programme of the Autonomous Region of Sardinia, European Social Fund 2007-2013 – Axis IV Human Resources, Objective l.3, Line of Activity l.3.1.).

Acknowledgements

Acknowledging all the people that helped me through this work is impossible. I will try to do my best, but please, don't feel angry if your name is not in this list! And please consider that I've wrote everything in random order!

I wish to thank my advisor, prof. Giuliano Armano, for having mentored me for a grand total of three different theses, and having always believed in my capabilities.

Thank you to all my colleagues, in particular Alessandro(s), Francesca, Matteo(s), Amir, Mario and Cristina, for the endless hours of work and laugh, and for having stood me for so many years...

Thank you to Paolo, Luigi, Luigi, Silvia, Miriam, Tore, Roberto, Valeria, Andrea, Basti, Michele, and all other friends of Communion and Liberation, for having always sustained me to live the Ideal in the reality.

Thank you to Giulia, Matteo, and Salvatore, for being great friends.

Thank you to Chiara, because you are here and show me that what changes my life is not my projects on things, but His project on me.

Thank you to Don Felice, as without him I couldn't understand anything of what I live.

Thank you, Antonio and Alberto, for always being present, in some way, just as only brothers can be.

Thank you, mamma and babbo, because you are the best parents anyone can hope to have!

Contents

List of Figures

List of Tables

A Mamma e Babbo!

Chapter 1

Introduction

In the last few years, pattern recognition and machine learning systems have become ubiquitous. They are used from industrial infrastructure to everyday products, like mobile phones and cars. However, they are far from being a crystallized research field. On the contrary, every day sees the light of new classification or regression algorithms, data mining tools, application fields, theoretical insights.

The focus of this thesis is on *classification systems*, and in particular, we provide novel contributions on *dataset analysis* techniques. The red line that connects the remaining three chapters of this manuscript (each of which represents the result of one year of research of our PhD) is our perception that the performance of a classification system can be widely improved by just choosing the best preprocessing steps and tuning them optimally, without any modification of the classification model being used. This approach is surely an hazard, as most of the research effort on the field of classification systems goes to defining novel, more accurate models for classification (we cite, for example, the work on Conditional Random Fields, on Mixtures-of-Experts, and on Gaussian Mixtures), that consider the underlying data as "immutable".

With this guiding principle in mind, we developed three different dataset analysis approaches.

- *Correlation Score.* Chapter 2 presents Correlation Scores, a novel method to assess encoding techniques. Defining the optimal description of an object (its encoding) is a complex task, typically done manually by an expert of the field, except for few cases like feature selection or discretization: in these cases,

several algorithms exist, capable of searching through the configuration space in order to find an optimal (or nearly optimal) configuration. Our method can instead be applied to *any* kind of data preprocessing for classification tasks, and works by computing the overall correlation between the encoded data (input) and the labels (output). It provides a "score", that can then be used to rank the configurations: we show that this score is strongly correlated to the error rate of a classification system trained using the same encoded data. Moreover, we show that our Correlation Scores can outperform information-based approaches. One obvious application of our Correlation Scores is its use as a *fitness function* in genetic optimization algorithms and other similar search methods, when there is no obvious way to compute the derivative between the fitness function and the parameters to optimize.

- *Multi-Resolution Complexity Analysis.* Chapter 3 discusses our novel method for measuring the complexity of a given domain. Once again, we moved away from the main stream of research work, where complexity analysis may refer to either the definition of measures able of ranking different classification domains from the easiest to the hardest (e.g.: the *Bayes error rate*), or to the definition of a set of measures *related to the complexity* of a given domain, aimed at highlighting *why* it is complex (e.g.: non-linearity of boundary between classes, sparseness of the input space, et cetera). In either cases, such measures are *global* characteristics of a domain. Instead, our purpose is to measure the *local* complexity inside a given domain, in order of splitting it in regions of different classification complexity. In order to do that, we developed *Multi-Resolution Complexity Analysis* ($MRCA$ for short). $MRCA$ works by evaluating the complexity of each point in a dataset at different resolutions (hence the name): this gives rise to a *complexity profile space*, in which is possible to cluster observations with similar complexity.

- *Forests of Local Trees.* Chapter 4 presents a novel ensemble algorithm developed by us, based on Random Forest and on the concept of *local trees*. A local tree is a decision tree trained to give more importance to a specific region of the domain. By training local trees, we increase the diversity of the ensemble, without impairing the performance of the single learners. We show that our Forests of Local Trees (FLTs for short) compare favorably to state of the art algorithms like Random Forests, even for small ensemble sizes.

The topics that we discuss in this thesis are meant to be a foundation for a more complex framework. For example, we expect to improve the performance of our Forests of Local Trees by using Multi-Resolution Analysis to place the local trees in optimal regions inside the datasets. On the other side, $MRCA$ can be used together with the Correlation Scores (through the definition of a new *probe function*, described in Chapter 3). However, these interactions are not discussed in the present manuscript.

The reader will surely notice how the maturity of the researcher grows chapter after chapter. In effect, each topic represents one year of research, and we decided to present the results in strictly increasing temporal order. Personally, we are proud of the results achieved each year, in particular the ones obtained by our Forests of Local Trees.

The rest of this Introduction presents the notation shared by each chapter, that is then assumed as known in the remaining of the manuscript. We also briefly list and describe the datasets used in the experiments.

1.1 Notation and Datasets

We assume that the reader is familiar with the field of pattern recognition and in particular of classification algorithms.

A dataset $\mathcal{D}$ is a set of N object-label pairs. It is a sample from the underlying population domain. The dataset can be defined in two ways. The first one doesn't assume anything about the representation of the objects, and we indicate it simply as $\mathcal{D} = \{(\boldsymbol{o}_1, y_1), \ldots, (\boldsymbol{o}_N, y_N)\}$, where $\boldsymbol{o}_i$ is the i-th object in the sample, and y_i is the *class label* associated to it (we can use also the expression *raw* dataset to refer to this one). The second definition of dataset assumes that the objects have already been encoded in some way, so that they are represented through a set of measures, or *features*, that form a *feature vector* for the object; in this case, we write $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, where $\boldsymbol{x}_i$ is the feature vector associated to the i-th object in the sample. In either cases, $y \in \mathcal{Y}$, where $\mathcal{Y}$ is a finite set of class labels or categories (e.g., it may indicate the state of an instrument: $\mathcal{Y} = \{\text{working}, \text{damaged}\}$; or a kind of flower: $\mathcal{Y} = \{\text{iris-virginica}, \text{iris-setosa}, \text{iris-versicolor}\}$); the number of class labels is indicated as m. The feature vector may contain either *continuous* or *categorical* features,

Dataset name	Classes m	Objects N	Scalar features	Discrete features
anneal	6	898	6	32
anneal-orig	6	898	6	32
audiology	24	226	69	0
autos	7	205	10	16
balance-scale	3	625	4	0
breast-cancer	2	286	0	10
breast-w	2	699	9	0
colic	2	368	7	16
colic-orig	2	368	7	16
credit-a	2	690	6	9
credit-g	2	1000	7	13
diabetes	2	768	8	0
glass	7	214	9	0
heart-c	5	307	6	7
heart-h	5	294	6	7
heart-statlog	2	270	13	0
hepatitis	2	155	6	13
hypothyroid	4	3772	7	22

TABLE 1.1: Datasets used in the experiments.

Dataset name	Classes m	Objects N	Scalar features	Discrete features
ionosphere	2	351	34	0
iris	2	150	4	0
kr-vs-kp	2	3197	0	36
labor	2	57	8	8
letter	26	20000	16	0
lymph	4	148	3	15
primary-tumor	22	239	0	17
rootstock	6	48	4	0
segment	7	2310	19	0
sick	2	3772	7	23
sonar	2	208	60	0
soybean	19	683	0	35
splice	3	3190	0	60
vehicle	4	846	18	0
vote	2	435	0	16
vowel	11	990	10	3
waveform	3	5000	40	0
zoo	7	101	2	16

TABLE 1.2: Datasets used in the experiments (*cont.d*).

and we indicate the feature space as $\mathcal{X}$, so that $\boldsymbol{x} \in \mathcal{X}$. The number of components of $\boldsymbol{x}$ is indicated as n and each feature is x_j, so that $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$.

A classifier is an algorithm that associates an *unlabeled* feature vector to a label. It is generally *trained* by an algorithm that takes a *learning dataset* as input and produces a classifier *fitted* in order to associate the optimal label to each unlabeled instance. Optimality is generally defined as minimizing the *error rate* on a *test dataset* that is assumed unknown during the training phase.

Tables 1.1 and 1.2 shows the datasets used in the experiments. Clearly, they are not raw datasets, as they come with a set of precomputed features. They are part of the UCI dataset repository, and are used for most experimental comparisons in the literature [1].

Chapter 2

Assessing Encoding Techniques through Correlation Scores

Any classification system can be decomposed into two parts, as depicted in Figure 2.1. The decomposition does not consider the training process, but only the "steady state" of the system.

The "classifier" block represents the classification model. Current research on classification algorithms attempts to either provide models specialized on specific domains, or general purpose classifiers that fit well on a wide variety of applications. Together with the models, the literature provides plenty of information on how to train them.

The "encoder" block is the necessary connection between the data that comes from the real world and the classification algorithm. With the terms "encoding", "encoding technique" and "encoder", we refer to all the types of preprocessing techniques that can be used to provide the classifier with a properly formatted and enhanced version of the data fetched from the domain at hand. For example, we could refer to image enhancement, feature extraction, normalization and reduction,

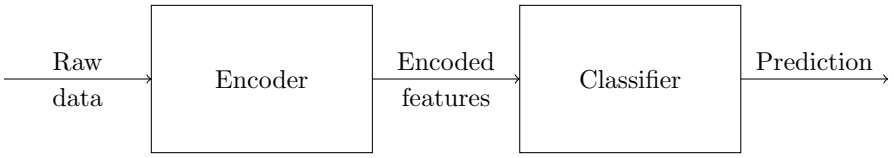


FIGURE 2.1: Decomposition of a classification system.

data augmentation, de-noising, et cetera, or even domain-specific techniques like amino acid encodings [2], [3], [4].

The performance of a classification system greatly depends on the selected encoding technique and on its parameters. However, it is typically impossible to express mathematically the connection between the parameters of an encoder and the performance of the classification system in which it will be used[1], so that the encoder parameters cannot be optimized with derivative-based methods like gradient descent. For this reason, optimization has to be done using brute-force search techniques, like grid-search, or using genetic algorithms. However, both optimization systems are only viable when an effective and fast *fitness function* is available. A fitness function is a method for assessing the quality of each candidate element found during the search, so that the one with the greatest fitness is selected as the optimal one.

To our best knowledge, such fitness functions have been defined only for a few types of encodings, e.g. for categorical feature selection (see, for example, [5], [6], [7]). Most of these functions are based on the concept of mutual information (shortly discussed in Sec. 2.4) and are typically presented as part of an optimization algorithm, so that it is difficult to consider them independently from the overall system.

All other types of encoding techniques lack a fitness function capable of assessing them in isolation from the rest of the classification system. In fact, as of now, evaluating the impact of an encoding technique on a classification system typically requires to train it end-to-end and test it by means of a performance metric deemed relevant (e.g., the classification error). This leads to the following typical strategy for encoder parameters optimization: from a set of "candidate" encoders, the one that optimizes the performance metric of the complete classification system is chosen as the best one. Assessing a generic encoding technique is thus a time consuming activity, which introduces some additional degrees of freedom (the parameters of the training algorithm and of the testing protocol) that are uncorrelated with the encoding technique to be assessed. The computational cost of optimizing the encoder is typically so high, and the statistical significance of the

[1] E.g., the reader would probably agree on the fact that writing down a mathematical expression for connecting the level of brightness of an image to the classification error of a face recognition system is not a very effective way to spend anyone's time.

presented strategy so low, that the researchers tend to use "reasonable" values for the encoding parameters, selected by hand.

In this chapter, we propose a general method, the *Correlation Score*, capable of assessing the quality of any type of encoding technique in an efficient and statistically significant manner, and in isolation from the rest of the classification system. The main application for the Correlation Score is as a fitness function for genetic optimization algorithms.

The concept that drives the Correlation Score is simple and powerful, and can be stated as follows: when assessing an encoding technique on a classification dataset, we want it to maximize the correlation between the encoded feature vectors and the label associated with each of them, while minimizing the redundancy between the features. This way, when the Correlation Score of an encoder, say E_1, is greater than the score of another encoder, E_2, we expect that also the performance of a classification system will be better when using E_1 than when using E_2[2].

Behind this simple concept there are some important technical problems that needs to be solved: (a) we have to define a correlation coefficient that can handle both vectorial and categorical variables; (b) the estimation of such a correlation coefficient has to be stable in presence of high-dimensional feature vectors, large number of labels and noise; (c) the computational cost of the estimation should be kept to a minimum in order to make the Correlation Score usable as a fitness function.

In the remainder of the chapter, we give a brief overview of the correlation measures present in the literature, then we will describe the general algorithm to calculate the Correlation Score. We tested our method on a wide range of datasets and the experimental results show the effectiveness of our work and how it compares favorably with respect to information-theoretic approaches, that are the direct generalization of the fitness functions used in categorical feature selection.

We conclude the chapter discussing the results and talking about possible future research directions.

[2]Let us note that E_1 and E_2 do not necessarily need to be *different* encoders: they might be two parameter settings for the same underlying encoding technique.

2.1 Correlation and Association Measures

Let X and Y be two random variables, associated with a probability distribution $P_X(x)$ and $P_Y(y)$, that can either be continuous or discrete. In the case of vectorial random variables, we will use the bold symbols $\boldsymbol{X}$ and $\boldsymbol{Y}$.

Correlation refers to the tendency of two (or more) random variables to diverge from *probabilistic independence*: saying that two variables are correlated means that, to some extent, one of the two (the controlled or dependent variable) can be described as a function of the other one (the control or independent variable); if one of the two variables is a functional transformation of the other one, we say that they are totally correlated; on the other hand, if no relationship exists between the two, we say that they are totally uncorrelated. An ideal measure of correlation would then have the following two extreme values:

$$\text{corr}(X,Y) = 0 \iff P(X,Y) = P(X)P(Y) \tag{2.1}$$

$$\text{corr}(X,Y) = 1 \iff Y = f(X) \tag{2.2}$$

Values of $\text{corr}(X,Y)$ between 0 and 1 should represent an increasing relational bound between X and Y.

In practice, any given *correlation coefficient* will provide only an approximation of the previous two properties, and will work optimally only on selected use cases. Moreover, exact computation of a correlation coefficient requires the knowledge of the true distribution of X and Y, while we are typically given with a dataset of paired observations, $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, from which we can only calculate an estimate.

Another complication comes from the nature of the variables we are interested in. We want to calculate the correlation between the encoded feature vectors and the labels present in a dataset $\mathcal{D}$ (in the remainder of the chapter, we will refer to such type of correlation as vector-label correlation): $\boldsymbol{X}$ is then a vectorial variable, that is, the feature vector obtained by applying the encoding technique to the data, while Y represents the label associated with each feature vector, and is thus a categorical random variable. As we will shortly show, the literature concentrates on correlation coefficients that are either thought to be used between two *scalar* variables, between a vectorial variable and a scalar variable, or between one scalar variable and a categorical variable.

For this reason, for each correlation coefficient presented in the next sections, we will also show the adaptation needed to make it usable with a sample of paired vector-label observations.

2.2 Multivariate Linear Correlation

We start with the most known coefficient: its use is so widespread that, typically, when someone refers about a correlation coefficient without specification, he is probably referring to *Pearson's Product-Moment Correlation Coefficient*, also called *Linear Correlation Coefficient*.

Pearson's ρ is defined between two scalar variables and is the ratio between their covariance and the product of their standard deviations:

$$\rho(X, Y) = \frac{\text{cov}(X, Y)}{\sigma(X)\sigma(Y)} \tag{2.3}$$

To estimate ρ for a sample of N paired observations, we can substitute the covariance and of the standard deviations in the previous equation with their estimates, obtaining (when using MLE):

$$r(\underline{x}, \underline{y}) = \frac{\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y})}{s(\underline{x})s(\underline{y})} \tag{2.4}$$

Pearson's ρ ranges between -1 and $+1$, and can be interpreted as follows: when $\rho \approx +1$ or -1, then there is a strong linear relationship between X and Y; the sign of ρ is equal to the sign of the slope of the regression line between X and Y; when $\rho \approx 0$, no linear dependence exists.

The lack of linear correlation does not indicate that the two variables are independent, as is shown in. The figure represents various cases in which there is a clear relationship between X and Y, but because it cannot be expressed as a linear function, the associated Pearson's ρ assumes values near to zero. Even if its interpretation is often misleading, Pearson's ρ is a great tool to estimate correlation in a wide range of scenarios.

Pearson's ρ cannot be directly used to calculate the vector-label correlation, as it is defined between two scalar variables. We now show the steps needed to obtain a proper vector-label correlation coefficient based on ρ.

2.2.1 Converting Categories to Vectors

The first step is to convert the label variable, Y, to a vectorial representation, so that we can handle both variables, X and Y, as vectors. We call this step *label encoding*, but has nothing to do with the encoding technique that we want to assess, that is the one that was used to calculate $\boldsymbol{X}$ in the first place.

The simplest conversion occurs when there are exactly two possible labels, say $\mathcal{Y} = \{A, B\}$. In this case, we just need to map one of the two labels to the number 0 and the other one to 1, so that $\mathcal{Y} = \{0, 1\}$. We then say that in the case of *binary classification datasets*, we can convert the categorical label variable to a scalar that can only assume two values (zero and one).

In the general case of a set of m labels, we can resort to a *one-hot encoding*: each label is converted to an m-dimensional vector with all values set to zero except for one, whose position depends on the label. For example, if we have $\mathcal{Y} = \{A, B, C\}$, a possible one-hot encoding would be:

$$A = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

What we are doing is turning one m-class classification dataset into m binary datasets, one for each component of the output vector.

Once the label encoding has taken place, without lack of generality, we end up with two vectorial variables: $\boldsymbol{X}$, representing the encoded feature vectors with n components, and $\boldsymbol{Y}$, which represents the labels with one component if $m = 2$, or with m components if $m > 2$.

2.2.2 Correlation Matrices

A *correlation matrix* holds pairwise correlation coefficients between the components of two vectorial random variables. We are interested in building two correlation matrices. The first one contains the coefficients calculated between pairs of components of the encoded input vector, (X_j, X_h), and we call it *input-input correlation*

matrix:

$$\boldsymbol{C}^X = \begin{bmatrix} \rho(X_1, X_1) & \rho(X_1, X_2) & \dots & \rho(X_1, X_n) \\ \rho(X_2, X_1) & \rho(X_2, X_2) & \dots & \rho(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ \rho(X_n, X_1) & \rho(X_n, X_2) & \dots & \rho(X_n, X_n) \end{bmatrix} \tag{2.5}$$

Clearly, $\boldsymbol{C}^X$ is symmetric. The ideal input-input correlation matrix would be an identity matrix, as it would indicate that there is no (linear) redundancy between the features.

The second matrix we have to build is the one that contains the pairwise correlations between each feature and each component of the label encoding, (X_j, Y_k). This *input-output correlation matrix* is defined as:

$$\boldsymbol{C}^{XY} = \begin{bmatrix} \rho(X_1, Y_1) & \rho(X_1, Y_2) & \dots & \rho(X_1, Y_m) \\ \rho(X_2, Y_1) & \rho(X_2, Y_2) & \dots & \rho(X_2, Y_m) \\ \vdots & \vdots & \ddots & \vdots \\ \rho(X_n, Y_1) & \rho(X_n, Y_2) & \dots & \rho(X_n, Y_m) \end{bmatrix} \tag{2.6}$$

Let us notice that the k-th column of $\boldsymbol{C}^{XY}$ represents the correlation between each feature and the k-th label (remember that $\boldsymbol{Y}$ has been obtained by one-hot encoding of the labels). We will indicate the k-th column of $\boldsymbol{C}^{XY}$ as $\boldsymbol{c}_k^{XY}$:

$$\boldsymbol{c}_k^{XY} = \begin{bmatrix} \rho(X_1, Y_k), \\ \rho(X_2, Y_k), \\ \vdots, \\ \rho(X_n, Y_k) \end{bmatrix} \tag{2.7}$$

2.2.3 Coefficient of Determination

Once we defined $\boldsymbol{C}^X$ and $\boldsymbol{C}^{XY}$, we need a way to sum the information contained in these two matrices up into one variable, which synthetically represents the overall correlation between the feature vector and the labels.

We find a similar correlation coefficient in multivariate regression, where you need a metric that indicates how well the regression line fits the actual data. The metric

we are referring to is the *coefficient of (multiple) determination*:

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{N} (y_i - \overline{y})^2} \tag{2.8}$$

eq. (2.8) indicates that R^2 increases when the sum of squared errors decreases, that is, when the values found by the model, $\hat{y}_i$, fit more closely with the available data. The denominator term is needed to make the sum of squared errors proportional to the total variance of the data, so that the maximum value of R^2 is 1.

We use R^2 to calculate the overall correlation between the feature vectors and the k-th label. It can be shown that such a R^2 can be calculated by using the information contained in the input-input correlation matrix and the k-th input-output correlation vector:

$$R_k^2 = \left(\boldsymbol{c}_k^{XY}\right)^T \left(\boldsymbol{C}^X\right)^{-1} \left(\boldsymbol{c}_k^{XY}\right) \tag{2.9}$$

Equation (2.9) shows that the coefficient of determination is directly proportional to the correlation between the encoded features and the label, but each feature is weighted considering the redundancy it has with all the other features. This way, encoders with high redundancy among features get penalized with respect to encoders that keep them independent from one another.

2.3 Coefficients of Multivariate Association

The coefficient of determination described in section 2.2.3 is very powerful, yet it is based on the assumption of a linear relationship between the input vector $\boldsymbol{X}$ and the encoding of each label, Y_k. However, another type of relationship has to be assessed when considering the dependency between a scalar or a vector and a category (that is, the label associated with it), and it is investigated by Analysis of Variance (ANOVA) and Multivariate Analysis of Variance (MANOVA) [8].

Analysis of Variance and its multivariate counterpart give the tools to answer the following question: if we are given with samples drawn from different populations (that is, each group comes with a different label), is there enough evidence to say that there is a significant difference in the means of the populations? Said

differently, should we reject or accept the following null hypothesis:

$$H_0 : \boldsymbol{\mu}_{y_1} = \cdots = \boldsymbol{\mu}_{y_m} = \boldsymbol{\mu} \tag{2.10}$$

In the process of assessing the stated null hypothesis, (M)ANOVA provides a very useful measure: the *coefficient of association*, η^2. This coefficient is modeled to be directly proportional to the separation between the means of the considered populations, and ranges from 0 to 1. Values of η^2 near to 1 indicate that there is a clear distinction between the mean values of each population, and this in turn suggests the presence of a functional relationship (correlation) between the class label and the features.

In this section we present the coefficient of association due to Fisher, also called *correlation ratio* [9]. Unfortunately, once again this coefficient is not suited to be used directly to measure vector-label correlation, as it is only defined between a scalar and a categorical variable. We then describe a generalization of the same concept in the multivariate case, *Wilks' Measure of Association*.

As we shall discuss in section 2.5, Wilks' Measure of Association would be the preferred coefficient on which to base our Correlation Score, but its calculation is too unstable. Finding an improved algorithm for its computation is one of our main objectives as future works.

2.3.1 Fisher's Correlation Ratio

When measuring the dependency between a scalar feature and a categorical variable, often we want to investigate if the knowing the category narrows the variance of the feature, or modifies its expected values. This kind of dependence can be quantified by the *correlation ratio* introduced by Fisher. Various equivalent definition of the correlation ratio exists. We opted for the following one for the sake of clarity:

$$\eta^2(X|Y) = \frac{\sigma^2\left[E\left(X|Y\right)\right]}{\sigma^2(X)} \tag{2.11}$$

When Y can only assume discrete values, as in the case of a categorical variable, the correlation ratio can be interpreted as the ratio between the intraclass dispersion of X and its overall dispersion.

In order to explain the correlation ratio, it can be shown that [10]:

$$\eta^2(X|Y) = \max_{f(X)} \rho^2(f(X), Y) \tag{2.12}$$

that is, η^2 equals the maximal linear correlation between Y and any function of X. Hence, it can be used to highlight non-linear relationships between variables.

An estimate of η^2 on a sample of N paired observations, when Y is a categorical variable, is:

$$h^2 = \frac{\sum_{y \in \mathcal{Y}} N_y \left(\overline{x}_y - \overline{x}\right)^2}{\sum_{i=1}^{N} \left(x_i - \overline{x}\right)^2} \tag{2.13}$$

where N_y is the number of observations that fall in the category y, and $\overline{x}_y$ is the mean of the sample $\underline{x}_y$.

The correlation ratio is a non-symmetric coefficient ($\eta^2(X|Y) \neq \eta^2(Y|X)$), and ranges from 0 to 1. $\eta^2 \approx 0$ indicates no correlation, while $\eta^2 \approx 1$ shows that a strong dependence may exists. In particular, when Y is categorical, so that it indicates the class from which each observation has been taken, values of η^2 near to 1 indicate that the mean of each class is significantly different from that of the other ones.

2.3.2 Generalized Correlation Ratio

Fisher's correlation ratio is a very useful coefficient; however, it is properly defined only when X is scalar. Here we present a generalization to the multivariate case, due to Wilks. We just provide a quick overview of the generalization; for further information see [8].

Let us formalize some concepts first. Each category $y \in \mathcal{Y}$ is associated with a vectorial random variable $\boldsymbol{X}_y$, defined in the feature space $\mathcal{X}$, with population mean $\boldsymbol{\mu}_y$. $\boldsymbol{X}_y$ represents the population of feature vectors that belong to the category y. For each random variable $\boldsymbol{X}_y$ we have a sample $\underline{\boldsymbol{x}}_y = (\boldsymbol{x}_{y,1}, \ldots, \boldsymbol{x}_{y,N_y})$, consisting of N_y observations. Let us consider the null hypothesis that there is no difference in the population means, see eq. (2.10). In order to verify the hypothesis in the scalar case, we can easily compute the between-sample sum of squares and

the within-sample sum of squares:

$$\mathrm{SS}_b = \sum_{y \in \mathcal{Y}} N_y \left(\overline{x}_y - \overline{x}\right)^2 \tag{2.14}$$

$$\mathrm{SS}_w = \sum_{y \in \mathcal{Y}} \sum_{i=1}^{N_y} \left(x_{y,i} - \overline{x}_y\right)^2 \tag{2.15}$$

These two measures can be used to form independent estimators for $\sigma^2(X)$, so that, under the null hypothesis 2.10, the following ratio follows an F-distribution:

$$F = \frac{\mathrm{SS}_b / m-1}{\mathrm{SS}_w / m(N-1)} \tag{2.16}$$

Let us now define an equivalent of eq. (2.14) and eq. (2.15) for the multivariate case, in order to arrive to an equation similar to 2.16. By writing SS_b and SS_w in vectorial form, we obtain the "between" and "within" matrices:

$$\boldsymbol{B} = \sum_{y \in \mathcal{Y}} N_y \left(\overline{\boldsymbol{x}}_y - \overline{\boldsymbol{x}}\right) \left(\overline{\boldsymbol{x}}_y - \overline{\boldsymbol{x}}\right)^T \tag{2.17}$$

$$\boldsymbol{W} = \sum_{y \in \mathcal{Y}} \sum_{i=1}^{N_y} \left(\boldsymbol{x}_{y,i} - \overline{\boldsymbol{x}}_y\right) \left(\boldsymbol{x}_{y,i} - \overline{\boldsymbol{x}}_y\right)^T \tag{2.18}$$

So that we can define a likelihood ratio test for the multivariate H_0, equivalent to eq. (2.16), called *Wilks'* Λ:

$$\Lambda = \frac{|\boldsymbol{W}|}{|\boldsymbol{W} + \boldsymbol{B}|} \tag{2.19}$$

and we reject H_0 for small values of Λ: $\Lambda < \Lambda(\alpha, p, \nu_b, \nu_w)$, where $\Lambda(\alpha, p, \nu_b, \nu_w)$ is the *Wilks'* Λ *distribution*. It can be shown that Λ can be equivalently expressed in terms of the eigenvalues of the matrix $\boldsymbol{W}^{-1}\boldsymbol{B}$:

$$\Lambda = \prod_{i=1}^{s} \frac{1}{1 + \lambda_i} \tag{2.20}$$

where $(\lambda_1, \ldots, \lambda_s)$ are the non-null eigenvalues of the matrix $\boldsymbol{W}^{-1}\boldsymbol{B}$, and $s = \min(n, m-1)$. This equation is particularly useful for binary problems, in which $s = m - 1 = 1$, so that only one eigenvalue, say λ_1, is non null. This allows us to look only for the greatest eigenvalue instead of all of them (so that we can use algorithms for largest eigenvalue search, such as power iteration, which are far more stable than full-search algorithms). It follows that for binary problems we

can write:

$$\Lambda = \frac{1}{1 + \lambda_1} \tag{2.21}$$

Once we have Λ, we can define *Wilks' generalized* η^2:

$$\eta^2_\Lambda = 1 - \Lambda \tag{2.22}$$

Wilks' η^2_Λ is a very powerful coefficient, that provides us with a strong foundation on which to base our Correlation Score. The use of η^2_Λ leaves us with a few minor technical annoyances, as we will shortly describe, but shows to be more effective than either the Correlation Score based on the Coefficient of Determination and than the Score based on Mutual Information.

2.4 Uncertainty Coefficient

One of the reasons for the notoriousness of information theory is its widespread use in other fields, so it should not be a big surprise for the reader to find it cited here.

Information-based correlation coefficients estimate the dependency between two variables by considering how the uncertainty about one variable varies when the other one is known.

Let us recall that the uncertainty of a random variable, from an information-theoretic point of view, is calculated through its *entropy*:

$$H(X) = E\left[I(x)\right] = \sum_{x \in \mathcal{X}} P(x) I(x) \tag{2.23}$$

Equation (2.23) says that the entropy of a variable is the expected value of the *information* it provides:

$$I(x) = \log\left(\frac{1}{P_{\mathcal{X}}(x)}\right) = -\log\left(P_{\mathcal{X}}(x)\right) \tag{2.24}$$

Equation (2.24) asserts that the information to a value $x \in \mathcal{X}$ is inversely proportional to the probability of obtaining that value.

Another classical measure from information theory is the *mutual information* between two random variables, that is defined as follows:

$$I(X,Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(x,y)\frac{P(x,y)}{P(x)P(y)} \tag{2.25}$$

It can be shown that $I(X,Y)$ can be rewritten as:

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{2.26}$$

Where $H(X|Y)$ is called *conditional entropy* and is defined as:

$$H(X|Y) = \sum_{y\in\mathcal{Y}} P(y)H(X|Y=y) \tag{2.27}$$

Considering the two previous equations, we can interpret the mutual information between X and Y as the amount of entropy *removed* from a variable when the other is known. If our uncertainty about X remains the same after knowing Y (or vice versa), we can say that there is no correlation between X and Y. On the other hand, if all our uncertainty about the value of X is removed after we know Y, there is clearly a functional relationship between X and Y, so that there is total correlation between the two. Mutual information can then be used as a base function for a correlation coefficient. However, it needs to be scaled to $[0,1]$. Various normalization methods exists. We will use the *uncertainty coefficient*:

$$U(X,Y) = \frac{I(X,Y)}{H(Y)} = \frac{H(X) - H(X|Y)}{H(Y)} \tag{2.28}$$

The reasons for choosing this coefficient among all the possible ones will become apparent in the following.

As happened for the other correlation coefficients we discussed in the previous sections, we face the problem that the measures involved in calculating the uncertainty coefficient need some adaptations before we can use them with a vectorial variable, $\boldsymbol{X}$, and a categorical one, Y.

In particular, entropy is properly defined only in the case of *categorical* or discrete random variables, so that we need a definition for vectorial variables.

Moreover, we need en expression for mutual information that applies when one variable is vectorial and the other one is categorical. Many estimators for the

mutual information exists for pairs vectorial variables, but not for the case of vector-label pairs.

However, by using eq. (2.26) we can write the mutual information as:

$$I(\boldsymbol{X}, Y) = H(\boldsymbol{X}) - \sum_{y \in \mathcal{Y}} P(y) H(\boldsymbol{X}|Y = y) \tag{2.29}$$

Where we only need an estimate for $P(y)$ and for $H(\boldsymbol{X})$, as estimating $H(\boldsymbol{X}|Y = y)$ is just a special case of the estimation of $H(\boldsymbol{X})$ in which we consider only the samples labeled as y. $P(y)$ is straightforwardly estimated using the frequentist definition of probability:

$$P(y) \approx \frac{N_y}{N} \tag{2.30}$$

where N_y is the number of observations labeled as y.

On the other hand, estimating $H(\boldsymbol{X})$ is a non trivial task. We tested many different estimators but only one of them proved effective:

$$H(\boldsymbol{X}) = \sum_{i=1}^{N} \log\left(N \rho(\boldsymbol{x}_i, \underline{\boldsymbol{x}})\right) + \log 2 + \gamma \tag{2.31}$$

where $\rho(\boldsymbol{x}_i, \underline{\boldsymbol{x}})$ is the distance of $\boldsymbol{x}_i$ from its nearest neighbor in the sample $\underline{\boldsymbol{x}}$ (obviously excluding $\boldsymbol{x}_i$ itself) and γ is the Euler's constant (approximately equal to 0.577). Let us point out that when calculating the estimate of $H(\boldsymbol{X}|Y = y)$, we have to replace $\underline{\boldsymbol{x}}$ with $\underline{\boldsymbol{x}}_y$ (the observations labeled as y) and consequently N with N_y.

Using eq. (2.31), eq. (2.30) and eq. (2.23) (the latter to calculate the entropy of Y), we can finally compute an effective estimate of the uncertainty coefficient 2.28 for the vector-label case.

2.5 Definition of Correlation Scores

As we have seen, the literature on correlation coefficients is very rich. The main part of our work was to go further than the classical machine-learning related statistical methods, which concentrate on information-theoretic stuff or on simple scalar-to-scalar correlation coefficients. This search has proved to be valuable, as we found a classical correlation measure, Wilks' Measure of Association, that is

substantially ignored in the rest of machine learning literature. Even the use of the Coefficient of Determination, R^2, as a measure of the correlation between the features and the labels in a dataset is something rare to find in contemporary works on the subject.

However, these measures prove all their power in assessing the encoding techniques, as we show in the section dedicated to experimental results.

To "convert" each of the correlation coefficients just presented to a *Correlation Score*, very few technical issues needs to be taken care of.

The Correlation Score algorithm can be formally stated as follows:

$$\mathrm{cs}(\mathrm{enc}(\underline{\boldsymbol{o}}), \underline{y}) = \frac{1}{m} \sum_{k=1}^{m} \mathrm{corr}\left(\mathrm{p}(\underline{\boldsymbol{x}}), \mathrm{le}_k(\underline{y})\right) \tag{2.32}$$

where $\mathrm{enc}(\underline{\boldsymbol{o}}) = \underline{\boldsymbol{x}}$ indicates explicitly that we consider the sample of *encoded* feature vectors $\underline{\boldsymbol{x}} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$. For the sake of clarity, we wrote $\mathrm{corr}(\underline{\boldsymbol{x}}, \underline{y})$ to indicate the *estimate* of the true correlation coefficient $\mathrm{corr}(\boldsymbol{X}, Y)$.

The Correlation Score can be then described as a "synthetic value" that measures the effectiveness of an encoding technique by using an underlying measure of correlation between the encoded instances and the associated labels. In order to obtain an useful value, we have to make sure that the data is fed to the underlying correlation coefficient in the correct shape, as indicated in eq. (2.32) with $\mathrm{p}(\underline{\boldsymbol{x}})$ and $\mathrm{le}_k(\underline{y})$, functions that we now discuss.

The correlation coefficients described in 2.2, 2.3 and 2.4 are statistical measures that assume the presence of a certain degree of noise in the data and that enough data exists to make the computation. This consideration proved true in preliminary tests, in which we found how injecting a small random gaussian noise to the input sample $\underline{\boldsymbol{x}}$ improved the overall performance of the Correlation Scores. Moreover, feeding the correlation coefficient with a bootstrapped sample of the original observations may increase the effectiveness of the estimation, in particular when there is a strong imbalance in the number of observations among classes. For these reasons, we apply a "preparation" step to the encoded sample $\underline{\boldsymbol{x}}$, indicated as $\mathrm{p}(\underline{\boldsymbol{x}})$ in eq. (2.32).

As for the function $\mathrm{le}_k(\underline{y})$, it indicates that an encoding of the label variable is made, as discussed in section 2.2.1. This might be needed by the underlying correlation measure, as in the case of the Coefficient of Determination, or to increase the

stability of the estimate, as in the case of Wilks' Measure of Association, that might become unstable in presence of a large number of classes. The index k in $\text{le}_k(\underline{y})$ indicates that the k-th component of the label encoding is considered.

In our work, we experimented with three Correlation Scores. The first one is the Correlation Score based on the Coefficient of Determination:

$$\text{cs}_\rho(\underline{\boldsymbol{x}}, \underline{y}) = \frac{1}{m} \sum_{k=1}^{m} R^2 \left(\underline{\boldsymbol{x}}, \text{le}_k(\underline{y})\right) \tag{2.33}$$

in which we always use a one-hot label encoding, but the preparation of the feature vectors is in general not needed.

The second Correlation Score is based on Wilks' Measure of Association:

$$\text{cs}_\eta(\underline{\boldsymbol{x}}, \underline{y}) = \eta^2_\Lambda \left(\text{p}(\underline{\boldsymbol{x}}), \underline{y}\right) \tag{2.34}$$

in which we do not need to encode the labels, but the calculation of the eigenvalues to compute η^2_Λ become stabler if we inject Gaussian noise on $\underline{\boldsymbol{x}}$, so that $\text{p}(\boldsymbol{x}_i)$ is:

$$\text{p}(\boldsymbol{x}_i) = \boldsymbol{x}_i + \epsilon_n \mathcal{N}(\mu = 0, \sigma^2 = 1) \qquad i = 1, \ldots, N \tag{2.35}$$

and $\text{p}(\underline{\boldsymbol{x}}) = (\text{p}(\boldsymbol{x}_1), \ldots, \text{p}(\boldsymbol{x}_N))$. The *noise level* ϵ_n is typically set in the order of $10^{-4} \div 10^{-6}$.

Finally, we define a Correlation Score based on the Uncertainty Coefficient:

$$\text{cs}_U(\underline{\boldsymbol{x}}, \underline{y}) = U \left(\text{p}(\underline{\boldsymbol{x}}), \underline{y}\right) \tag{2.36}$$

that once again does not need encoded labels, just Gaussian noise injection.

cs_U has been defined for comparison purposes only. In fact, information-based measures are the standard in feature selection algorithms, and we wanted to compare our approach with the one most used in the literature. As we show in the next section, the first two Correlation Scores show overall better performances when we have to assess multiple continuous features at once, so that they prove to have a more widespread application for encoding assessment.

2.6 Experiments on Feature Subset Assessment

We conducted experiments on the 35 UCI datasets presented in the introduction. This forced us to limit our tests on only one type of encoding technique, that is, feature selection, as the dataset features were already computed and provided to us in their definitive form. On the other hand, by using our metrics for assessing feature subsets, we are able to compare the results obtained by the Correlation Scores based on the Coefficient of Determination and on Wilks' Measure of Association with the one based on the Uncertainty Coefficient, and in so doing, we can compare our approach on feature selection with the one most present in the literature.

2.6.1 Experimental Protocol

We wanted to verify which of the Correlation Score algorithms described in the previous section is able to act as a fitness function. In order to make this assessment, we compared the values found by each Correlation Score with the experimental error rate of a set of classifiers trained on the same data.

We will now formally describe the experimental protocol. Let us represent each dataset as $\mathcal{D} = \{(\boldsymbol{o}_1, y_1), \ldots, (\boldsymbol{o}_N, y_N)\}$. Here, $\boldsymbol{o}_i$ represents the i-th *uncoded* instance of the dataset, that comes associated to its own label, y_i. Let $\text{enc}(\boldsymbol{o}|\boldsymbol{\theta})$ be an encoding technique for the instances of the dataset, with parameters $\boldsymbol{\theta}$. If we encode the dataset using Q different configurations for the encoding parameters, we obtain Q different encoded datasets: $\mathcal{D}_q = \{(\boldsymbol{x}_{q,1}, y_1), \ldots, (\boldsymbol{x}_{q,N}, y_N)\}$, for $q = 1, \ldots, Q$. In the experiments, we set Q to 10.

For each encoded dataset, we then compute the Correlation Score, as defined in the previous section, and the classification error rate obtained by a set of predefined classifiers (see Section 2.6.2). To ensure statistical significance, we ran a 10-fold cross validation to compute both the Correlation Score and the error rate.

Once the computation is finished, we have a sample of Correlation Scores, $\underline{\text{cs}} = (\text{cs}_1, \ldots, \text{cs}_Q)$, and a sample of classification errors, $\underline{\text{err}} = (\text{err}_1, \ldots, \text{err}_Q)$. In both cases, the q-th observation in the sample is the value associated to the q-th encoded dataset.

We can now measure the linear correlation between $\underline{\text{cs}}$ and $\underline{\text{err}}$. Presence of a strong linear correlation between the two vectors indicates that the Correlation Score is

indeed capable of acting as a fitness function. In particular, we look for values of the correlation near -1, so that high values of the Correlation Score corresponds to low values of the classification error.

We will show the scatter plot between each Correlation Score and the classification error of each classifier, in order to show the typical output of the comparison. Moreover, we provide tables containing the numerical estimates of the linear correlation between the Correlation Scores and the error rates. Lastly, we highlight that the Correlation Score as a fitness function is far less expensive than calculating the classification error, from a time cost point of view.

2.6.2 On the Significance of the Comparison

It could be argued that comparing cs with err to assess the quality of the Correlation Scores as fitness functions might lack statistical significance, as the error rate of a classifier is influenced by the parameters of the classification algorithm being used. It would be better to compare cs with, for example, an estimate of the Bayes error. However, as our principal aims in this thesis is to improve the performances of classifier ensembles, it seems reasonable to compare cs with the error rate obtained by state-of-the-art ensemble algorithms, in order to find out how the Correlation Scores estimates the error rate of an ensemble when used together with a certain input encoding.

Moreover, the variance and bias of the predictions of a classifier ensemble is generally very low, so that the error rate obtained can be used as an estimate of the Bayes error, in particular when the size of the ensemble is large.

For the sake of completeness, we compared cs with the error rates obtained by three types of classifier ensembles: AdaBoost, Bagging and Random Forest. We assume that the reader is already familiar enough with these classification algorithms. For further information, see Chapter 4 and [11].

We used the default values used by *scikit-learn* for each classifier. The ensemble size has been set to 100.

2.6.3 Random Feature Subset Encoder

Once the experimental protocol has been set, we have to specify the encoder on which we want to base our tests. For the reasons described at the beginning of the section, we decided to use a Feature Subset Encoder. Moreover, to make sure that the encoder itself does not give bias in favor of our Correlation Scores, we opted for a *Random Feature Subset Encoder*. That is, the encoding technique selects a random subset of the features, of specified size t. By running the encoder Q times, we get Q encoded dataset, each of them with a different subset of t features. By evaluating the correlation score of each encoded dataset, we are able to rank the subsets according to their expected discriminative power (that is, their potential of increasing the classification performance).

For each dataset, we ran a comparison for a maximum of 15 different values for t: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35. Of course, if the total number of feature in a dataset was lower or equal to a chosen t, the corresponding experiment has not been run.

2.6.4 Experimental Results

Table 2.1 shows the overall performance of each Correlation Score on each dataset, when compared with Random Forest error rate. The value in parenthesis is the total number of experiments on the corresponding dataset. A value lower than 15 indicates that the dataset does not have enough features to run all the experiments, as discussed above. For each dataset, we ran experiments on the original feature set and on a version in which each feature has been normalized to $[0, 1]$. The number under each column indicates how many times the corresponding Correlation Score has reached a significant linear correlation with the error rate of Random Forest. Values in bold indicate that at least two thirds of the experiments have reached significance.

The Table shows that the correlation scores based on either the Coefficient of Determination or Wilks' η^2 perform better than those based on the Uncertainty coefficient. Preliminary experiments showed the same trend when using other information-based coefficients. Even on datasets with only categorical features, as the *audiology* dataset, the Uncertainty Correlation Score only reaches significance on very few experiments.

Dataset (runs)	R^2		Wilks' η^2		Uncertainty	
	orig	norm	orig	norm	orig	norm
anneal-orig (15)	4	4	3	3	1	1
anneal (15)	**12**	2	**10**	2	1	1
audiology (15)	3	5	3	5	0	0
autos (15)	9	**10**	9	**10**	0	0
balance-scale (3)	0	1	0	1	1	0
breast-cancer (15)	3	8	3	8	0	0
breast-w (8)	**7**	0	**7**	0	**5**	4
colic-orig (15)	**14**	**14**	**14**	**14**	0	0
colic (15)	4	**15**	4	**15**	2	4
credit-a (15)	**14**	6	**14**	6	0	0
credit-g (15)	6	**10**	6	**10**	0	0
diabetes (7)	4	2	4	2	3	3
glass (8)	1	4	2	4	0	0
heart-c (12)	**10**	**11**	**10**	**11**	0	0
heart-h (12)	**10**	**10**	**10**	**10**	0	1
heart-statlog (10)	**10**	**9**	**10**	**9**	0	3
hepatitis (13)	**9**	4	**9**	4	2	0
hypothyroid (14)	**14**	**10**	**14**	**10**	3	6
ionosphere (14)	2	1	2	1	2	2
iris (3)	**2**	**2**	**3**	1	0	1
kr-vs-kp (15)	**15**	**15**	**15**	**15**	5	8
labor (13)	**11**	**9**	**11**	**9**	0	0
letter (11)	**11**	**11**	**11**	**11**	**8**	**9**
lymph (15)	3	1	0	1	0	3
primary-tumor (13)	**12**	**11**	**12**	**12**	1	0
rootstock (3)	**3**	1	**3**	1	1	0
segment (11)	**11**	**11**	**10**	6	**9**	1
sick (14)	**14**	**10**	**14**	**10**	7	8
sonar (15)	7	8	7	8	3	0
soybean (15)	**13**	**12**	8	**10**	2	0
splice (15)	**14**	**14**	**14**	**14**	0	2
vehicle (11)	3	5	3	4	4	2
vote (14)	**13**	**13**	**13**	**13**	2	2
vowel (13)	**13**	**10**	**13**	**10**	5	4
waveform (15)	**15**	**15**	**15**	**15**	**14**	**14**
zoo (15)	**15**	**15**	**11**	**13**	0	0
Total (447)	311	289	297	278	81	79

TABLE 2.1: Overall significant results per dataset when compared to Random Forest error rate.

Dataset (runs)	R^2		Wilks' η^2		Uncertainty	
	orig	norm	orig	norm	orig	norm
anneal-orig (15)	3	3	3	3	1	1
anneal (15)	**11**	2	**10**	2	1	1
audiology (15)	3	5	3	4	0	0
autos (15)	9	8	9	8	0	0
balance-scale (3)	0	1	0	1	1	0
breast-cancer (15)	2	6	2	6	0	0
breast-w (8)	4	0	4	0	4	4
colic-orig (15)	**14**	**14**	**14**	**14**	0	0
colic (15)	4	**15**	4	**15**	3	4
credit-a (15)	**14**	7	**14**	7	0	0
credit-g (15)	6	**10**	6	**10**	0	0
diabetes (7)	**5**	2	**5**	2	3	3
glass (8)	2	3	2	4	0	0
heart-c (12)	**10**	**10**	**10**	**10**	0	0
heart-h (12)	**10**	**11**	**10**	**11**	0	0
heart-statlog (10)	**7**	**9**	**7**	**9**	0	2
hepatitis (13)	**9**	5	**9**	5	1	0
hypothyroid (14)	**14**	**10**	**14**	**10**	3	6
ionosphere (14)	1	1	1	1	4	2
iris (3)	**2**	1	**3**	**2**	0	0
kr-vs-kp (15)	**15**	**15**	**15**	**15**	5	8
labor (13)	**13**	**13**	**13**	**13**	0	0
letter (11)	**11**	**10**	**11**	**11**	**8**	**9**
lymph (15)	4	1	0	0	0	3
primary-tumor (13)	**11**	**12**	**11**	**12**	0	0
rootstock (3)	**2**	1	**2**	1	1	0
segment (11)	**11**	**11**	**10**	6	**9**	1
sick (14)	**14**	**10**	**14**	**10**	7	8
sonar (15)	6	6	6	6	3	0
soybean (15)	**13**	**11**	8	**10**	2	0
splice (15)	**14**	**14**	**14**	**14**	0	1
vehicle (11)	3	6	3	4	2	2
vote (14)	**12**	**12**	**12**	**12**	1	2
vowel (13)	**12**	**10**	**12**	**11**	5	5
waveform (15)	**15**	**15**	**15**	**15**	**14**	**14**
zoo (15)	**15**	**14**	**11**	**13**	0	0
Total (447)	301	284	287	277	78	76

TABLE 2.2: Overall significant results per dataset when compared to Bagging error rate.

Dataset (runs)	R^2		Wilks' η^2		Uncertainty	
	orig	norm	orig	norm	orig	norm
anneal-orig (15)	1	0	0	1	0	0
anneal (15)	1	0	1	0	0	0
audiology (15)	1	5	5	3	0	0
autos (15)	**11**	4	**11**	2	0	0
balance-scale (3)	0	1	0	1	1	0
breast-cancer (15)	**10**	9	**10**	9	0	0
breast-w (8)	**8**	0	**8**	0	4	4
colic-orig (15)	**14**	**14**	**14**	**14**	0	0
colic (15)	3	**15**	3	**15**	2	3
credit-a (15)	**15**	7	**15**	7	0	0
credit-g (15)	9	**10**	9	**10**	0	0
diabetes (7)	**6**	4	**6**	4	1	4
glass (8)	1	1	1	2	0	0
heart-c (12)	**9**	**11**	**9**	**11**	0	0
heart-h (12)	**9**	**9**	**9**	**9**	0	0
heart-statlog (10)	**8**	**10**	**8**	**10**	0	0
hepatitis (13)	**9**	1	**9**	1	0	0
hypothyroid (14)	5	4	5	4	3	7
ionosphere (14)	2	**10**	2	**10**	3	1
iris (3)	1	1	**2**	**2**	1	1
kr-vs-kp (15)	**15**	**15**	**15**	**15**	5	8
labor (13)	**12**	**10**	**12**	**10**	0	0
letter (11)	**7**	**8**	**8**	**8**	6	6
lymph (15)	7	2	1	2	0	2
primary-tumor (13)	3	3	3	4	0	0
rootstock (3)	**3**	**3**	**3**	**3**	1	0
segment (11)	2	3	3	3	0	0
sick (14)	**14**	**14**	**14**	**14**	7	8
sonar (15)	**11**	7	**11**	7	1	0
soybean (15)	2	1	1	0	0	0
splice (15)	**14**	**14**	**14**	**14**	0	1
vehicle (11)	3	3	3	3	0	0
vote (14)	**12**	**12**	**12**	**12**	1	2
vowel (13)	**8**	6	**8**	6	3	0
waveform (15)	**15**	**15**	**15**	**15**	**14**	**14**
zoo (15)	**12**	**13**	**10**	**12**	0	0
Total (447)	263	245	260	243	53	61

TABLE 2.3: Overall significant results per dataset when compared to AdaBoost error rate.

Subset size (runs)	R^2		Wilks' η^2		Uncertainty	
	orig	norm	orig	norm	orig	norm
1 feature (36)	**23**	17	**23**	16	0	0
2 features (36)	**26**	**23**	**26**	**22**	2	1
3 features (36)	**27**	**22**	**25**	**21**	5	5
4 features (33)	**28**	**23**	**26**	**23**	7	5
5 features (33)	**24**	**20**	**25**	**20**	5	5
6 features (33)	**23**	**24**	**24**	**24**	6	4
7 features (33)	**23**	**23**	**22**	**23**	6	7
8 features (32)	**23**	**22**	**23**	**22**	6	7
9 features (30)	**21**	**22**	**22**	**21**	5	7
10 features (30)	**19**	**20**	**19**	**19**	7	9
15 features (29)	**18**	**18**	**17**	**18**	8	9
20 features (26)	**19**	**16**	**16**	**15**	7	5
25 features (24)	**13**	**16**	11	**14**	6	5
30 features (20)	**12**	**13**	9	**11**	5	5
35 features (16)	**12**	**10**	**9**	**9**	6	5
Total (447)	311	289	297	278	81	79

TABLE 2.4: Overall significant results per feature subset size when compared to Random Forest error rate.

The comparison with Bagging and AdaBoost error rates is shown in Tables 2.2 and 2.3. The same trend is present, but in particular for AdaBoost with notice less overall correlation. This effect has still to be investigated.

Table 2.4 presents the same results of Table 2.1 from a different perspective. It shows the overall performance grouped by sample size. Again, the value in parenthesis indicates the total number of experiments with the corresponding subset size. The table shows how the performance of the first two Correlation Scores remains constant even when larger sets of features are considered at once. The slight decrease in performance is due to the fact that more data is needed to evaluate the coefficients, so that the estimate on smaller datasets become instable. Table 2.5 shows the results of the comparison against AdaBoost error rate.

Another interesting insight is shown in Figures 2.2 and 2.3. They show how the performance varies when the spread in the Correlation Score increases. Clearly, a large difference in Correlation Score between two encoding configurations should

Subset size (runs)	R^2		Wilks' η^2		Uncertainty	
	orig	norm	orig	norm	orig	norm
1 feature (36)	**24**	**19**	**24**	**19**	0	0
2 features (36)	**19**	17	**20**	**19**	3	1
3 features (36)	**23**	**22**	**20**	**21**	3	5
4 features (33)	**19**	**20**	**21**	**20**	3	3
5 features (33)	**23**	**18**	**23**	**18**	4	4
6 features (33)	**22**	**20**	**21**	**20**	3	3
7 features (33)	**20**	**17**	**22**	**17**	3	5
8 features (32)	**21**	**17**	**21**	**17**	6	6
9 features (30)	**16**	**17**	**16**	**16**	4	6
10 features (30)	**18**	**17**	**18**	**16**	5	6
15 features (29)	13	**15**	14	14	4	6
20 features (26)	**15**	**14**	**14**	13	5	5
25 features (24)	10	12	9	12	4	5
30 features (20)	**11**	**11**	10	**13**	4	4
35 features (16)	**9**	**9**	7	8	2	2
Total (447)	263	245	260	243	53	61

TABLE 2.5: Overall significant results per feature subset size when compared to AdaBoost error rate.

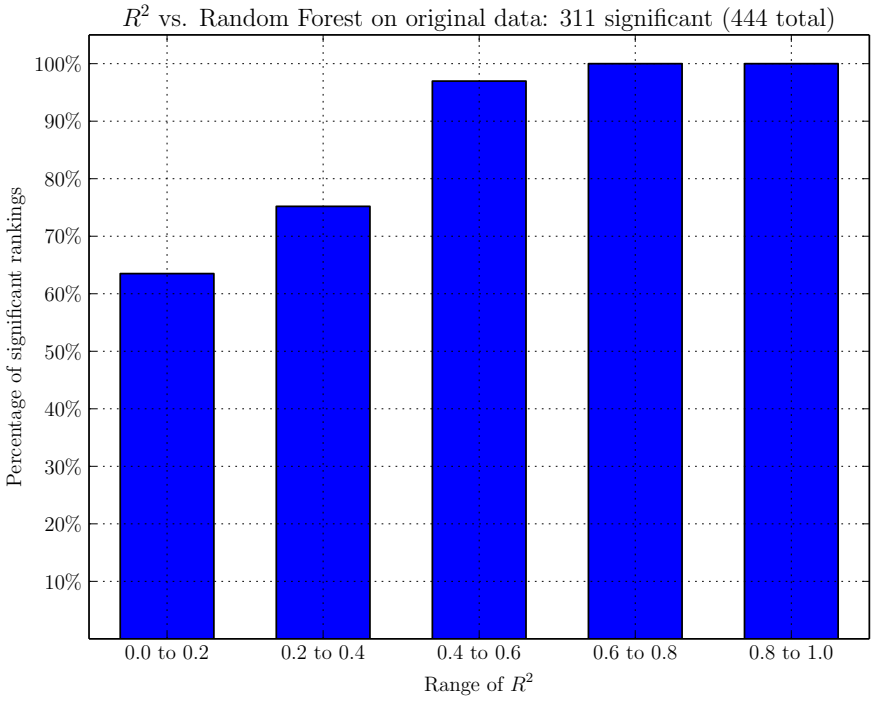


FIGURE 2.2: Percent of significant results grouped by range of R^2.

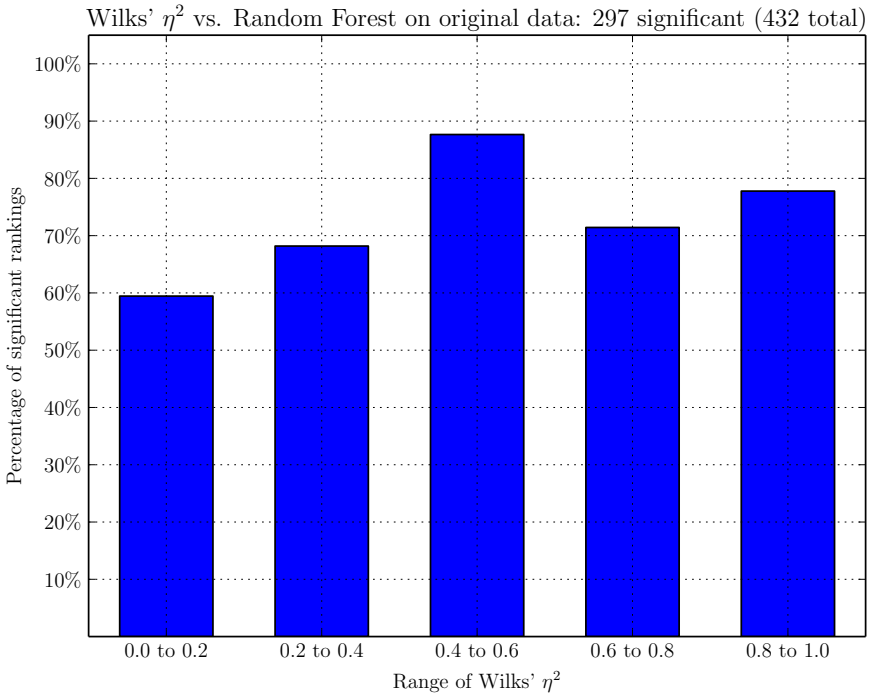


FIGURE 2.3: Percent of significant results grouped by range of Wilks' η^2.

indicate that one of the two is far better than the other. If two configurations have roughly the same correlation score, the "ranking" between the two becomes less reliable. In effect, this is what happens, as shown in particular in Fig. 2.2: when the range of R^2 in the cs becomes larger than 0.4, then the correlation between the error rate and the Correlation Score is almost always significant.

This indicates that the Correlation Scores, and in particular the one based on R^2, can be readily used to prune the most ineffective encoding configurations, so that the search space for the optimal encoder can be significantly narrowed.

2.7 Conclusions

We have built a framework for assessing encoding techniques and encoder configurations in isolation from the complete classification system. This framework is based on the well known concept of correlation coefficients, on top of which we provided a protocol to evaluate the effectiveness of an encoding in improving the classification performance of the overall system. This is done by calculating the correlation between the features (input) and the labels (output), and then adjusting the estimate considering the correlation between the features (intra-input

or input-input correlation). The experimental results, conducted on the field of feature selection, proved the effectiveness of our approach, in particular when compared with information-based methods.

The fact that we only conducted experiments on feature subset assessment should not mislead the reader. Many other types of encoding exist, and the application of our Correlation Score for their optimization is definitively one of our priorities in future works. Among these types of encoding techniques, we cite image enhancement, amino acid encodings and supervised feature discretization.

We provide a short unordered list of our planned future work on the topic:

- Consider the use of better estimates for R^2 and η^2_Λ, in particular the use of *robust* estimates.
- Construct an algorithm that uses our Correlation Scores as fitness function, capable of optimizing the most broad possible range of encoding techniques.
- Investigate on the use of more correlation coefficients. In particular, *distance correlation* seems promising, as it aims at measuring the effective dependence between two samples, with very few conditions on the nature of the relationship.

Chapter 3

Multi-Resolution Complexity Analysis

As discussed in the previous chapter, the performance of a classifier system is strongly affected by the intrinsic characteristics of the domain at hand, and a lot of studies exist on the evaluation of the *classification complexity* of a domain. In the next section, we present the State of the Art on this field.

However, our work has an aim that is somewhat different from the one shared by most of the literature. In our view there are at least three different objectives in the field of classification complexity.

The first one is to look for a measure able of *ranking* domains from the most difficult to the easiest. A typical measure is the *Bayes Error Rate* [12], but other measures have been proposed, as it is often troublesome to compute the Bayes error rate with enough confidence.

The second objective is to measure the *complexity type* of a domain: identifying the characteristics that make a particular problem more or less complex. This objective stems from the observation that two domains may rank at a similar complexity level, but their classification complexity may have different reasons (e.g.: sparseness of data, population overlap, noise, poor dataset quality). The typical application of such studies is the identification of the domain of competence of a classification algorithm.

The third objective – our objective – is estimation of the inherent classification complexity of each instance of a domain. In effect, the classification complexity of

an instance in a dataset varies with its position inside the feature space. Assigning a "complexity estimate" to each observation in a dataset may help the classifier algorithm in the labeling task, and may give the researcher a better understanding of the characteristics of the domain at hand – without the need of comparing it with other datasets.

In this chapter we present our Multi-Resolution Complexity Analysis (or $MRCA$ for short), a method for identifying regions of increasing classification complexity inside a dataset. To our knowledge, no other works in the literature provide a comparable method.

The idea that drives $MRCA$ is that the classification complexity of each instance in a domain varies with the characteristics of its neighborhood, so that it is not a function of the whole domain space, but is associated to the "local" characteristics of a domain. For example, if an instance of a given class is surrounded by instances labeled differently, assigning the correct label to that instance would be a very complex task; on the other hand, an instance surrounded by examples of the same category is easily labeled in the correct way (this is the idea behind methods like k-nearest neighbors and Parzen's windows). In presence of an infinite number of examples, one could take the previous observation to the limit and consider only an infinitely small hyper-sphere around any given instance in order to evaluate its classification complexity. As we are provided with finite datasets, another approach should be pursued. We would have to look for an hyper-sphere with a population large enough to convey the desired information, but not too large, as it would lose the "locality" requirement.

To overcome the problem of looking for the right hyper-sphere, we propose to take complexity estimates at different resolutions, and to use this *multi-resolution complexity profile* as an intermediate space with which we describe the complexity of each observation. This space would have the following property: instances with similar complexity would be close to each other, so that it is possible to cluster them. Once the clusters have been found, it would be possible to estimate the classification complexity of each cluster by using a metric of our definition, the *Multi-Resolution Index*. In this regard, we approximate the the intrinsic classification complexity of each sample (its "point-wise" complexity) with the complexity of the cluster in which it resides.

The rest of the chapter is organized as follows: Section 3.1 shortly describe the State of the Art on classification complexity; in Section 3.2 we discuss the overall approach, while the details are described in Section 3.3; in Section 3.4 we show the experimental results, proving the effectiveness of *MRCA*. Section 3.5 concludes the chapter.

3.1 Studies on Classification Complexity

A simple strategy for estimating the intrinsic complexity of a classification task consists of analyzing the results of several classification trials, typically against a validation set. This approach has the major limitation already discussed in the previous chapter: results are not intrinsic properties of the dataset under analysis, being also related to the characteristics of the selected learning algorithms – including the way they explore the space of solutions and their dependence on the parameters that affect their behavior. A trivial solution to this problem is to increase the number of trials, ranging over the parameters of the learning algorithms. However, to reach a level of statistical significance, the time required for running experiments may considerably increase.

On the other hand, estimating the complexity of a domain independently from a classification algorithm is a hard task. Metrics used to this purpose take into account some aspects deemed potentially relevant for assessing the characteristics of the decision surfaces, e.g. the interclass separability or the geometry of clusters. Hence, to perform a comprehensive analysis of the classification complexity, a large number of metrics should be used (see [13, 14] for a review on these issues). It is worth pointing out that the analysis of classification complexity is often framed in the study of misclassification errors. As the Bayes error represents the lower limit of the classification error, it can be used to estimate the classification complexity, which is typically calculated by measuring the error of a Bayes classifier built on (an estimate of) class-conditional distributions. In [15], these distributions are represented as parametric models (although they can also be generated using non-parametric models [16, 17] or density estimators [18]).

In accordance with the principle of *divide and conquer*, several works focus on the impact of a single feature on the classification task. Fisher Correlation Ratio

is a notable example of this approach [19]. The presence of at least one highly-efficient feature identifies an easy problem (however, the cited metric is not suited in presence of mutual interactions among features). The volume of overlapping regions [14] is another relevant metric, which shares similarities with the previous one. It quantifies the classification complexity by measuring the amount of overlapping on the tails of class distributions. In a work that goes back to 1998, Ho and Baird [20] propose the feature efficiency measure, with the goal of estimating to which extent a feature contributes to class separability. Given a feature, the whole training set is projected onto it –typically using a linear transformation. The percent of separable points is then evaluated and the resulting value is considered as an estimate of the discriminant power of the feature.

The classification complexity can also be estimated by considering the separability of a problem. In this case, the mutual-interaction among features is analyzed to estimate the behavior of a classifier. The most commonly adopted measure of complexity in this framework, called linear separability, is based on the error made by a linear classifier, which is expected to be an indicator of separability between classes. Several algorithms have been proposed for this purpose (see, for instance, [21]). In all cases, a classifier is represented as a hyperplane whose parameters are calculated by minimizing an error function. Regardless of overlapping, the lesser the linear separability is, the greater the effort a classifier must make to model the decision surfaces is. Here, the underlying assumption is that the difficulty in modeling decision surfaces affects the complexity of the corresponding classification task.

Separability can also be framed in a statistical perspective. The effectiveness of defining distinct class-conditional distributions can be related to the identifiability of classes and therefore to the complexity of the given dataset. This method, commonly referred as mixture identifiability, is based on a test used for determining whether different class samples fall in the same or in distinct distributions [22]. Clearly, the more class distributions are interlaced, the greater the effort required for classification is.

More complex methods have also been proposed. In particular, Hoekstra and Duin [23] propose a method for estimating the non-linearity of a classification model. To give an insight of the method, let us consider the points of a linear path drawn between two arbitrary samples labeled in the same way. This method assumes that the probability to belong to the same class of the selected samples, for the points

that occur in between, is related to the non-linearity of the problem. Looking at the method from a different perspective, we can say that it investigates the shape of class clusters by detecting the presence of concavities of class boundaries.

In 2003, Singh [13] proposes a multi-resolution analysis for complexity estimation. In [24], another method based on feature space partitioning is proposed. This method applies an adaptive partitioning strategy, similar to that used in decision trees, for estimating the non-linearity and the homogeneity of the resulting clusters. This method estimates the inherent classification complexity by measuring the so-called data purity and neighborhood separability, at different degrees of resolution. Results are then integrated to produce a single estimate. In 2011, Jie Li et al. [25] study the problem of feature space complexity estimate in a framework of computer vision, proposing a robust tensor subspace learning algorithm able to capture the appearance changes by adaptively updating the tensor subspace. In this framework, the spatial structure information is maintained and utilized for extracting object features. In 2012, Nan Li et al. [26] address the problem of complexity estimation proposing an updated version of the k-nearest neighbor algorithm. In particular, to facilitate the classification task, an optimal subspace classification method has been devised, able to project different training samples onto their own optimal subspace and to construct the corresponding class cluster as the basis of classification.

As said at the beginning of the chapter, one of the most common applications of complexity estimation is to discover the domain of competence of a class of classification algorithms. For such kind of studies see, for example, [27] and [28]. In [29], the authors describe the design of an algorithm that automatically select the optimal classifier given the complexity characteristics of a domain. In [30] the authors apply the measures of complexity to analyze the behavior of various techniques for imbalance reduction.

3.2 Multi-Resolution Complexity Analysis

Let $\mathcal{D}$ be a dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$. We want to split $\mathcal{D}$ into s disjoint sets (or clusters) $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(s)}$ for which the following inequality has to hold:

$$\text{err}(\mathcal{D}^{(1)}) > \text{err}(\mathcal{D}^{(2)}) > \cdots > \text{err}(\mathcal{D}^{(s)}) \tag{3.1}$$

where $\text{err}(\mathcal{D}^{(k)})$ is the lowest error rate achievable by a classifier on the elements of the cluster. Equation (3.1) indicates that we want to split the dataset in regions of decreasing *classification complexity*.

Formally assuring that eq. (3.1) holds is achievable only when we know the true probability distribution of the samples or, equivalently, the optimal classification boundaries.

We provide an heuristic method to obtain a valid splitting for a reasonable number of clusters: *multi-resolution complexity analysis*, or $MRCA$.

Our algorithm can be divided into three steps:

1. Transform $\mathcal{D}$ to a multi-resolution *profile space*;
2. Building the clusters in the profile space;
3. Estimating the complexity of each cluster and sorting them.

3.2.1 Transformation to the Profile Space

The first step of our algorithm is to transform the elements of the dataset under exam into a space in which their inherent complexity becomes more evident. The objective is to construct a space in which observations with similar complexity become close to each other and distant from observations that are either easier or more difficult to classify.

To construct such a space, we evaluate t features for each element $(\boldsymbol{x}_i, y_i)$ of $\mathcal{D}$. We call this t-dimensional space the *(multi-resolution) profile space*. Each feature is estimated through a *probe function*, that has the following form:

$$p_j(\boldsymbol{x}_i, y_i) = p(\boldsymbol{x}_i, y_i; \mathcal{N}_{\mathcal{D}}(\boldsymbol{x}_i, \rho_j)) \quad j = 1, \ldots, t \tag{3.2}$$

where $\mathcal{N}_{\mathcal{D}}(\boldsymbol{x}_i, \rho)$ is the *ρ-neighborhood* of $\boldsymbol{x}_i$, that is, the set of the elements of $\mathcal{D}$ that are at a distance not greater than ρ from $\boldsymbol{x}_i$:

$$\mathcal{N}_{\mathcal{D}}(\boldsymbol{x}_i, \rho) = \{(\boldsymbol{x}_h, y_h) \in \mathcal{D} : \|\boldsymbol{x}_i - \boldsymbol{x}_h\| \leq \rho\} \tag{3.3}$$

Equation (3.2) means that the value of the probe p_j depends on the observation under exam, $\boldsymbol{x}_i$, and of its neighborhood. In our view, p_j should measure the

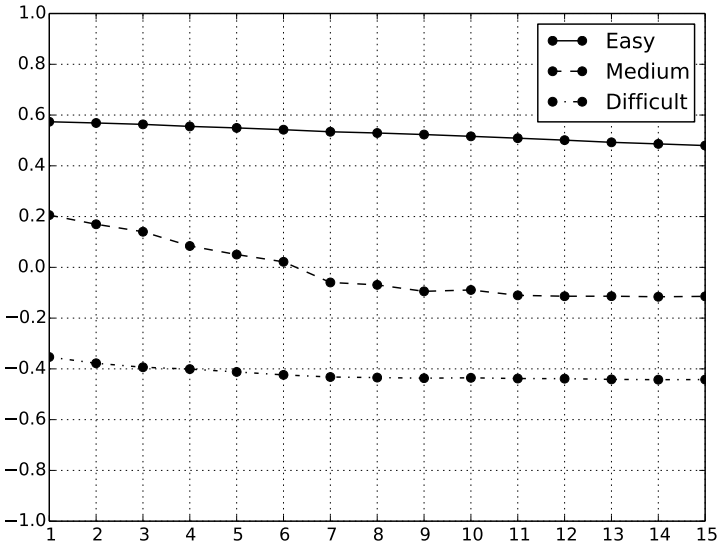


FIGURE 3.1: Profile examples, with $t = 15$.

perceived complexity of classifying $\boldsymbol{x}_i$ when the only information you have is its j-th neighborhood $\mathcal{N}_{\mathcal{D},j}(\boldsymbol{x}_i) = \mathcal{N}_{\mathcal{D}}(\boldsymbol{x}_i, \rho_j)$.

The *profile* $\boldsymbol{p}_i$ of $\boldsymbol{x}_i$ is a vector of probes that represents the perceived complexity of $\boldsymbol{x}_i$ at different resolutions:

$$\boldsymbol{p}_i = \boldsymbol{p}\left(\boldsymbol{x}_i, y_i\right) = \left(p_1\left(\boldsymbol{x}_i, y_i\right), \ldots, p_t\left(\boldsymbol{x}_i, y_i\right)\right) \tag{3.4}$$

By design, we have decided to sort the probes in decreasing resolution order:

$$\rho_1 < \rho_2 < \cdots < \rho_t \tag{3.5}$$

In the experimental section we will show that the radii, ρ_j, needs to be tuned carefully, as they are the most relevant parameter of the hole $MRCA$. We provide an heuristic to choose them in section 3.3.2.

Let us further discuss about the requirements that a function must meet in order to be used as a probe function, as in eq. (3.2). We designed the rest of the system on the following assumptions: $p_j \approx -1$ indicates a very complex zone, in which the information provided by the neighborhood is against the labeling of the current instance; on the other hand, $p_j \approx 1$ indicates an "easy" zone. When $p_j \approx 0$, a "change of complexity" is occurring, similar to the phase change in a dynamic system.

In section 3.3.1 we provide our proposals for the probe function, but $MRCA$ is designed to allow any kind of probe that has the properties just described.

We can finally motivate why we call this new space a *profile space*: each observation $(\boldsymbol{x}_i, y_i)$ is associated with a *complexity profile*, as shown in Figure 3.1, that makes it easy to associate observations with similar complexity.

3.2.2 Clustering Elements in the Profile Space

Once we have calculated the profile for each observation in the dataset, we have a *profile set*:

$$\mathcal{P} = \{\boldsymbol{p}_i = \boldsymbol{p}\,(\boldsymbol{x}_i, y_i) \quad \forall\,(\boldsymbol{x}_i, y_i) \in \mathcal{D}\} \tag{3.6}$$

Let us note that we can always keep a one-to-one relationship between the i-th element of the original dataset $\mathcal{D}$, and the i-th element of the profile set, and between any subset of the profile set and of the original dataset.

Let us cluster the elements in $\mathcal{P}$ into s disjoint subset, either manually (please refer to Section 3.3.4) or using any centroid-based or network-based clustering algorithm. In this work we decided to experiment with a "manual" selection of centroids and with the k-means clustering algorithm, as they are the two most straightforward choices. Both approaches are centroid-based, meaning that the set of clusters is uniquely defined by a set of s *centroids*, by using the following expression:

$$\mathcal{P}^{(k)} = \left\{\boldsymbol{p}_i \in \mathcal{P} : \boldsymbol{p}_c^{(k)} = \underset{\boldsymbol{p}_c}{\operatorname{argmin}} \|\boldsymbol{p}_i - \boldsymbol{p}_c\|\right\} \qquad k = 1, \ldots, s \tag{3.7}$$

That is, each instance $\boldsymbol{p}_i$ is assigned to the cluster associated with the nearest centroid. Equation (3.7) ensures that a fixed set of centroids uniquely identifies the corresponding set of clusters[1].

Let us stress again that we always rely on a one-to-one relationship between the clusters $\mathcal{P}^{(k)}$ in the profile set and the clusters $\mathcal{D}^{(k)}$ in the original dataset:

$$\mathcal{D}^{(k)} = \left\{(\boldsymbol{x}_i, y_i) : \boldsymbol{p}_i \in \mathcal{P}^{(k)}\right\} \tag{3.8}$$

[1]Except for the cases in which equality holds in eq. (3.7) for some $\boldsymbol{p}_i$ and some pair of indexes (h, k): in those cases, a rule to assign $\boldsymbol{p}_i$ to either cluster $\mathcal{P}^{(k)}$ or $\mathcal{P}^{(h)}$ must be devised.

Let us remind that the most important difference between $\mathcal{D}^{(k)}$ and $\mathcal{P}^{(k)}$ is that each cluster in the profile space is a connected set, and eq. (3.7) holds, but its image in original dataset, $\mathcal{D}^{(k)}$, could possibly be disconnected.

3.2.3 Multi-Resolution Index

Projecting the observations to the profile space is an intermediate step needed to cluster together the instances with similar complexity. Once the clusters have been found, we have to measure their classification complexity. In order to do so, we propose the following *Multi-Resolution Index*, or mri for short.

The Multi-Resolution Index is defined both for an instance and for a cluster. The mri of a cluster is simply the average value of the mri of its elements.

Let us first show the expression for the instance-wise Multi-Resolution Index:

$$\text{mri}(\boldsymbol{p}) = \frac{1}{\sum_j w_j} \sum_{j=1}^{t} w_j \frac{1 - p_j}{2} \tag{3.9}$$

For convenience, we use the same term, $\text{mri}(\cdot)$, to refer to the instance-wise index and to the cluster-wise index, that is defined as:

$$\text{mri}(\mathcal{P}^{(k)}) = \frac{1}{|\mathcal{P}^{(k)}|} \sum_{\boldsymbol{p} \in \mathcal{P}^{(k)}} \text{mri}(\boldsymbol{p}) \tag{3.10}$$

The Multi-Resolution Index operates by shifting and rotating the complexities seen at different resolutions so that they range from 0 (lowest) to 1 (greatest), and then averaging these estimates using a different weight for each resolution. In our view, these weights should be higher for high-resolution estimates, as they carry more information about the "local" characteristics of the domain:

$$w_1 > w_2 > \cdots > w_t \tag{3.11}$$

The Multi-Resolution Index ranges between 0 and 1, with 0 indicating the lowest complexity, while 1 indicates a very difficult cluster. To obtain the inequality in eq. (3.1), we need to sort the clusters in decreasing mri order.

In the following, with a small abuse of notation, we will use interchangeably the expressions $\text{mri}(\mathcal{P}^{(k)})$ and $\text{mri}(\mathcal{D}^{(k)})$, to express the fact that the classification

complexity of $\mathcal{D}^{(k)}$ is evaluated by using its projection to the profile space. For the same reason, we could write both $\text{mri}(\boldsymbol{p})$ and $\text{mri}(\boldsymbol{x}_i, y_i)$ to refer to the instance-wise complexity estimate.

3.3 Implementation Details

MRCA is an open approach for estimating the classification complexity of a domain, so that it doesn't strictly depend on the implementation details of its single components. In this section, we discuss *our* implementation of each component, but other approaches and solutions are possible.

3.3.1 Probe Functions

Equation (3.2) gives the general form of a probe function. In this section we present two possible choices for the probe. The first one estimates the complexity as a function of the *local imbalance*, while the second one bases its estimate on the concept of *local linear separability*.

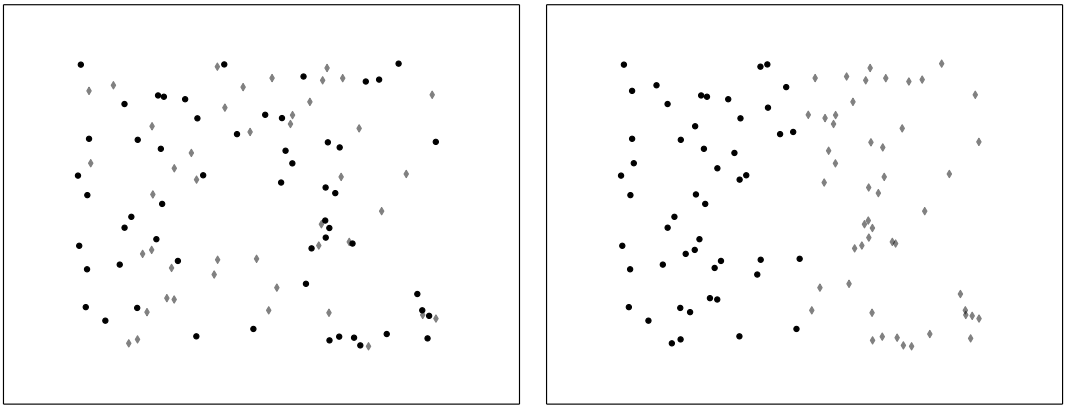
3.3.1.1 Imbalance Probe Function

In order to estimate the local classification complexity, we can look at the *local imbalance* between class labels: if the observation we want to classify, say $(\boldsymbol{x}_i, y_i)$, is surrounded by observations with a different label, $y_h \neq y_i$, it is less likely for a classification algorithm to classify that instance correctly; in this case, we say that there is an imbalance *against* the class y_i. On the other hand, regions with an imbalance *toward* a certain class give more chances for a correct classification. Estimating the local class imbalance can thus provide us with an approximation of the local classification complexity. As a consequence, we can define the *imbalance probe function*:

$$\phi(\boldsymbol{x}_i, y_i; \mathcal{N}_\mathcal{D}) = \frac{c(\mathcal{N}_\mathcal{D}, y_i) - \sum_{h \neq i} c(\mathcal{N}_\mathcal{D}, y_h)}{|\mathcal{N}_\mathcal{D}|} = 2\frac{c(\mathcal{N}_\mathcal{D}, y_i)}{|\mathcal{N}_\mathcal{D}|} - 1 \tag{3.12}$$

Where $c(\mathcal{N}_\mathcal{D}, y)$ is the number of instances in the neighborhood labeled as y:

$$c(\mathcal{N}_\mathcal{D}, y) = |\{(\boldsymbol{x}_i, y_i) \in \mathcal{N}_\mathcal{D} : y_i = y\}| \tag{3.13}$$



(A) Noisy neighborhood (B) Linear boundary neighborhood

FIGURE 3.2: Two neighborhoods with same imbalance, but different complexity.

It is easy to verify that eq. (3.12) ranges between -1 and $+1$, with the lower bound indicating strong imbalance against the class y_i and $+1$ indicating strong imbalance toward the same class.

3.3.1.2 Linear Boundary Probe Functions

The imbalance probe function just described gives a quick estimate of the local complexity, but cannot distinguish between noisy and boundary zones, as shown in Figure 3.2.

In effect, the imbalance probe is proportional to the number of instances in the neighborhood labeled in the same way as the instance being assessed, as in eq. (3.12), but does not consider the pattern with which they are scattered around it.

The probe we propose in this section attempts to overcome this limitation by calculating a local separating hyperplane, using the standard linear classification algorithm trained on the observation found in the neighborhood; once the hyperplane has been found, the probe measures the distance of the instance being assessed from the hyperplane; eventually, if it lies on the "wrong" half-space (i.e., if the local linear classifier would label it in the wrong way), the probe makes the distance negative, so that the probe properties discussed earlier are kept (see section 3.2.1).

We are then saying that the local classification complexity of each instance is proportional to its "signed distance", say d, from the local separating hyperplane. In particular, $d \gg 0$, the instance is deeply inside the correct half-space and the complexity is very low, so that $p \approx 1$; if $d \ll 0$, the instance is in the wrong

half-space, very far from the hyperplane, and the complexity is high, $p \approx -1$; finally, if $d \approx 0$, the complexity is undefined, $p_j \approx 0$, because the instance is near the separating hyperplane.

The mathematical expression for this *linear boundary probe* is:

$$\beta(\boldsymbol{x}_i, y_i; \mathcal{N}_{\mathcal{D}}) = S\left(\boldsymbol{n}^T \boldsymbol{x}_i - \left[\frac{1}{2}\boldsymbol{n}^T \left(\overline{\boldsymbol{x}}_{y_i} + \overline{\boldsymbol{x}}_{\neg y_i}\right)\right]\right) \tag{3.14}$$

Where S is a sigmoid function, $\boldsymbol{n}$ is the normal vector of the separating hyperplane, $\overline{\boldsymbol{x}}_{y_i}$ is the sample mean of the instances labeled y_i in the neighborhood and $\overline{\boldsymbol{x}}_{\neg y_i}$ is the sample mean of the instances *not* labeled y_i. Let us notice that in this case there is no term that takes into account the local imbalance.

We apply a sigmoid function, $S(\cdot)$, to the term in curly brackets so that the probe ranges in $[-1, 1]$. One possible choice for it is the following:

$$S(x) = \frac{2}{1 + e^{-x}} - 1 \tag{3.15}$$

The normal, $\boldsymbol{n}$, is obtained by using Linear Discriminant Analysis on the observations found in the neighborhood:

$$\boldsymbol{n} = \boldsymbol{S}_{pl}^{-1} \left(\overline{\boldsymbol{x}}_{y_i} - \overline{\boldsymbol{x}}_{\neg y_i}\right) \tag{3.16}$$

In which, of course, the sample mean and the pooled covariance matrix are evaluated by using only the observations found in the neighborhood. Let us recall that the pooled covariance matrix between two samples consisting of N_1 and N_2 observations, with covariance matrices $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, is defined as:

$$\boldsymbol{S}_{pl} = \frac{1}{N_1 + N_2 - 2}\Big[\left(N_1 - 1\right)\boldsymbol{S}_1 + \left(N_2 - 1\right)\boldsymbol{S}_2\Big] \tag{3.17}$$

The vector obtained using eq. (3.16) has to be divided by its norm in order to be used in eq. (3.14), because we are measuring the distance between the point $\boldsymbol{x}_i$ and an hyperplane in an affine space.

3.3.2 Resolutions

As we already pointed out at the beginning of the chapter, in presence of an infinite number of observations, the classification complexity of each instance of a domain could be evaluated by simply taking the limit of a given probe function, for the radius of the neighborhood tending to zero:

$$\text{complexity}(\boldsymbol{x}_i, y_i) = \lim_{\rho \to 0} p\Big(\boldsymbol{x}_i, y_i; \mathcal{N}_{\mathcal{D}}(\boldsymbol{x}_i, \rho)\Big) \tag{3.18}$$

But we only have datasets containing a finite number of observations. To cope with this, we introduced the multi-resolution profile discussed above: it makes the complexity estimate, given by the multi-resolution index, more robust with respect to the sparseness of the input space and to the underlying distribution of the data points. Nevertheless, choosing the right values for the radii used in the multi-resolution profile, $(\rho_1, \ldots, \rho_t)$, is one of the most important tasks to get a valuable estimate of the local classification complexity.

We provide some simple heuristics for choosing ρ and t, but a complete analysis of this problem needs further study, and it is listed as one of our future works.

Let us start by noting that the most sensible radius to set is the smallest one, as if we set a value too small for ρ_1, the associated neighborhood could be empty for some (or all) the instances of the dataset[2], and this is definitively something we want to avoid. Let us define the c-nearest neighbor distance, $d^{nn}(\boldsymbol{x}_i, c)$, as the distance between the i-th feature vector in the dataset and its c-th nearest neighbor. Then $\underline{d}^{nn}(c)$ is the sample vector containing the c-nearest neighbor distances of every instance in the dataset. We use $\underline{d}^{nn}$ in the following rule for selecting ρ_1:

$$\rho_1 = \max \underline{d}^{nn}(c_1) \tag{3.19}$$

This equation indicates that we are making sure that $\mathcal{N}_{\mathcal{D}}(\boldsymbol{x}_i, \rho_1)$ contains at least c_1 elements for any observation in the dataset. In some cases, there could be a particular observation $\boldsymbol{x}_i$ that is too far away with respect to the rest of the dataset, and it will make the previous rule unreliable; in these cases, the maximum can be replaced with the sample mean, or with the median, but in both ways we lose the guarantee that every ρ_1-neighborhood contains at least c_1 elements.

[2]Depending on the implementation details for $\mathcal{N}_{\mathcal{D}}$, it will never be empty, as its center will always be at distance zero from itself. However, this doesn't change the point being discussed, that an (almost) empty neighborhood does not convey the necessary information for our purposes.

Once we have set ρ_1, we have several options for setting the remaining resolutions. For example, we can use ρ_1 as step between two consecutive radii:

$$\rho_j = j \cdot \rho_1 \qquad j = 2, \ldots, t \tag{3.20}$$

Or we can generalize eq. (3.19) to get the remaining resolutions:

$$\rho_j = \max \underline{d}^{nn}(c_j) \tag{3.21}$$

$$c_j = c_1 + (j-1) \cdot \frac{c_t - c_1}{t-1} \quad j = 2, \ldots, t-1 \tag{3.22}$$

With eq. (3.22) we propose to calculate the values $c_2, \ldots, c_{t-1}$ by linearly splitting the difference between c_1 and c_t.

An hybrid solution could be also used. Let us suppose we found ρ_1 and ρ_t using eq. (3.21); we can set all other radii by linearly splitting the difference between the two *radius values*, instead of passing through each intermediate c value:

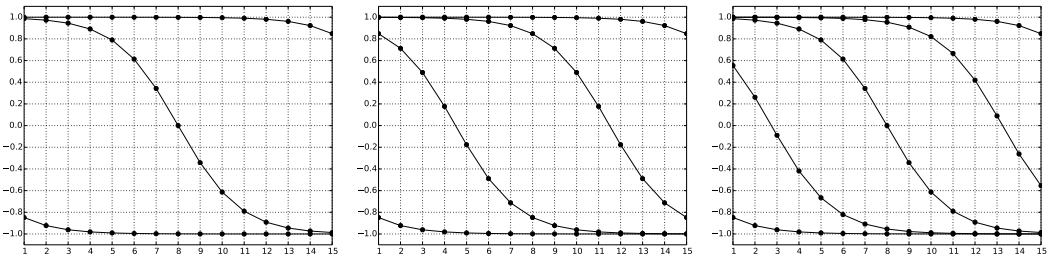
$$\rho_j = \rho_1 + (j-1) \cdot \frac{\rho_t - \rho_1}{t-1} \quad j = 2, \ldots, t-1 \tag{3.23}$$

All the approaches proposed require the user to set c_1, while eq. (3.22) and eq. (3.23) require to set manually also c_t. Sensible values for these two parameters are proposed in Table 3.1, and discussed in the experiments. A deeper investigation on the selection of optimal neighborhood sizes is proposed as future work.

The profile size, t, does not influence too much the performance of $MRCA$. A value between 5 and 25 will typically work just fine, as we will show in the next section.

3.3.3 Feature Space Normalization

The profile transformation, and thus the whole $MRCA$, relies heavily on the concept of neighborhood, so that defining a proper distance metric between the instances in the feature space is fundamental to get valuable complexity estimates. As we decided to use the euclidean distance function in our implementation, a proper feature normalization has to be done before any other calculation occurs. The most straightforward solution is to standardize each feature separately, so that we have a space in which each component has zero mean and unit variance.



(A) $s = 3$ (B) $s = 4$ (C) $s = 5$

FIGURE 3.3: Manual centroids for $t = 15$.

Another approach is to apply the *Mahalanobis transformation* to the instances in the dataset [31]. It standardizes the single features and removes the correlation between them, making the covariance matrix equal to an identity matrix. The transformation takes each instance in the dataset, $\boldsymbol{x}_i$, and produces a corresponding instance $\boldsymbol{z}_i$:

$$\boldsymbol{z}_i = \boldsymbol{S}^{-1/2} (\boldsymbol{x}_i - \overline{\boldsymbol{x}}) \qquad i = 1, \ldots, N \tag{3.24}$$

Where $\boldsymbol{S}$ is the sample covariance matrix and $\overline{\boldsymbol{x}}$ is the sample mean, both calculated on the whole dataset.

Applying the Mahalanobis transformation is not always effective, in particular when we have a large number of features (so that we could have an instable estimate of the inverse square root of $\boldsymbol{S}$) or a small number of observations (so that the estimate of the covariance matrix is not accurate enough). Nevertheless, applying the Mahalanobis transformation to the dataset has the following advantage: to calculate $\boldsymbol{n}$ in the Linear Boundary Probe we do not need anymore an estimate of the pooled covariance matrix, as we know that it is an identity matrix.

3.3.4 Cluster Centers

We propose two ways to define the cluster centers (or centroids). The first one is the well known k-means algorithm. For reference, see for example [11].

The second one is a "manual" method, that is, the centers are chosen using a fixed expression, that does not depends on the profiles in $\mathcal{P}$, but only on the number of clusters and on the number of dimensions of the profile.

Figure 3.3 shows the centroids calculated in the case of 3, 4 and 5 clusters. Let us motivate this approach. While k-means search for clusters that maximize the

inter-group distance, this manual approach ensures that each group is similar to a certain *prototype complexity*. This enables us to generate clusters that are expected to have clearly distinct mri values, while with k-means this is not always the case, as we will show in the experimental results (Section 3.4).

In the experiments, we used the following analytical expression to calculate the j-th component of the k-th cluster center:

$$p_{c,j}^{(k)} = S\left\{-10\left[\frac{j-1}{t-1} - 1.5\frac{k-1}{s-1} + 0.25\right]\right\} \qquad \begin{matrix} j = 1,\ldots,t \\ k = 1,\ldots,s \end{matrix} \tag{3.25}$$

Where S is the sigmoid function defined in Eq. 3.15.

3.3.5 Weights of the Multi-Resolution Index

The mri synthesizes the information contained in a profile $\boldsymbol{p}$ by averaging its components, p_j, weighted according to the resolution with which they have been calculated. In our view, the highest resolutions contain the most information about the classification complexity of a given instance, and then they should get an higher weight. We use the following expression to calculate each weight w_j:

$$w_j = 1 - \frac{j-1}{t} \qquad j = 1,\ldots,t \tag{3.26}$$

3.4 Experiments

We verified the performance of $MRCA$ on 34 UCI datasets[3]. We applied standardization to the feature space of each dataset. As discussed in section 3.3.3, we did not choose to use the Mahalanobis transformation as many datasets have too many dimensions or too few samples. We then preferred to apply a simpler transformation to make the results comparable between datasets.

We tested both the Imbalance Probe and the Linear Boundary Probe, and 5 different profile sizes: 5, 10, 15, 20, and 25 dimensions.

[3]We had computational problems with the datasets *splice* and *letter*, because of the size of the former and of the number of features of the latter. This will be addressed in the future.

Parameter name	Values
Probe function (p_j)	Imbalance Probe (Eq. 3.12) Linear Boundary Probe (Eq. 3.14)
Profile size (t)	5, 10, 15, 20, 25
Smallest neighborhood (c_1)	5%, 10%, 15%, 20%, 25%
Largest neighborhood (c_t)	40%, 45%, 50%, 55%, 60%

TABLE 3.1: Profile configuration values.

To select the resolutions, we opted for the rule expressed in eq. (3.23) together with eq. (3.22), but the max function has been replaced with the sample mean. We ran the analysis with five different choices for c_1 and for c_t, resulting in 25 different configurations for the pair (c_1, c_t). As we used datasets with a very different number of instances among them, it would not be wise to express c as an absolute value. Instead, we opted for using a number between 0 and 1, indicating the percentage of dataset instances to consider. The values chosen for c_1 and c_t are shown in Table 3.1.

The cluster centers have been calculated using the k-means algorithm and the "manual" procedure described in Section 3.3.4. We experimented with five different values for the number of clusters, between 2 and 6. The interpretation of the clustering depends slightly on the number of clusters selected. Clearly, when we use two clusters we aim at splitting the dataset into one "easy" and one "difficult" part. When using three clusters, we add an additional "intermediate" region. The use of more clusters adds more levels of complexity between the easiest and the most difficult regions.

In order to validate the results, we had to define a way to evaluate the relationship between the estimated complexity of each cluster, $\text{mri}(\mathcal{D}^{(k)})$, and its "actual" complexity, that we described at the beginning of Section 3.2 as the lowest error rate achievable by a classifier on the instances of the cluster. However, as already discussed in the previous chapter, obtaining a sensible estimate of the lowest error rate for a given dataset is a hard task, so that we decided to replace it with the error rate obtained by a set of classifier ensembles. The classification error rate has been obtained by running a 5-fold cross-validation on each dataset. The classifier ensembles used are: a Bagging Ensemble, a Random Forest and an AdaBoost Ensemble. Each of them has been trained with the default parameters of *scikit-learn* [32] with an ensemble size of 100 classifiers.

We have to compare two samples, more or less as we have already described in the previous chapter. In the current case, each dataset is split into s clusters, so that we produce two samples of s observations for each dataset: a sample of mri values and one of error rates. We compare the two samples using the linear correlation coefficient: high positive values of correlation indicates that the splitting is effective, that is, it properly separates the difficult zones of the dataset from the easy ones.

3.4.1 Experimental Results

The quantity of data produced by our experiments is very large. By testing two different probes, each five profile sizes, five values of c_1 and five more values of c_t, we obtained 250 different estimates for the mri. Moreover, we ran the experiments on 34 datasets, using two different clusterers and five values for the number of clusters, for a total of 340 clustering configurations. This means that the experiments generated $250 \times 340 = 85000$ clusterings, 2500 for each dataset. Each clustering produced a different mri sample, and each mri sample has been compared with three different error rate samples (each of them produced by a different classifier ensemble), resulting in 7500 comparisons *per dataset*, 255000 in total.

We opted to present here the most valuable results, in order to show the effectiveness of the approach and to indicate its limitations too. Further discussion will be carried on as future work.

In Table 3.2 we present a synthesis of the performances of all the profile configurations. The table is constructed as follows. Each value represents the number of times that the corresponding profile configuration has a squared linear correlation with the error rate of the Random Forest greater than or equal to 64%. We use the squared linear correlation as it is a measure of the variation on the error rate *explained* by the variation in the mri. Then, each value represents the number of "successes" of the corresponding configuration.

The maximum possible number of successes is 340, that is, on every clustering configuration, that particular profile configuration has at least 80% correlation with the error rate sample (80% squared equals 64%).

We built similar tables to compare the mri samples with the Bagging error rate and the AdaBoost error rate. They are very similar to the one just presented, so they are not listed here. You can see them in section 3.6.

		Imbalance Probe					Linear Boundary				
c_1	c_t	5	10	15	20	25	5	10	15	20	25
	40%	257	260	258	258	256	181	172	175	176	173
	45%	257	256	261	264	263	176	173	177	174	172
5%	50%	260	258	256	258	257	177	179	178	175	177
	55%	256	262	260	256	256	179	179	179	176	176
	60%	255	257	259	255	257	180	174	185	175	179
	40%	246	243	244	245	244	161	171	173	169	168
	45%	243	246	244	248	250	175	167	172	172	170
10%	50%	249	246	249	255	254	177	173	177	169	172
	55%	245	245	252	248	254	178	175	165	172	173
	60%	245	244	246	255	254	166	166	170	171	176
	40%	243	237	240	242	248	179	169	170	175	176
	45%	248	245	242	248	246	175	168	171	176	179
15%	50%	249	244	244	247	250	174	169	172	167	169
	55%	248	250	252	249	246	173	169	169	174	167
	60%	243	242	244	244	248	168	171	166	162	166
	40%	229	236	229	234	231	171	168	162	166	164
	45%	232	233	234	229	235	166	167	167	165	166
20%	50%	230	235	238	238	238	171	171	163	156	165
	55%	236	232	238	236	240	158	171	167	164	164
	60%	241	236	242	240	241	167	167	167	165	166
	40%	221	227	225	227	231	162	163	158	163	164
	45%	226	224	228	228	226	166	169	167	167	165
25%	50%	220	229	230	234	228	175	169	166	168	166
	55%	217	230	232	231	233	161	172	171	164	169
	60%	227	235	240	239	240	168	174	167	171	168

TABLE 3.2: Number of positive results for each profile configuration. Classifier: Random Forest

Let us discuss the overall results. First of all, it is clear that even if less powerful, the Imbalance Probe proves to be way better than the Linear Boundary Probe in identifying regions of different complexity. In our view, this is due to the fact that the added complexity of calculating a Linear Boundary for each neighborhood reduces the performance of the Linear Boundary probe on smaller datasets, and on datasets with a large number of features.

From the table we can also evince that the smallest neighborhood has to be kept as small as possible: the number of successes in the first block of the table are

		Imbalance Probe					Linear Boundary				
c_1	c_t	5	10	15	20	25	5	10	15	20	25
	40%	88.1	88.2	88.4	88.6	88.8	88.2	88.7	89.4	88.4	89.1
	45%	88.0	88.9	88.4	88.5	88.5	88.0	88.5	89.1	89.5	89.3
5%	50%	88.6	88.8	89.0	88.6	89.0	88.3	87.8	88.5	89.5	88.5
	55%	88.3	89.1	88.8	88.9	89.2	88.5	88.5	88.2	88.6	88.7
	60%	88.6	89.3	88.5	89.1	89.3	88.2	89.2	87.8	88.9	89.2
	40%	87.7	88.9	89.1	89.2	88.8	88.2	88.7	88.1	89.0	89.6
	45%	89.0	89.3	89.1	88.5	89.1	87.8	88.4	88.6	89.6	88.9
10%	50%	88.5	89.1	89.1	88.7	88.9	87.4	88.4	88.5	89.0	89.4
	55%	88.5	89.0	89.2	89.4	89.1	87.7	88.0	88.3	89.3	88.7
	60%	89.2	89.3	89.7	89.4	89.1	87.4	88.3	88.0	87.8	88.4
	40%	88.2	88.6	88.8	89.2	88.5	87.6	88.7	88.5	88.9	89.4
	45%	87.5	88.7	89.1	88.8	88.9	88.2	89.0	89.0	89.2	89.1
15%	50%	87.8	88.8	89.5	88.5	88.6	88.6	88.4	88.6	89.5	88.9
	55%	87.5	88.8	88.6	88.9	89.2	87.6	87.6	87.7	88.7	89.3
	60%	88.5	89.4	88.9	89.3	89.2	87.8	88.8	87.9	89.2	88.7
	40%	88.6	88.2	89.2	89.1	89.4	88.2	88.1	89.7	89.5	89.6
	45%	89.2	89.0	88.9	89.4	89.4	88.9	89.0	88.7	89.8	89.6
20%	50%	88.7	88.9	88.9	89.2	89.7	87.9	88.5	88.5	89.0	88.9
	55%	88.8	89.3	89.1	89.3	89.8	89.6	88.1	88.4	88.3	88.7
	60%	88.3	89.4	89.3	89.5	89.5	88.6	88.5	88.3	89.0	88.2
	40%	89.3	88.1	88.9	89.5	89.0	89.4	88.4	88.9	88.1	88.7
	45%	88.6	88.9	89.6	89.2	89.4	88.8	88.9	88.4	88.4	89.0
25%	50%	89.0	89.2	89.5	89.5	89.5	88.5	88.9	88.2	88.5	89.0
	55%	90.0	88.9	88.8	89.5	88.9	89.0	88.7	87.7	88.9	87.1
	60%	88.2	88.5	88.8	88.7	88.6	88.0	88.3	88.6	88.3	87.6

TABLE 3.3: Number of positive results for each profile configuration. Classifier: Random Forest

slightly greater than the remaining ones (on average, using $c_1 = 5\%$ adds around 10 successes with respect to all other configurations).

A different perspective on the same data is given by Table 3.3 where it is shown the average (squared) correlation on every configuration considered as success in the previous table. It is evident that on a large part of the clustering configurations, the correlation between the mri and the error rate is well over the threshold. This happens using both the Imbalance Probe and the Linear Boundary probe, indicating that the latter works on a smaller number of clustering configurations, but, when it works, it does not reduce the effectiveness of $MRCA$.

We also present the results obtained on each dataset when using the two best profile configurations, one with the Imbalance Probe and one with the Linear Boundary Probe. As in the other cases, the comparison is against the Random Forest error rate.

Table 3.4 shows the results obtained when calculating the clusterings and the mri using a profile with $t = 15$, $c_1 = 5\%$ and $c_t = 45\%$. The clustering algorithm is the "manual" procedure described earlier. The circle next to each value indicates that $MRCA$ has succeeded in ranking the clusters in increasing classification complexity. The table clearly shows that when the number of clusters is small, $MRCA$ is almost always able to achieve its goal, but it remains effective on two thirds of the datasets even when using 6 clusters.

Let us highlight a particular result. In the *audiology* and *rootstock* datasets, you can notice that there are high values for the squared correlation, but without a circle next to them. This is due to the fact that in those cases the correlation happened to be *negative*.

The same "inversion" effect becomes more frequent when using the Linear Boundary Probe, as shown in Table 3.5. The datasets *credit-a*, *audiology*, *kr-vs-kp*, *rootstock*, *waveform* and *zoo* are affected by such "inversion". Investigation of the reasons behind this particular effect will be carried on in the future.

Table 3.6 shows the results obtained when using the best possible configuration for each dataset. We present this table to demonstrate that, with accurate tuning, $MRCA$ can be effective on any dataset and with any number of clusters. In effect, with two or three clusters, $MRCA$ had problems with only one dataset (*audiology*), and even with six clusters, $MRCA$ failed on only four datasets out of 34.

We terminate our discussion with the plots of the mri samples compared with the Random Forest error rate samples, for each dataset and for each number of clusters: Fig. 3.4 – 3.10. For each clustering configuration, we have plotted the best result. The mri is on the x-axis, and the error rate on the y-axis. At the top of each figure it is reported the squared linear correlation between the two samples, together with a full circle if the plot represents a success. You can easily see that in the majority of the cases there is an almost perfect correlation between the two samples, so that they generate a straight line. Only in a few cases you can notice some minor imperfection in the relationship.

	Number of clusters				
Dataset	2	3	4	5	6
anneal-orig	100.00•	70.55•	72.88•	57.97	33.36
anneal	100.00•	75.33•	68.20•	52.51	44.21
audiology	100.00	96.06	26.94	36.52	11.69
autos	100.00•	64.61•	74.76•	68.73•	49.73
balance-scale	100.00•	95.88•	88.24•	88.47•	88.80•
breast-cancer	100.00•	99.78•	82.33•	91.20•	75.77•
breast-w	100.00•	84.71•	82.85•	86.00•	79.33•
colic-orig	100.00•	7.79	13.22	71.91•	2.25
colic	100.00•	93.06•	85.58•	69.15•	81.07•
credit-a	100.00•	91.64•	94.07•	90.67•	81.71•
credit-g	100.00•	94.61•	85.32•	82.91•	90.47•
diabetes	100.00•	96.23•	97.10•	95.51•	96.23•
glass	100.00•	63.68	64.57•	66.20•	53.89
heart-c	100.00•	98.12•	97.02•	95.88•	91.77•
heart-h	100.00•	97.82•	98.64•	92.29•	96.31•
heart-statlog	100.00•	99.31•	92.42•	89.60•	96.52•
hepatitis	100.00•	97.63•	96.35•	90.94•	71.45•
hypothyroid	100.00•	86.67•	71.42•	4.60	2.61
ionosphere	100.00•	89.34•	80.81•	84.13•	86.66•
iris	100.00•	90.33•	85.69•	80.20•	91.66•
kr-vs-kp	100.00•	80.44•	67.79•	59.02	58.07
labor	100.00•	0.00	0.00	0.00	0.00
lymph	100.00•	95.36•	89.32•	80.69•	78.99•
primary-tumor	100.00•	93.84•	92.17•	81.31•	86.85•
rootstock	100.00	31.83	4.10	14.03	1.67
segment	100.00•	79.26•	67.80•	59.93	62.93
sick	100.00•	96.62•	89.61•	83.13•	85.40•
sonar	100.00•	79.89•	86.28•	62.59	69.91•
soybean	100.00•	81.91•	77.53•	72.88•	67.74•
vehicle	100.00•	76.63•	82.98•	78.48•	74.32•
vote	100.00•	99.96•	96.66•	89.81•	83.84•
vowel	100.00•	100.00•	100.00•	100.00•	100.00•
waveform	100.00•	91.50•	86.94•	83.03•	81.49•
zoo	100.00•	85.21•	68.80•	6.61	7.35
Positive results	32	29	30	24	22

TABLE 3.4: Results for Imbalance Probe, $t = 15$, $c_1 = 5\%$, $c_t = 45\%$. Clusterer: Custom Centroids. Compared classifier: Random Forest.

	Number of clusters				
Dataset	2	3	4	5	6
anneal-orig	100.00•	98.41•	5.94	6.02	3.64
anneal	100.00•	73.53•	67.65•	64.91•	61.49
audiology	100.00•	100.00	23.69	92.78	29.58
autos	100.00•	100.00•	85.19•	41.01	46.51
balance-scale	100.00•	100.00•	80.20•	83.58•	84.75•
breast-cancer	100.00•	86.58•	75.62•	74.73•	67.75•
breast-w	100.00•	9.56	25.08	33.63	0.13
colic-orig	100.00•	100.00•	100.00•	11.16	7.66
colic	100.00•	76.32	0.00	0.04	2.15
credit-a	100.00	100.00	3.24	0.96	1.27
credit-g	100.00•	87.68•	90.14•	80.66•	78.83•
diabetes	100.00•	95.33•	84.79•	69.16•	82.18•
glass	100.00•	100.00•	3.07	31.10	12.37
heart-c	100.00•	99.38•	77.75•	41.72	93.44•
heart-h	100.00•	100.00•	45.76	14.78	44.84
heart-statlog	100.00•	94.90•	87.63•	97.87•	73.54•
hepatitis	100.00•	100.00•	86.31•	95.76•	94.14•
hypothyroid	100.00•	93.03•	7.71	2.34	9.18
ionosphere	100.00•	1.44	81.98•	72.42•	75.88•
iris	100.00•	100.00•	92.55•	67.19•	86.87•
kr-vs-kp	100.00	100.00	99.77	13.22	53.91
labor	100.00•	100.00•	0.00	0.00	0.00
lymph	100.00•	100.00	0.04	38.47	16.29
primary-tumor	100.00•	88.50•	23.64	7.79	78.89•
rootstock	100.00•	100.00	46.06	13.89	46.80
segment	100.00•	100.00•	16.73	9.18	20.51
sick	100.00•	82.62•	71.27•	64.69•	67.52•
sonar	100.00	100.00•	72.89•	78.10•	68.23•
soybean	100.00•	100.00•	69.91•	96.57•	97.72•
vehicle	100.00•	100.00•	72.67•	64.72•	56.94
vote	100.00•	100.00•	90.76•	83.71•	41.53
vowel	100.00•	96.29•	87.64•	77.31•	2.77
waveform	100.00	92.26	83.66	76.63	92.96
zoo	100.00	100.00	92.15	6.87	5.95
Positive results	29	24	18	15	13

TABLE 3.5: Results for Linear Boundary Probe, $t = 15$, $c_1 = 5\%$, $c_t = 60\%$. Clusterer: Custom Centroids. Compared classifier: Random Forest.

	Number of clusters				
Dataset	2	3	4	5	6
anneal-orig	100.00•	99.90•	95.76•	83.55•	75.50•
anneal	100.00•	99.60•	88.69•	82.44•	65.55•
audiology	100.00•	29.27	26.22	0.39	31.22
autos	100.00•	100.00•	99.99•	100.00•	100.00•
balance-scale	100.00•	100.00•	96.71•	94.31•	90.93•
breast-cancer	100.00•	100.00•	99.40•	97.58•	98.83•
breast-w	100.00•	100.00•	100.00•	98.02•	95.34•
colic-orig	100.00•	100.00•	100.00•	98.94•	74.68•
colic	100.00•	99.97•	96.74•	89.66•	81.07•
credit-a	100.00•	93.09•	94.64•	91.69•	81.71•
credit-g	100.00•	100.00•	96.87•	95.50•	93.78•
diabetes	100.00•	99.27•	97.68•	96.21•	96.68•
glass	100.00•	100.00•	77.57•	74.96•	71.76•
heart-c	100.00•	100.00•	99.68•	99.47•	99.27•
heart-h	100.00•	100.00•	99.81•	99.93•	97.11•
heart-statlog	100.00•	100.00•	99.94•	99.74•	99.00•
hepatitis	100.00•	100.00•	100.00•	100.00•	99.76•
hypothyroid	100.00•	100.00•	77.98•	84.62•	70.63•
ionosphere	100.00•	99.99•	99.94•	90.30•	89.33•
iris	100.00•	100.00•	99.86•	94.48•	94.12•
kr-vs-kp	100.00•	81.01•	71.52•	62.60	62.33
labor	100.00•	100.00•	0.00	0.00	0.00
lymph	100.00•	99.08•	92.06•	84.89•	82.69•
primary-tumor	100.00•	100.00•	99.98•	98.75•	94.00•
rootstock	100.00•	100.00•	60.39	0.13	3.16
segment	100.00•	100.00•	68.34•	63.11	62.93
sick	100.00•	100.00•	100.00•	100.00•	92.27•
sonar	100.00•	100.00•	99.95•	99.92•	96.30•
soybean	100.00•	100.00•	100.00•	98.95•	99.57•
vehicle	100.00•	100.00•	98.99•	82.35•	79.99•
vote	100.00•	100.00•	99.37•	97.15•	98.91•
vowel	100.00•	100.00•	100.00•	100.00•	100.00•
waveform	100.00•	92.20•	87.70•	96.31•	92.35•
zoo	100.00•	95.65•	96.84•	95.81•	88.13•
Positive results	34	33	31	29	29

TABLE 3.6: Best results for each dataset and number of clusters. Clusterer: Custom Centroids. Compared classifier: Random Forest.

The most important information that emerges from the plots is that the clusters identified as most difficult, in some cases happen to have an error rate higher than 50%, even for binary problems (see for example the plots for *heart-c*).

3.5 Conclusions

We developed a novel method to assess the *local* classification complexity of a domain, and proved its effectiveness by experimenting with it on a wide range of datasets. In particular, $MRCA$ is always able at identifying the "hard" region of the dataset and to separate it from the "easy" region, that is, it works perfectly with 2-3 clusters. The use of $MRCA$ with more clusters needs a more precise tuning, but we showed that some profile configurations achieve almost perfect performances on a wide range of cases. In particular, we suggest to use the following one: $t = 15$, $c_1 = 5\%$ and $c_t = 40\%$, together with the Imbalance Probe and the "manual" clustering procedure described in section 3.3.4.

However, a lot of work can still be done to improve $MRCA$. Moreover, we expect its use on many applicative domains. Let us highlight the main directions of work for the nearest future:

- a training algorithm that also learn the optimal parameters to build the profile space;
- an explanation for the "inversion" effect that has been found when using the Linear Boundary Probe;
- the use of the Correlation Scores presented in the previous chapter as probe functions;
- the use of $MRCA$ as part of the training algorithm of a classifier: in particular, $MRCA$ could be used to generate the weights for a Boosting-like ensemble.
- the knowledge of the local classification complexity of a given instance can be used as an heuristic for a dynamic ensemble selection algorithm.

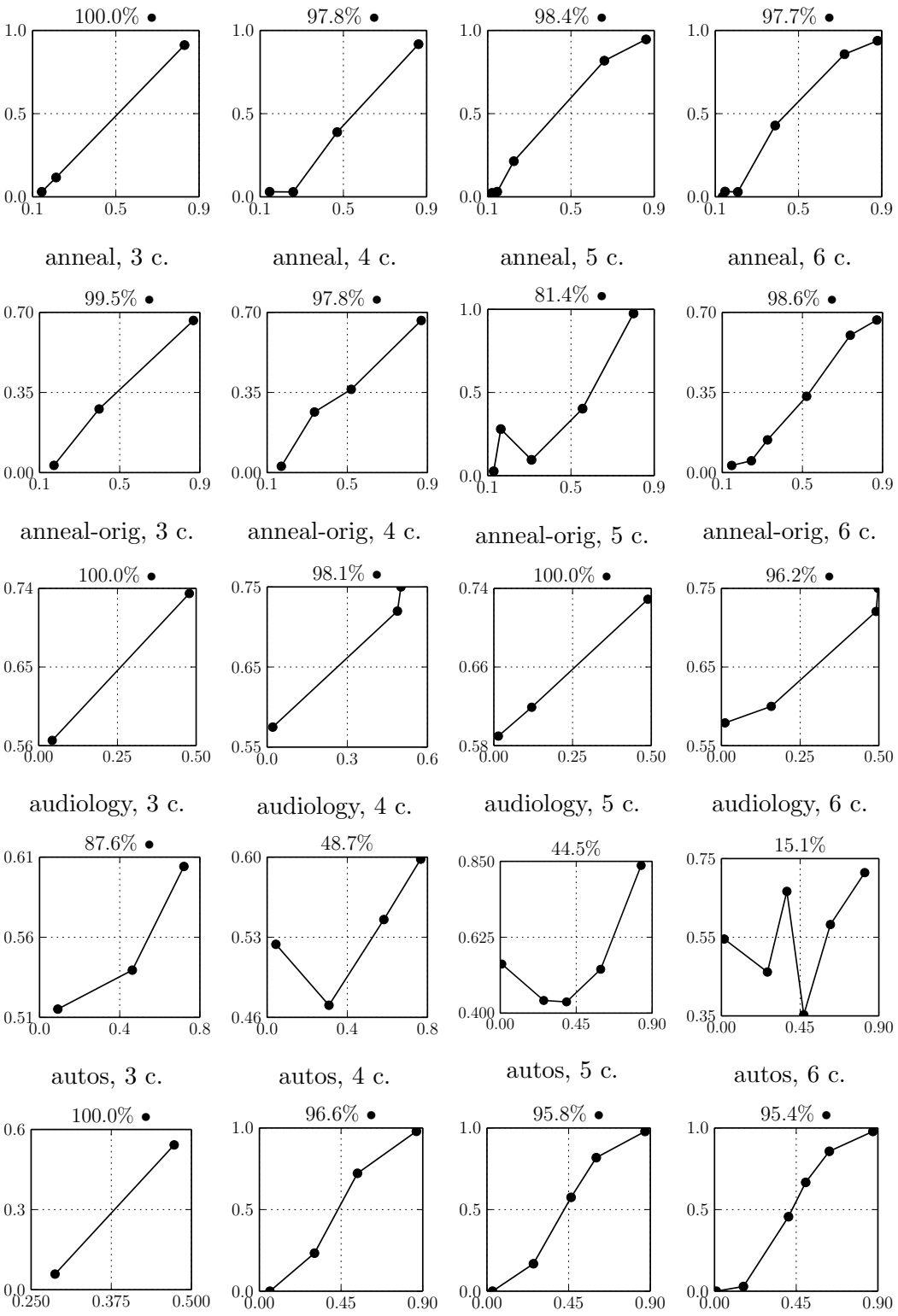


FIGURE 3.4: Comparison plots (part 1)

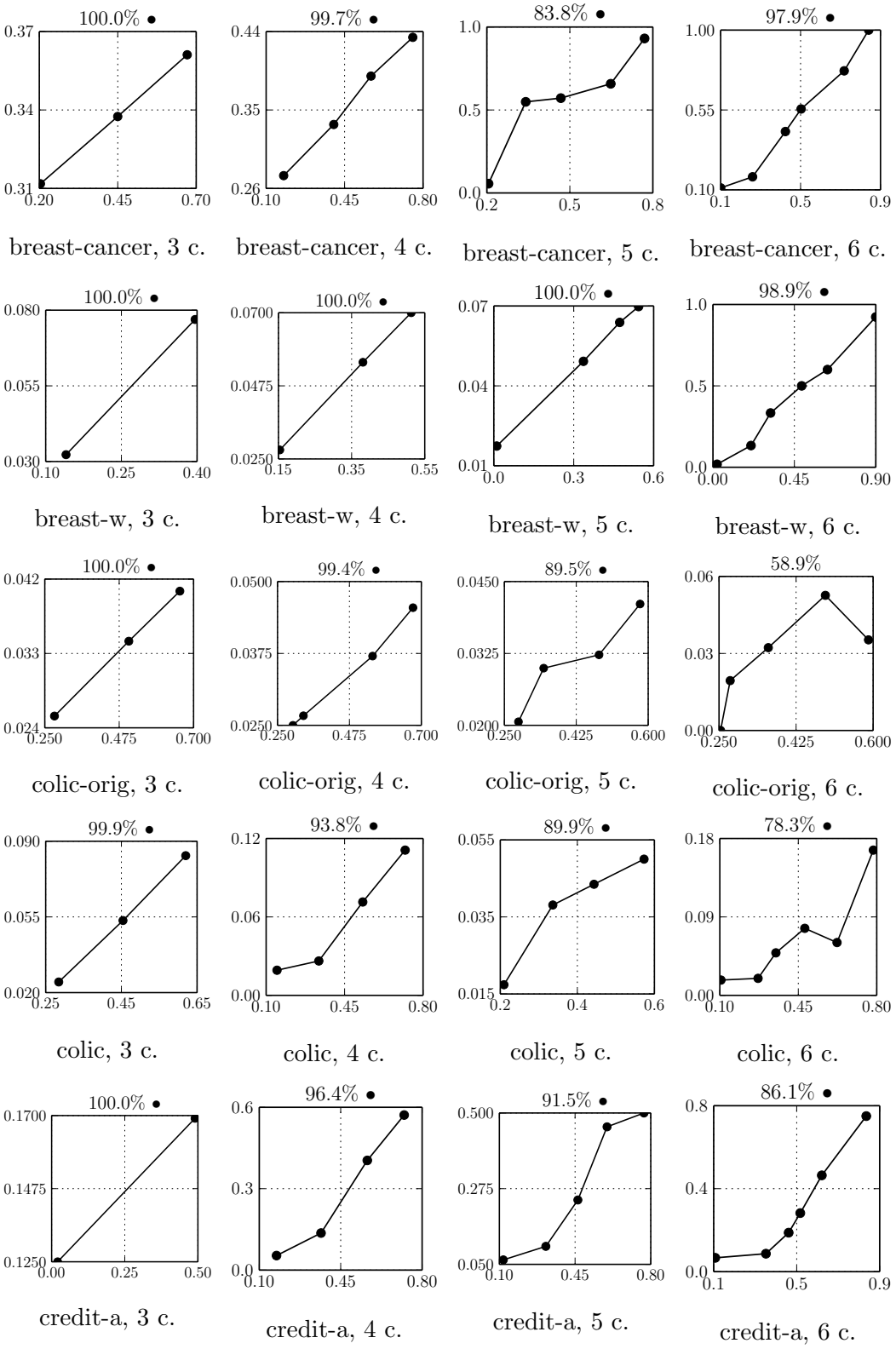


FIGURE 3.5: Comparison plots (part 2)

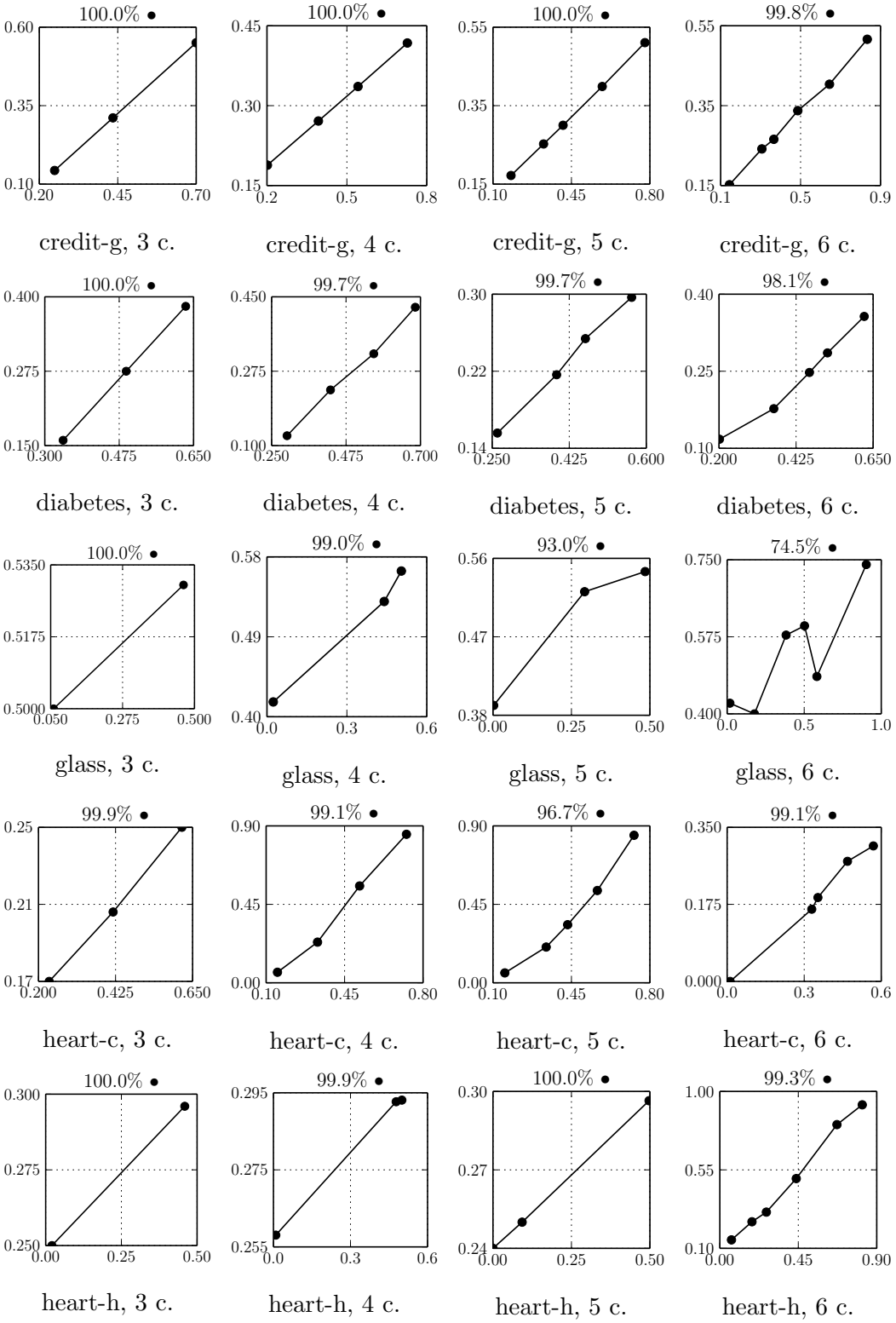


FIGURE 3.6: Comparison plots (part 3)

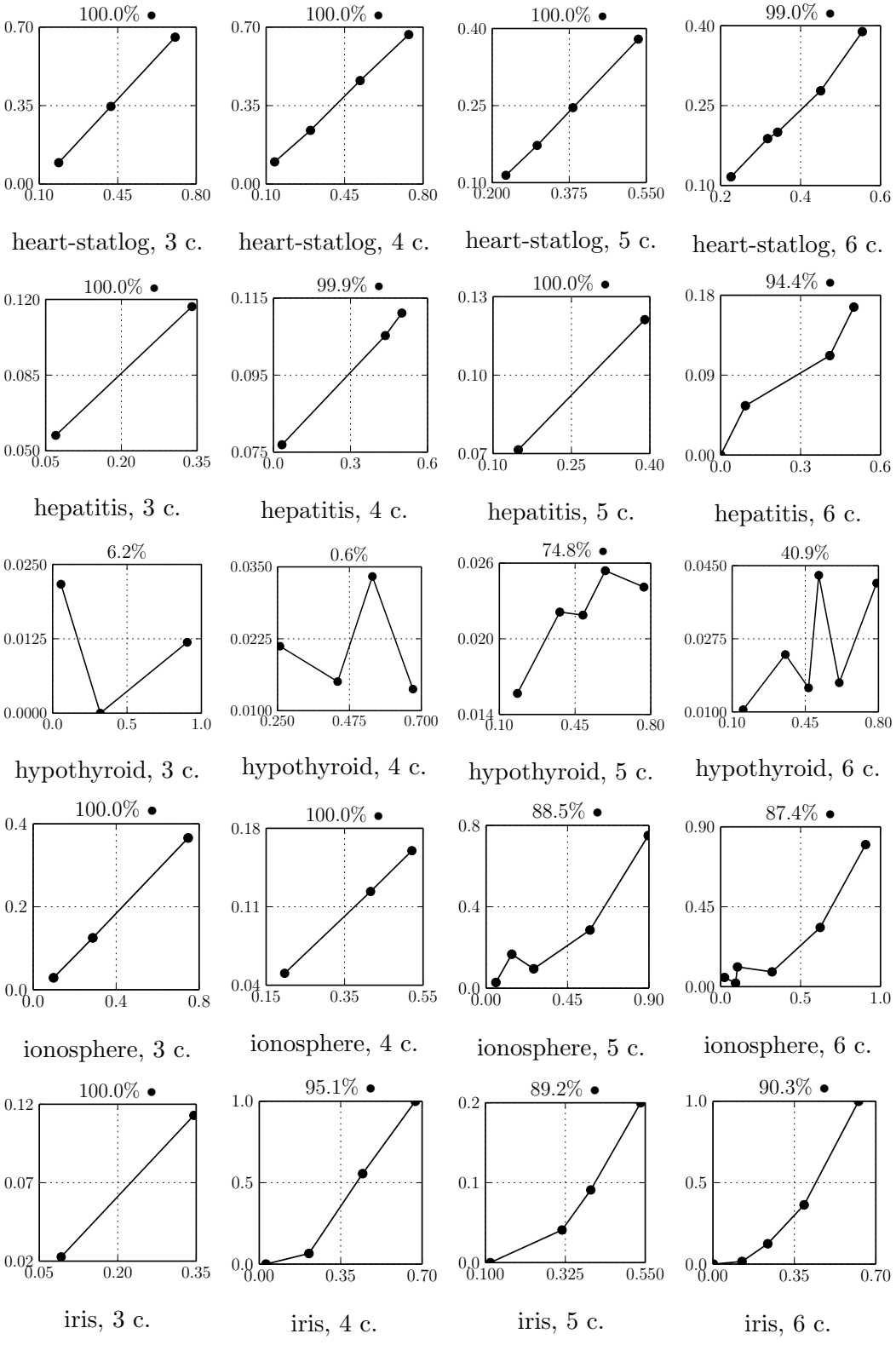


FIGURE 3.7: Comparison plots (part 4)

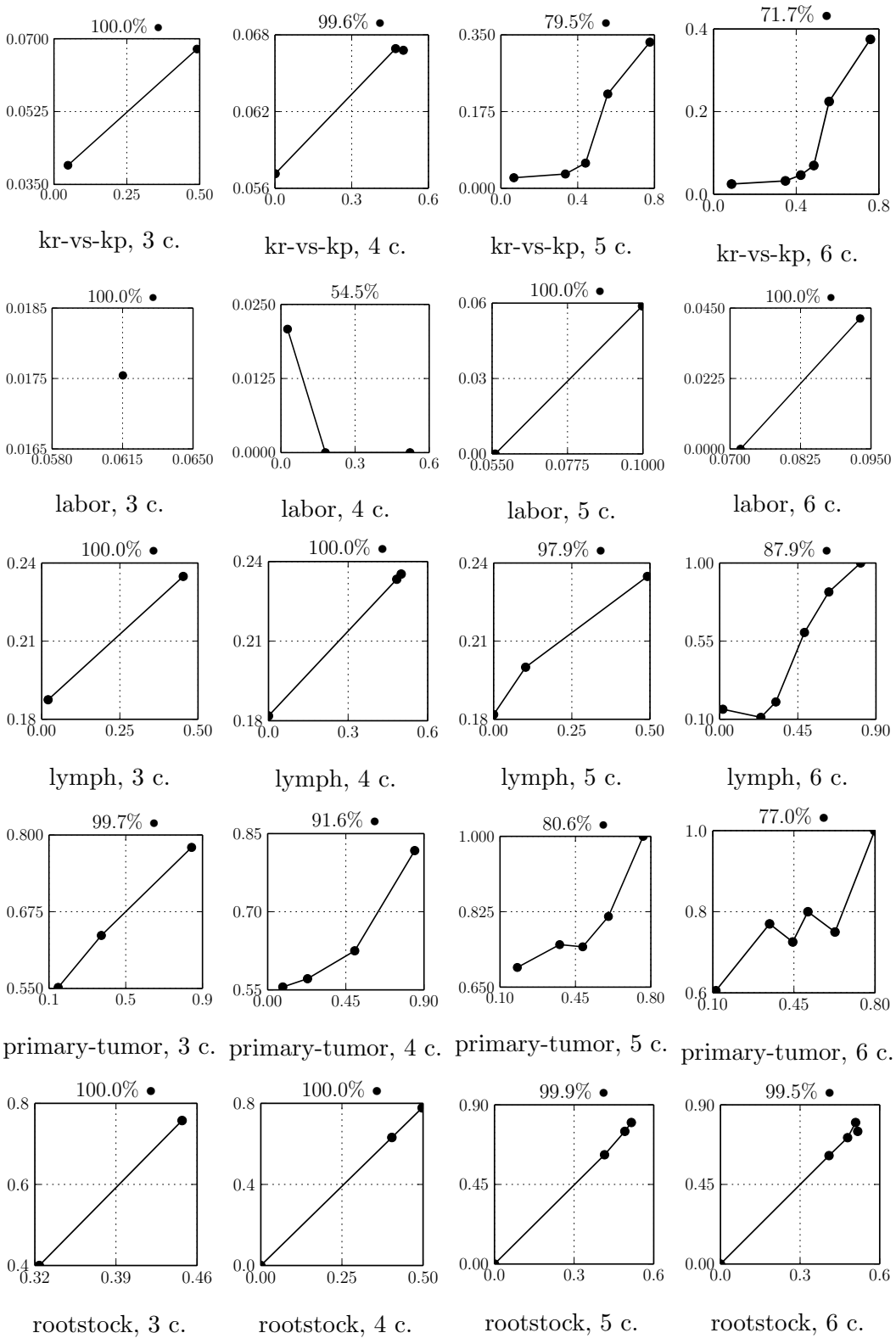


FIGURE 3.8: Comparison plots (part 5)

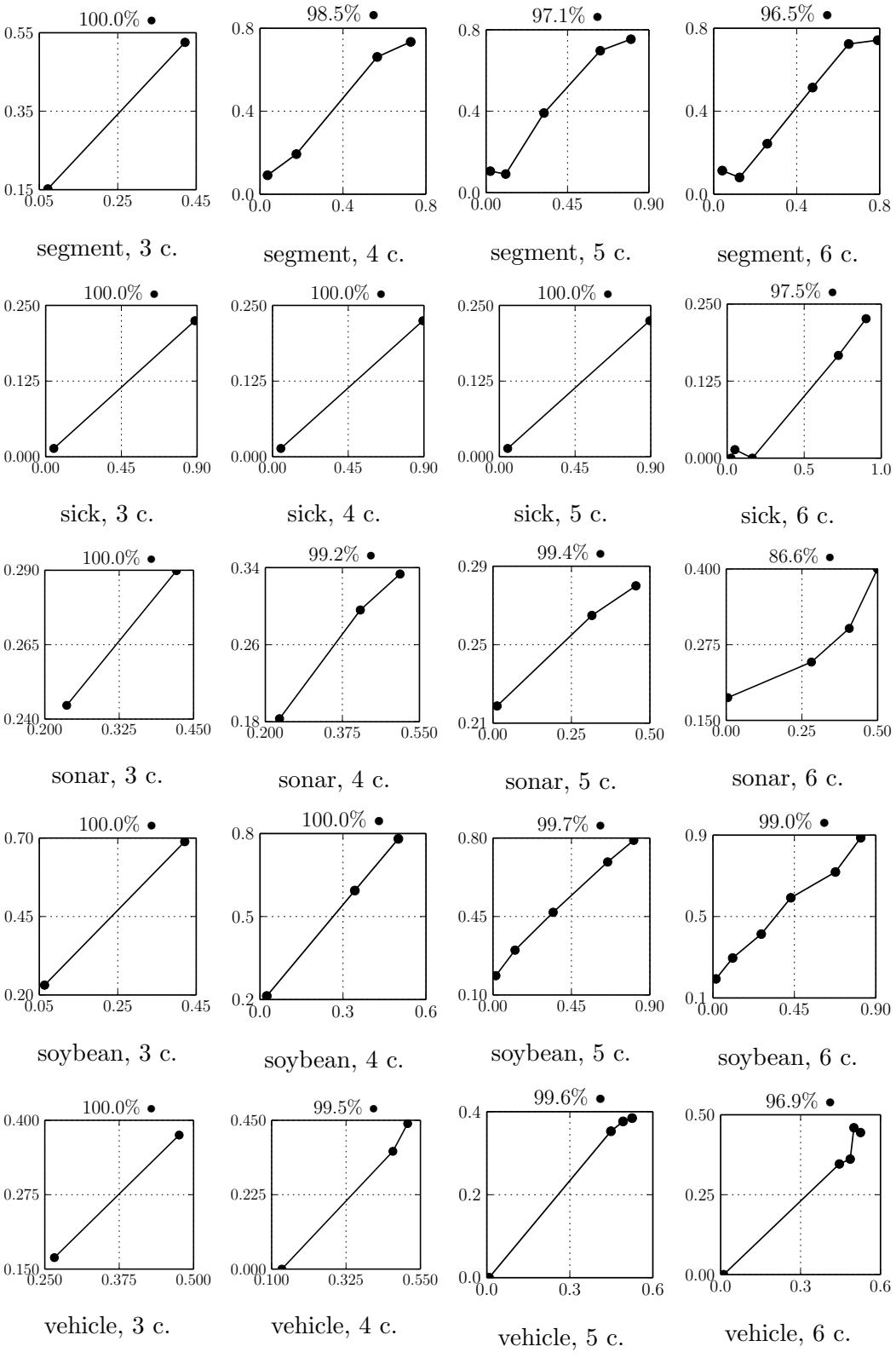


FIGURE 3.9: Comparison plots (part 6)

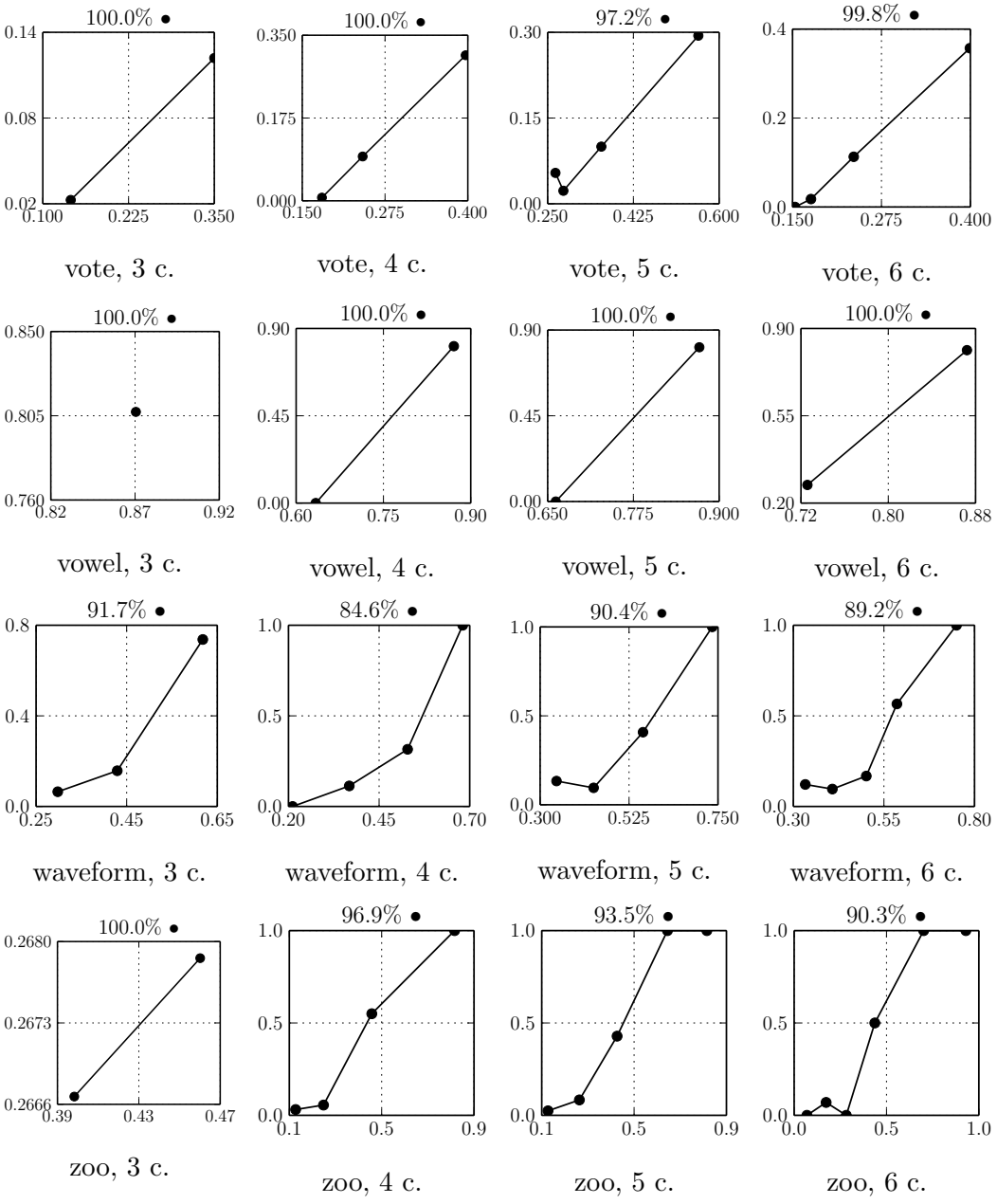


FIGURE 3.10: Comparison plots (part 7)

3.6 Additional Tables and Figures

This section will be completed in the final version, sorry for the inconvenience. It will contain the tables with the comparison with Bagging and AdaBoost, and the plots for the best performing profile configuration.

Chapter 4

Forest of Local Trees: a Novel Ensemble Method

Methods for classifier combination have been very popular for many years [33], [34], [35]. The reasons behind their popularity are their conceptual simplicity and their top-level performances [11].

The basic idea that drives *classifier ensembles*, the term with which the literature refers to all methods for classifier combination, is that a pool of *weak learners* can achieve a better performance than a single *strong* learner. Let us explain the idea with an analogy: a very intelligent person and a group of one hundred normal people are asked to study a very difficult new discipline from its reference book; a judge asks questions about that discipline, and who gives most correct answers wins. The group decides to give to each question the answer most agreed upon, while the intelligent person must rely on himself alone. Who will win, the lonely very intelligent person, or the very large group of normal people?

A classifier ensemble, to say it with the help of the analogy, is the group of normal people. Take each person of the group separately, and the intelligent guy will always win over him. But take them as a whole, and either the intelligent person has a deal with the judge, or his chances of victory drop dramatically.

Of course, the theoretical foundations of classifier combination go far beyond the simple explanation just given, and a lot of effort has been put to quantify how much performance is gained when using an ensemble instead of a single classifier.

Let us briefly recall a few concepts that will be useful to understand the motivation behind classifier ensembles in general and behind our algorithm. For a more complete review on classifier ensembles, see for example [36], [11], [37], [38], [39].

Let us suppose we want to produce an estimate, $\hat{y}$, for an instance $\boldsymbol{x}$. $\hat{y}$ estimates a quantity related to $\boldsymbol{x}$: it can be a class label, or a scalar, or a vector. The "true" quantity that we want to approximate, $y(\boldsymbol{x})$, is in general a non deterministic function of $\boldsymbol{x}$, that comes from a random variable that we will call $Y(\boldsymbol{x})$, or just Y. We aim at identifying an *optimal estimate for* y, y_*: it is the estimate that minimizes the expected value of a fixed loss function (this expected value is calculated over the probability distribution of Y). As Y is non deterministic, this minimum expected loss will always be greater than zero. Let us call this minimum loss the *noise* or *unavoidable error*, σ_ϵ^2. The noise is then the loss that we expect to have even if we are provided with an optimal algorithm to estimate y.

Let us now consider a sub-optimal algorithm: we want to describe its expected loss in estimating $y(\boldsymbol{x})$. The estimate, $\hat{y}$, produced by this algorithm is in general a function of the dataset with which it has been trained; we call *central estimate*, $\hat{y}_m$, the *best* estimate obtainable by this classifier if we could train it on all possible datasets; here, "best" is relative to a certain measure of optimality: in more rigorous terms, it is the estimate that minimizes the expected loss between itself and all the estimates obtained over the (infinite) set of possible training datasets. We can now define the *bias* of the algorithm: it is the loss between the optimal estimate, $\hat{y}_*$, and its central estimate: $L(\hat{y}_*, \hat{y}_m)$. On the other hand, the *variance* of the algorithm is the quantity $E_{\mathcal{D}}\left[L(\hat{y}_i, \hat{y}_m)\right]$, that is, the average loss between the central estimate and each estimate obtained when training over a particular dataset, $\mathcal{D}$.

It is possible to show that the expected loss of the algorithm when estimating y, $E_{\mathcal{D},Y}\left[L(Y, \hat{y})\right]$ is the sum of the three components just described: bias, variance and noise. For example, the following classical result holds:

$$E\left[(Y - \hat{y})^2\right] = \underbrace{(\hat{y}_m - \hat{y}_*)^2}_{\text{Bias}} + \underbrace{E\left[(\hat{y} - \hat{y}_m)^2\right]}_{\text{Variance}} + \sigma_\epsilon^2 \tag{4.1}$$

Equation (4.1) is the expression of the *bias-variance decomposition* when the loss function is the mean squared error, and is prevalently used in regression problems, that is, when $\hat{y}$ is a scalar or a vector.

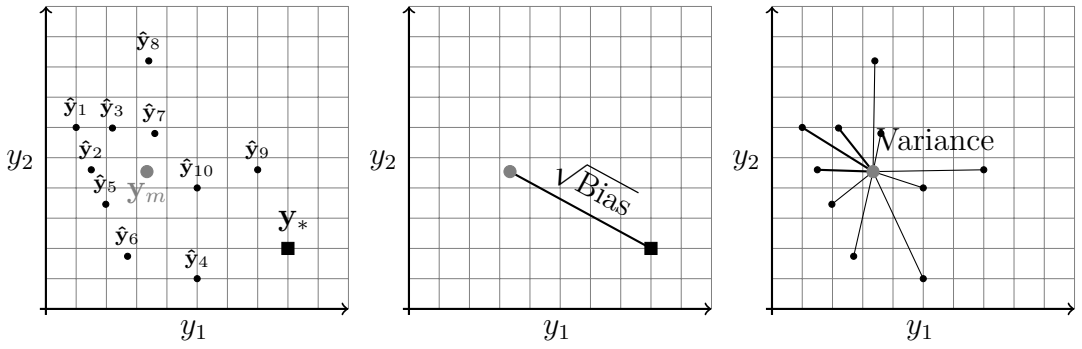


FIGURE 4.1: Graphical demonstration of the bias-variance decomposition.

To better understand eq. (4.1), consider Figure 4.1. It shows the possible results obtained by a regression algorithm, when $y(\boldsymbol{x})$ is a two-dimensional vector. Each black point in the figure is the estimate of the regressor, $\hat{y}(\boldsymbol{x})$, when trained on a certain dataset. As expected, each time we obtain a slightly different estimate. The square mark is the optimal estimate, $\hat{y}_*$. The gray point is the *central estimate* of the regressor, $\hat{y}_m$: it is the mean value of all the per-dataset estimates. The squared distance between $\hat{y}_m$ and $\hat{y}_*$ is the bias of the regressor, and the mean squared distance between $\hat{y}_m$ and each estimate is the variance of the regressor. The sum of these two contributions (and of the noise, not depicted) is equal to the overall loss of estimating y using this imaginary regression algorithm. Of course, in reality, the sum should be taken over the *infinite* set of possible estimates, one for each training dataset.

Equation (4.1) has to be modified when dealing with classification problems, as the expected value of the *zero-one loss*, $L_{0\text{-}1}$, is a more suitable indicator of classification performance than the mean squared error, because in this case $\hat{y}$ is not a numerical quantity but a class label [40]. When using this loss function, no unanimous expression for the bias-variance decomposition exists, but the same concept holds, so that we can write:

$$E\left[L_{0\text{-}1}(Y, \hat{y})\right] = \text{Bias} + \text{Variance} + \text{Irreducible Error} \tag{4.2}$$

As the noise is unavoidable and only depends on the intrinsic relation between the input features and the label, we don't discuss it here: we will focus on the first two terms.

The bias of a predictor is the "systematic" error that it commits, caused by the intrinsic limitations of the hypothesis space that it generates. For example, a linear

boundary classifier will show a strong bias when used to classify populations that are not linearly separable.

How to overcome the bias? By increasing the complexity of the hypothesis space, i.e., by increasing the number of parameters that can be set by the training algorithm. But this does not come without a price: increasing the number of parameters means that the classifier becomes more susceptible of *overfitting*, that is, it will not generalize well, or stated in other words, it will not classify unseen data correctly; moreover, slightly changing the training dataset may cause mayor changes in the predictor, as happens in polynomial interpolation. All these inconveniences will cause an increase in the *variance* term in eq. (4.2).

Classifier ensembles reduce the variance of the classification by averaging. It is known that the sample mean of N i.i.d. observations, each with variance σ^2, has variance $\frac{1}{N}\sigma^2$. The reduction of variance obtained by classifier ensembles is based on the same principle, but we cannot expect the observations (i.e., the label predicted by each weak learner in the ensemble) to be independent between each other. For example, consider the following inequality:

$$\text{err}_{RF} \leq \overline{\rho}\,\frac{1-s^2}{s^2} \tag{4.3}$$

It defines an upper bound for the generalization error rate of a Random Forest [41]. In particular, s is the mean *strength* of the trees in the Forest; broadly speaking, the strength of a classifier measures how well it separates observations with different labels among each other — it is inversely proportional to the bias just defined. The other term in the expression, $\overline{\rho}$, represents the pairwise correlation between the learners in the ensemble. This term is related to the concept of *diversity* between learners in an ensemble. Many definitions of diversity exists; qualitatively, it can be described as the property of two (or more) predictors of committing independent errors; in turn, many measures exist in the literature to quantify it. Several studies try to relate those diversity measures to ensemble accuracy, with mixed results and conclusions [42], [43].

In practice, *an ensemble will have an error rate lower than its constituting weak learners if they have low bias and are diverse between each other.* This means that even if we increase the expected error of the single learners by slightly increasing its variance, we could in effect improve the performance of the ensemble if we do so in order to make the learners more diverse. Of course, this process cannot be

carried on indefinitely: the variance of each classifier eventually reach a critical value, beyond which the ensemble error rate would start to increase.

In our work, we increase the diversity between the predictors in the ensemble by *sample weighting*: it makes them more diverse, at a cost of a slight reduction of their strength. To express the same concept using the terminology of eq. (4.2), we train predictors with slightly more variance and bias, but the overall error is reduced by the "ensemble effect", because we are increasing their diversity.

Let us provide a simple motivation of our approach, by going back to the metaphor with which we opened the chapter. To ensure "diversity" in the answers, it is reasonable to ask each person in the committee to study a different part of the book. An approach similar to the one used by Bagging would be to give each person a random subset of the pages of the book (with some pages repeated, by chance). We propose a slightly more informative approach: we ask each person to study more carefully a particular chapter of the book, chosen at random. The main difference between bagging and our approach is that we ensure *local competence*: each person of the committee knows which chapter has studied, so that he will be more confident on answering questions about that chapter, and he *knows* that he can be of little or no help when the question is about other parts of the book.

The fact that we inject "locality" during the training phase, together with the design choice of using decision trees as weak learners, like Random Forests, explain why we decided to call our classifier ensemble a "Forest of Local Trees", or FLT for short.

The rest of the chapter is organized as follows. In the next section, we will give a quick overview on the main ensemble methods found in the literature: Boosting, Bagging and Random Forests; in Section 4.2 we will describe our algorithm "out of metaphor", in rigorous terms and with mathematical expressions; in order to better understand the complete system, in Section 4.3 we present in detail the training algorithm of the weak learners of the ensemble. Section 4.4 provides experimental results and comparisons with other ensemble classifiers, showing the effectiveness of our approach: in effect, FLTs compares favorably with respect to the other algorithms. We conclude the chapter with an overall discussion of the progress so far and with some proposal for future work.

4.1 Ensemble Methods from the Literature

In this section we review the main contributions in the field of ensemble methods. Of course, it doesn't want to be an exhaustive discussion nor an historical perspective (please refer to [11] or [37] for more information on the topic).

Breiman first proposed his Bagging method in 1996 [44]. The name is a portmanteau for Bootstrap Aggregating. Bagging trains each learner in the ensemble by using different samples of the same underlying population. The classification is carried out by selecting the most frequent label among those proposed by the weak learners. In practice, we are not given with many samples of the population that we want to classify: we only have *one* dataset. Bagging simulates the presence of multiple samples by *bootstrapping* the dataset: each sample is generated by choosing N observations from the dataset with replacement. Bootstrap samples contains on average around the 66% of all the observations in the dataset. Taking different samples makes sure that the learners in the ensemble are diverse. Generally, the weak learners chosen for a Bagging ensemble are decision trees, because they are characterized by very low bias and high variance.

In 2001, Breiman himself proposed a similar ensemble algorithm, the Random Forest [41]. He noticed that more diversity could be injected in a Bagging ensemble, thus reducing its overall variance, by using *random* decision trees instead of standard ones. Random Decision Trees differ from their "deterministic" version in that only a random subset of all the features is considered to determine each split during the training phase (see section 4.3).

Since the definition of Bagging ensembles and Random Forests, a lot of studies have been carried on, both on the theoretical and on the experimental side (see, for example, [45], [46], [47]). Worth mentioning here are two other algorithm based on Bagging, and that use decision trees as weak learners: Random Subspace Ensemble, and Rotation Forests [48], [49].

A different approach to ensemble training is due to Schapire and Freund, who defined the Boosting algorithm [50]. Bagging and Random Forests are "parallel" training procedures, in which the learning phase of each tree is independent from that of the other ones; Boosting proceeds by iteratively improving the overall performance of the ensemble. In so doing, whereas Bagging and Random Forest reduce their error rate by reducing the overall variance, Boosting improves its

performance by reducing also its bias. Let us briefly illustrate this concept by presenting one of the most notable examples of Boosting: AdaBoost [51]. AdaBoost acts as follows. The first weak learner of the ensemble is trained normally, on the whole dataset. Once it has been trained, the algorithm weighs each instance in the dataset, giving to each misclassified observation a weight higher than that assigned to the instances classified correctly. Once the weighting phase has been done, another learner is trained using the *weighted dataset* just generated. This procedure is repeated until the expected number of classifiers is reached or the validation error starts to increase. Samples with higher weight are considered by the training algorithm of the weak learner as more important, and this increases the probability that they are classified correctly in successive rounds of training. This procedure affects the bias of the ensemble because it explicitly corrects the classification errors of each weak learner.

Another important difference between Boosting and Bagging is in the classification method. Whereas Bagging and Random Forest do not weigh the contribution of the weak learners, Boosting gives to each classifier a weight that is proportional to its validation error rate.

Let us point out that many more ensemble methods exist, and we do not discuss them here as their approach to classifier combination differs substantially from the one that we propose [52], [53], [54].

4.2 Forests of Local Trees

Consider a training dataset $\mathcal{D}$, on which we want to train T random decision trees, that will become the weak learners of our FLT. The training algorithm of each decision tree is described in section 4.3.

For reasons that will become evident later, every categorical feature must be transformed to a scalar representation (the most straightforward option being applying one-hot encoding). Let us notice that the decision surface of a tree is not influenced by this transformation, and is also invariant to any linear scaling and translation of the features.

As happens in Boosting, each tree sees the same training dataset $\mathcal{D}$, but with different weights applied to each observation. They are used by the training

algorithm to give more importance to the observations that have an higher weight associated to them (i.e., to increase the probability that they will be classified correctly). The weighted dataset seen by the t-th tree is:

$$\mathcal{D}_t^{(w)} = \{(\boldsymbol{x}_i, y_i; w_{i,t}) : (\boldsymbol{x}_i, y_i) \in \mathcal{D}\} \tag{4.4}$$

We introduce a novel method for calculating the weights of $\mathcal{D}^{(w)}$. We want that each tree becomes more competent on a specific region of the dataset. To do so, we assign to each observation in the dataset a weight that is inversely proportional to the distance between that observation and the so called *centroid* of the tree. As each tree will be assigned to a different centroid, we are building a forest of *local trees*: each of them will become more competent on the region of the dataset that is near to its centroid.

The algorithm for calculating the weights can thus be split into two parts:

1. pick the centroid from $\mathcal{D}$, considering the centroids already picked;
2. calculate the weight of each instance using the centroid just picked.

Let us discuss each step separately. In the next subsection we review the core of our method, the sample weighting strategy. The centroid picking policy is discussed in section 3.3.4.

4.2.1 Sample Weighting Strategy

Consider the usual training dataset $\mathcal{D}$ together with a centroid element $\boldsymbol{x}_t^c$ that we suppose already picked (see section 4.2.2 for details). We define the following *weighting strategy* for calculating the weight associated to the i-th observation in $\mathcal{D}$:

$$w_{i,t} = \exp\left\{-\frac{1}{2}\left[(\boldsymbol{x}_i - \boldsymbol{x}_t^c)\boldsymbol{V}(\boldsymbol{x}_i - \boldsymbol{x}_t^c)^T\right]^{p/2}\right\} \quad \forall\,(\boldsymbol{x}_i, y_i) \in \mathcal{D} \tag{4.5}$$

Where $\boldsymbol{V}$ is the *scaling matrix* and p is the *power* of the weighting strategy. When $p/2 = 1$, eq. (4.5) is clearly proportional to a multivariate normal distribution with mean $\boldsymbol{x}_t^c$ and covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{V}^{-1}$. In practice, it is often sufficient to use the following expression for the weighting strategy:

$$w_{i,t} = \exp\left\{-\frac{1}{2}\phi\left\|\boldsymbol{x}_i - \boldsymbol{x}_t^c\right\|^p\right\} \quad \forall\,(\boldsymbol{x}_i, y_i) \in \mathcal{D} \tag{4.6}$$

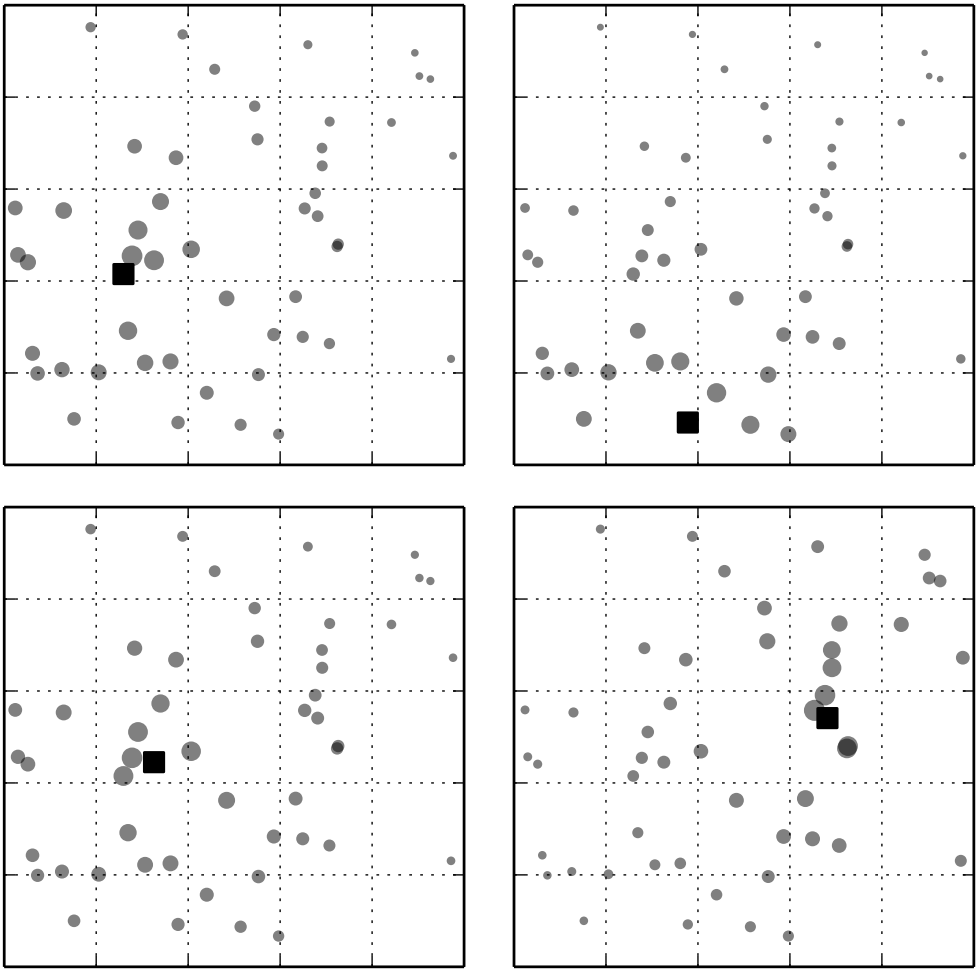


FIGURE 4.2: Examples of weighting distributions.

Where ϕ is the *precision* and p is again the *power* of the weighting strategy. It is easy to recognize that Equation (4.6) degenerates to a multivariate normal distribution with covariance matrix $\boldsymbol{\Sigma} = \phi^{-1}\boldsymbol{I}$ when $p = 2$. Let us highlight that when using eq. (4.6) instead of 4.5, we are just setting $\boldsymbol{V} = \phi\boldsymbol{I}$.

Figure 4.2 shows how the algorithm assigns the weights to the observations in the dataset. Each gray circle represents an instance, and its radius is proportional to the weight assigned to it. The black box is the centroid associated to a particular tree. Clearly, as each trees will focus its training on the instances with higher weight, it will become more competent on a specific region of the dataset. As a consequence, our approach is capable of increasing the diversity between the classifiers in the ensemble.

Of course, choosing the right centroids is fundamental, as it should aim at picking them with the largest possible "spread", in order to minimize the probability that

a region of the dataset remains uncovered.

4.2.2 Picking the Right Centroids

It is important to pick the centroids so that the forest would "span" as widely as possible on the whole dataset. Finding the best centroids can thus be stated as an optimization problem. But due to the complexity of that approach, in this thesis we present an alternative heuristic solution based on picking the centroids on a semi-random way. Other strategies will be experimented in the future, in particular the use of techniques like those found in the *mixtures of experts* [55], [56].

In order to select centroids that are reasonably distant between each other, we propose the following algorithm. It chooses T centroids by picking them randomly one at a time among the observations in the dataset $\mathcal{D}$. Initially, the probability of picking a certain instance $\boldsymbol{x}_i \in \mathcal{D}$ is set to be uniform. Each time that a centroid is picked, the algorithm updates the probabilities of choosing the remaining observations in the dataset, making the instances further away from the centroid just picked more likely to be chosen.

Initially, the *picking probability* is set to be a discrete uniform distribution over the observations in the dataset:

$$P\left(\boldsymbol{x}_1^c = \boldsymbol{x}_i\right) = P_1^c(\boldsymbol{x}_i) = \frac{1}{N} \quad \forall\left(\boldsymbol{x}_i, y_i\right) \in \mathcal{D} \tag{4.7}$$

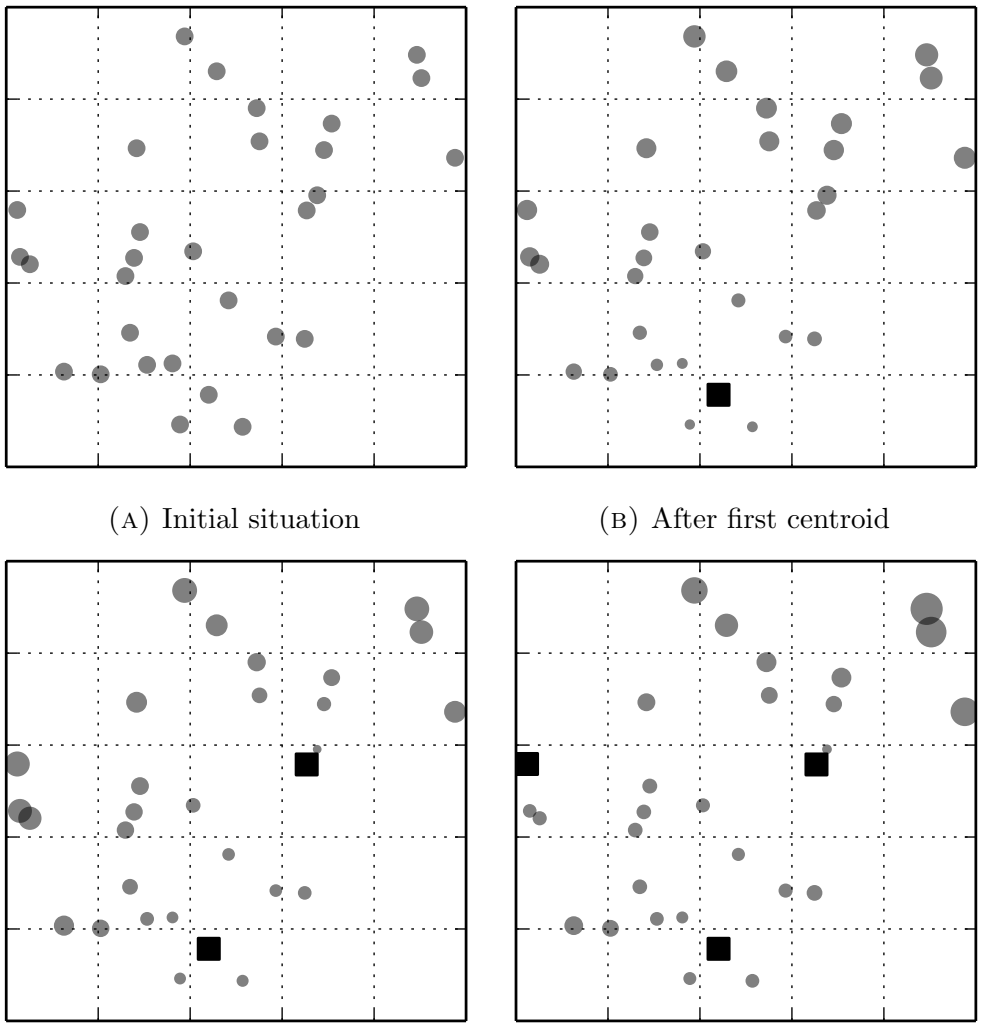
Equation (4.7) states the following: the probability that $\boldsymbol{x}_i$ is chosen as the first centroid, $\boldsymbol{x}_1^c$, is the same for all the instances in $\mathcal{D}$ and equals $\frac{1}{N}$.

After each pick, the picking probability of each observation in $\mathcal{D}$ is updated using the following rule:

$$P_t^c(\boldsymbol{x}_i) \propto P_{t-1}^c(\boldsymbol{x}_i) \cdot \log\left(1 + \left\|\boldsymbol{x}_i - \boldsymbol{x}_{t-1}^c\right\|\right) \quad t = 2, \ldots, T \tag{4.8}$$

Of course, after eq. (4.8) has been applied, the probabilities have to be normalized so that $\sum_{i=1}^{N} P_t^c(\boldsymbol{x}_i) = 1$.

Equation (4.8) does the following: it sets the picking probability of the observation just chosen as centroid to zero, as $\log 1 = 0$; then, as the logarithm is a monotonically increasing function, eq. (4.8) ensures that instances further away from the centroids



(A) Initial situation (B) After first centroid

(C) After second centroid (D) After third centroid

FIGURE 4.3: How the picking probabilities are updated after choosing the centroids.

already picked have an higher probability of getting chosen in the successive rounds. The algorithm terminates when all T centroids have been picked.

The algorithm is explained graphically in Figure 4.3. The gray circles are the observations in the dataset, and their radius is proportional to the picking probability. The centroids are represented as black boxes. Initially (Fig. 4.3a) every instance is equally likely to be picked as centroid. After the first centroid has been picked (Fig. 4.3b), the instances that are near to the centroid become less likely to be picked than those further away. The same happens after the second (Fig. 4.3c) and third (Fig. 4.3d) centroids have been picked. The figures highlight that each update does not cancel the effect of the previous ones.

4.2.3 Making the Classification

Once every tree in the forest has been trained, the FLT can be finally used for classification.

Let us consider an instance $\boldsymbol{x}$ to classify. Each tree in the Forest is fed with $\boldsymbol{x}$ and produces its guess for the label, $\hat{y}_t(\boldsymbol{x})$, $t = 1, \ldots, T$. When all the trees have made their prediction, the Forest is fed with this *prediction sample*, $\hat{\underline{y}}(\boldsymbol{x})$. The Forest labels the instance using the mode of the prediction sample:

$$\hat{y}_F(\boldsymbol{x}) = Mo\left[\hat{\underline{y}}(\boldsymbol{x})\right] \tag{4.9}$$

This classification rule for an FLT is thus the same used by a Random Forest and by Bagging.

It is also possible to use a different rule, which considers the probability estimates for the class labels produced by the single trees and whose output is in turn a probability estimate instead of a direct class label. In other works, the concept of probability estimate is called *confidence* of a classifier on a class label. Now, the output of each tree is supposed to be a vector $\hat{\boldsymbol{y}}_t(\boldsymbol{x})$ of length $m = |\mathcal{Y}|$, whose k-th component is the confidence of the t-th tree that the label associated with $\boldsymbol{x}$ is $y_k \in \mathcal{Y}$.

The Forest is now fed with the sample of probability estimates (that replaces the prediction sample), $\hat{\underline{\boldsymbol{y}}}(\boldsymbol{x})$; to produce the final probability vector, the FLT calculates its mean value:

$$\hat{\boldsymbol{y}}_F(\boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T}\hat{\boldsymbol{y}}_t(\boldsymbol{x}) \tag{4.10}$$

Once $\hat{\boldsymbol{y}}_F(\boldsymbol{x})$ has been calculated, the standard MAP rule can be applied to assign the label to $\boldsymbol{x}$.

4.3 Growing Decision Trees

A binary decision tree T can be defined a set of *nodes*, of which one is marked as *root*. The classification always starts from this root node, N_0. The other nodes are then eventually called recursively, to obtain the final prediction of the tree, using

the following expression:

$$N_c(\boldsymbol{x}) = \begin{cases} y_c & \text{if } N_c \text{ is a leaf} \\ N_l(\boldsymbol{x}) & \text{if } x_c < t_c \\ N_r(\boldsymbol{x}) & \text{if } x_c \geq t_c \end{cases} \tag{4.11}$$

Equation (4.11) states that when the instance to be classified reaches the node N_c, three things can happen: if N_c is a *leaf*, then the output of the node is the label associated with that leaf, and the classification ends; if N_c is not a leaf, then the following *splitting criterion* is applied: a particular feature of $\boldsymbol{x}$, x_c, is compared to a *threshold value*, if x_c is lower than the threshold, the decision is passed to the *left child node* (or left sub-tree) of the current node; if x_c is greater than or equal to the threshold, the decision is passed to the *right child node* (or right sub-tree) of the current node. Of course, which feature to check, and the threshold value, both depend on the current node.

There exists many algorithm for training (or *growing*) a decision tree [57], [11], [58], [59], [60], [61]. Let us briefly recall here the learning algorithm that grows a so called *Random Decision Tree*, that is the weak learner used by FLT and by Random Forest. We show first the version that does not make use of the weights contained in $\mathcal{D}^{(w)}$, then in section 4.3.1 we describe how to change the algorithm to use them.

The algorithm proceeds as follows, starting from the root node and proceeding until each path that starts from it ends on a leaf. A dataset $\mathcal{D}_c$ is associated to the current node, N_c. If N_c is the root, then $\mathcal{D}_c$ is the whole training dataset, $\mathcal{D}$ (or a bootstrap sample of the whole dataset, when it is used inside a Random Forest). The *purity* of $\mathcal{D}_c$ is estimated through the following measure, called the *Gini impurity index*:

$$g(\mathcal{D}_c) = 1 - \sum_{y \in \mathcal{Y}} \frac{|\{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_c : y_i = y\}|}{|\mathcal{D}_c|} = 1 - \sum_{y \in \mathcal{Y}} P(y) \tag{4.12}$$

The Gini index is zero if all the instances in $\mathcal{D}_c$ belong to the same class. This is the first and most important *stopping criterion* for the training algorithm: if $\mathcal{D}_c$ contains instances coming from only one population, say y_c, N_c is set to be a leaf and the label associated to the node is, of course, y_c. More stopping criteria could be defined, for example the following one, based on the size of $\mathcal{D}_c$: if $|\mathcal{D}_c| \leq N_{th}$,

the node is marked as a leaf and the label associated to it is the most frequent in $\mathcal{D}_c$.

If no stopping criterion is met, the algorithm has to set the feature, x_c, and the threshold, t_c, used by N_c as splitting criterion. x_c and t_c are chosen as follows. A set of *candidate features* are drawn at random from the feature space. The number of candidate features, n_c, is one of the parameters of the training algorithm: typically, for Random Forests, n_c is set to $\sqrt{n}$ or $\log n$, where n is the total number of features. This random choice of feature is the reason for calling the this kind of trees *Random Decision Trees*. When $n_c = n$, the algorithm presented here degenerates to a standard *CART Classification Tree* [59].

For each candidate feature, x_j, the optimal threshold is defined as the one that maximizes the *Impurity Variation*:

$$\Delta I(\mathcal{D}_c, x_j, t) = g(\mathcal{D}_c) - \frac{1}{|\mathcal{D}_c|}\Big(|\mathcal{D}_l|\, g(\mathcal{D}_l) + |\mathcal{D}_r|\, g(\mathcal{D}_r)\Big) \tag{4.13}$$

Where $\mathcal{D}_l$ and $\mathcal{D}_r$ are the datasets "generated" by the splitting criterion: $\mathcal{D}_l$ contains the observations in $\mathcal{D}_c$ that have $x_j < t$, and $\mathcal{D}_r$ contains the ones that have $x_j \geq t$.

The feature-threshold pair that maximizes the Impurity Variation among the candidate features is selected as splitting criterion for the current node, and the training process proceeds recursively on N_l and N_r, that are fed with the datasets $\mathcal{D}_l$ and $\mathcal{D}_r$, respectively.

The Impurity Variation determines another stopping criterion: if it is zero for all the candidate features, once again the current node becomes a leaf, and the label associated to it is the most frequent among those found in $\mathcal{D}_c$.

Let us highlight a small implementation note. Letting the tree output the class probability estimates instead of the labels themselves is a very simple task: it suffices to substitute y_c with the label probability estimated using the data in $\mathcal{D}_c$.

4.3.1 Using a Weighted Dataset for Training

The overall training algorithm just presented needs only a pair of minor modifications in order to handle a weighted dataset $\mathcal{D}^{(w)}$ instead of $\mathcal{D}$. The first change is

in eq. (4.12), in which the sample probability estimate, $P(y)$, has to be replaced by its "weighted" counterpart:

$$P(y) \text{ replaced by } P_w(y) = \frac{1}{\sum_{i=1}^{|\mathcal{D}^{(w)}|} w_i} \sum_{i=1}^{|\mathcal{D}^{(w)}|} I(y_i = y)\, w_i \qquad y \in \mathcal{Y} \tag{4.14}$$

Where I is the indicator function, which is 1 only when the argument is true. It is easy to verify that $P_w(y) = P(y)$ when we use an unweighted dataset, as in this case $w_i = 1$, $\forall i$.

The second modification occurs in eq. (4.13), where the cardinality of the datasets is replaced by the sum of the weights of their elements:

$$|\mathcal{D}| \text{ replaced by } \sum_{i=1}^{|\mathcal{D}^{(w)}|} w_i \tag{4.15}$$

4.4 Experiments

We ran the experiments on the full set of UCI datasets presented in the introduction. We show the effectiveness of our approach by comparing the performance of FLT with those of the three other ensemble classifiers presented in section 4.1.

4.4.1 Experimental Setup

We tested our algorithm with a wide range of parameter values. In order to compare the results between datasets, we normalized the features so that they all range in $[0, 1]$.

We trained each ensemble with 10 weak learners. The remaining parameters of the competing methods have been left to the standard settings of the software we used to carry out the experiments, *scikit-learn* [32].

To train our FLTs, we used the parameters and the values shown in Table 4.1. Each value is combined with the remaining ones, forming all possible combinations, for a total of 18 FLTs. Let us now discuss each parameter in detail.

The number of candidate features, n_c, has been already discussed in section 4.3: in the table, it is shown as percent of the total number of features of each dataset.

Parameter name	Symbol	Values
Percent of Candidate Features	n_c	10%, 30%, 50%
Sample Percent	ζ	50%, 100%
Maximum Number of Leaves	r_{leaf}	50, 100, ∞

TABLE 4.1: Parameters used in the experiments.

The *maximum number of leaves*, r_{leaf}, is a little tweak that we applied to the algorithm presented in section 4.3. As its name suggests, it indicates the maximum number of leaves that each tree can contain. In order to make sure that a particular path in the tree does not exhaust the available leaves, the tree must be grown using a breadth first rule, that is, each "level" of the tree has to be completed before going deeper (see documentation of [32]).

The *sample percent*, ζ, indicates how many observations can be used to train each tree. We opted to use this parameter to see if some datasets may benefit from the diversity gained when each tree is trained on a different subset of the whole dataset. We experimented with just two values for ζ: 100%, that indicates that no observation is removed, and 50%, that indicates that only one half of the observations are chosen for each tree.

In all the configurations, the precision and the power of the weighting strategy (see section 4.2.1) are set to 1. Further studies on these two parameters will be carried on in the future; in preliminary experiments, we noticed that no significant performance increase (averaged over all the datasets) is expected when choosing a different value for the precision or the power. However, on selected datasets (e.g., on *balance-scale*), increasing the precision leads to an overwhelming improvement of the accuracy.

4.4.2 Experimental Results

We repeated 10 times a 10-fold cross validation on every dataset, building a sample of 100 accuracies for each FLT and for each competing ensemble algorithm. These samples have been compared using the standard t-test to identify significant difference in the mean accuracy. The significance level has been set to 5 percent.

Table 4.2 shows a synthesis of the results. Each win/tie/loss triplets is relative to the classifier at the top of the column, and indicates on how many datasets

FLT parameters					Random
n_c	ζ	r_{leaf}	AdaBoost	Bagging	Forest
10%	50%	50	5/11/20	15/14/7	13/19/4
		100	4/ 9/23	15/14/7	13/20/3
		∞	3/ 9/24	14/17/5	11/22/3
	100%	50	5/ 5/26	10/15/11	9/17/10
		100	4/ 5/27	9/17/10	8/21/7
		∞	3/ 6/27	10/19/7	6/26/4
30%	50%	50	3/ 4/29	7/21/8	3/26/7
		100	2/ 5/29	6/22/8	2/29/5
		∞	0/ 4/32	4/25/7	2/27/7
	100%	50	2/ 3/31	3/22/11	2/19/15
		100	1/ 4/31	2/22/12	3/19/14
		∞	0/ 4/32	1/24/11	2/20/14
50%	50%	50	1/ 5/30	5/26/5	4/22/10
		100	1/ 4/31	4/29/3	3/26/7
		∞	0/ 3/33	1/29/6	3/28/5
	100%	50	1/ 2/33	3/25/8	7/16/13
		100	1/ 3/32	1/25/10	8/17/11
		∞	0/ 3/33	0/25/11	6/16/14

TABLE 4.2: Overview of the results: win/tie/loss triplets.

that classifier has had an higher/same/lower accuracy than the Forest of Local Trees whose parameters are shown at the beginning of the row. The results show that the Forests of Local Trees are significantly better than AdaBoost on more than 30 datasets, and outperform both Bagging ensembles and Random Forests on 12 to 15 datasets, while having the same performance on almost every other domain. The table highlights that on the vast majority of datasets, $\zeta = 100\%$ is the better option, but in Table 4.3 it is possible to see that, on some datasets (see in particular *balance-scale*), reducing the number of training samples improves the accuracy of the FLT.

Detailed results for the three best configurations are shown in Table 4.3, 4.4 and 4.5; the empty circle indicates that the FLT did significantly worse than the classifier shown at the top of the column, while a full circle indicates a significantly better result. At the bottom of the table are shown the win/tie/loss triplets. Let us notice that limiting the maximum number of leaves slightly increases the performance

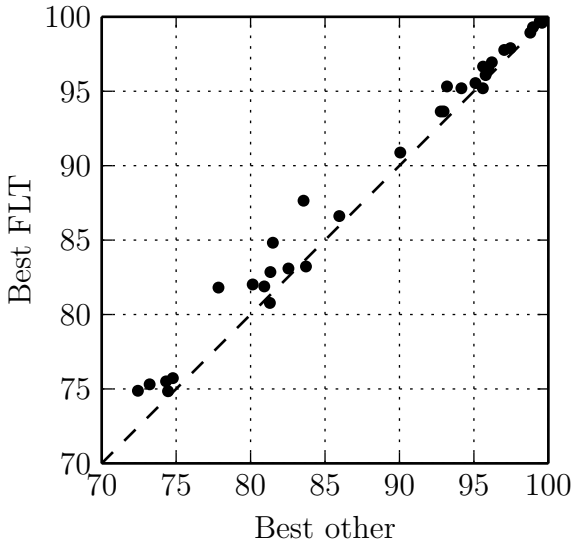


FIGURE 4.4: Overall accuracy comparison.

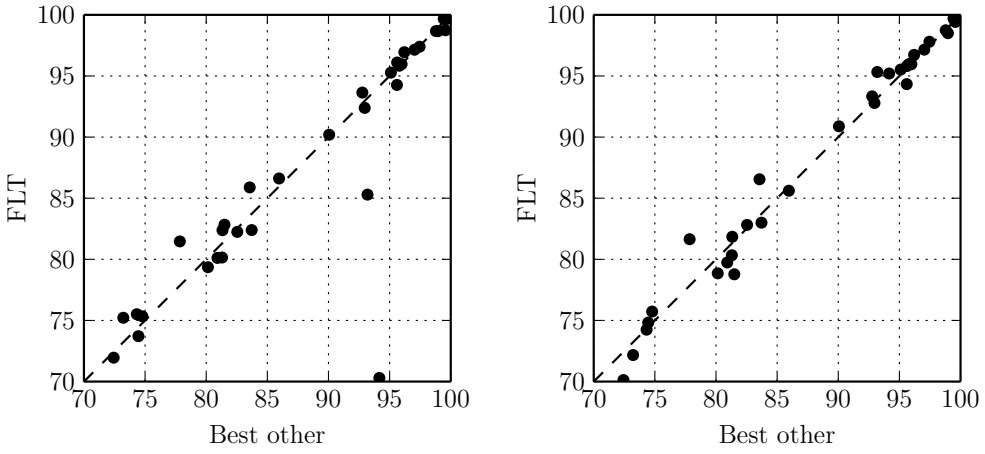
on some datasets, but at the cost of a strong accuracy reduction on others, in particular on *letter* and *vowel*.

Figure 4.4 gives a graphical overview of the results summarized in Table 4.2. On the y-axis is the accuracy of the best FLT, and on the x-axis is the accuracy of the best competing ensemble. If a point lies above the dashed line, the FLT has been better than the competing ensembles. As this is the case for almost every point in the scatter plot, we can conclude that our Forests of Local Trees, with proper tuning, can be used with profit on a wide range of domains.

Figure 4.5 shows how single FLTs compare to the competing methods. In Fig. 4.5a you can clearly see the performance loss for *letter* and *vowel*. To increase the accuracy on these two datasets it is sufficient to remove the leaf limit, as shown in 4.5b; however, this comes at the cost of a slight performance decrease on some other datasets.

4.5 Conclusions

We developed and tested a novel training algorithm for ensemble classifiers, which demonstrated its effectiveness on a wide range of standard classification domains. In particular, we showed that very good results can be achieved when using the following settings: $n_c = 30\%$, $r_{\text{leaf}} = \infty$, and $\zeta = 100\%$; in effect, the accuracy of



(A) Comparison for $r_{\text{leaf}} = 50$ (B) Comparison for $r_{\text{leaf}} = \infty$

FIGURE 4.5: Accuracy comparison for the two best configurations. In both cases $n_c = 30\%$, $\zeta = 100\%$.

our Forests of Local Trees is almost always higher than that obtained by AdaBoost, and at least as high as the one obtained by Bagging and Random Forests.

However, our approach deserves further study, that will be carried on as future work. In particular, we would like to highlight the following noteworthy points:

- Breiman motivates the approach of Bagging (and thus the one used in Random Forests), by noting that a bootstrap sample taken from the dataset reproduces the same probability distribution of the original populations that we want to classify. Our algorithm works by *modifying* the probability distribution seen by each classifier. Something similar happens in Boosting and to some extent also in the mixture of experts model.

  It is then very important to investigate why and when modifying the population distribution seen by the weak learners improves the overall performance of the ensemble.

- Similarly, we would like to discover if it is possible to define an *optimal* sample weighting strategy, and if it is possible to compute it efficiently.

- Our approach has some similarities with the mixture of experts model. In effect, our Forest of Local Trees computes the weights to be used only in the training phase. The mixture of experts model suggests to weight the prediction coming from each weak learner depending on its confidence on the instance to be classified. Moreover, the approach used for weight assignment

is based on a deterministic optimization algorithm that tries to minimize the error rate by calculating the weights and the parameters of each weak learner at the same time. Our procedure is different as it determines the weights in a semi-random way, and the weak learners can be trained only after the weights have been computed. We would like to investigate if the two approaches can be merged and seen from an unifying viewpoint.

- As each weak learner is trained to become a *local expert*, it is straightforward to apply dynamic ensemble selection techniques to our Forests of Local Trees. The term dynamic, in this context, refers to the fact that the selection is made instance-wise: each time a new feature vector $\boldsymbol{x}$ has to be classified, a different subset of the learners in the forest is activated.

  Early experimentation in this direction showed that we are able to reduce the ensemble size to one tenth, with minimal accuracy loss. In our view, the performance of the ensemble could even be improved by reducing the number of active learners in the following way: by selecting only the classifiers that are most competent on the region of the feature space in which the instance to be classified resides.

- Centroid selection has to be further studied. For example, we can use our Multi-Resolution Index (Chapter 3) to identify the best region of the dataset in which to put each centroid. Another possibility is to use an optimization based technique like that used in mixtures of experts.

- The term "forest" might be dropped if other types of weak learners can be used in place of Decision Trees. For example, it may be effective to use Neural Networks or Naive Bayes classifiers.

- We expect the results to change if we increase T, the number of weak learners in the ensemble: the difference in performance among the classification methods should fade out. However, preliminary experiments on biological data (to be published soon) showed that, on some domains, our Forests of Local Trees can obtain a far higher performance than Random Forests and Bagging, even with $T \gg 10$. We would like to investigate if this result is generalizable to a wider range of classification problems.

Dataset	FLT	AdaBoost	Bagging	Random Forest
anneal-orig	99.68±0.6	98.84±1.1 •	99.47±0.7 •	99.45±0.8 •
anneal	99.68±0.6	98.83±1.1 •	99.29±0.9 •	99.42±0.8 •
audiology	82.39±10.7	77.01±12.0•	83.72±10.5	79.99±12.5
autos	85.88±7.4	74.21±10.8•	83.56±8.2 •	80.99±8.8 •
balance-scale	82.83±3.1	77.67±4.7 •	79.79±3.9 •	81.50±3.5 •
breast-cancer	71.96±7.2	66.96±8.4 •	70.81±6.7	72.43±5.9
breast-w	95.97±2.3	93.65±2.4 •	95.35±2.6	95.98±2.3
colic-orig	97.15±2.7	95.33±4.0 •	97.04±2.6	94.45±3.7 •
colic	96.94±2.7	94.22±3.7 •	96.20±2.8	95.83±3.4 •
credit-a	86.61±3.9	81.07±4.4 •	85.79±4.1	85.96±4.1
credit-g	75.22±3.5	67.79±4.7 •	72.28±4.5 •	73.22±3.5 •
diabetes	75.51±4.3	68.84±5.3 •	74.32±4.4	73.93±4.6 •
glass	75.34±9.1	65.16±9.3 •	73.96±7.7	74.78±9.3
heart-c	80.11±7.6	73.58±6.4 •	79.68±7.6	80.92±7.5
heart-h	79.36±7.0	74.57±7.6 •	78.29±6.4	80.14±6.6
heart-statlog	80.15±7.8	73.22±7.7 •	79.81±6.8	81.30±6.6
hepatitis	90.19±6.4	86.58±8.5 •	89.68±6.7	90.05±7.2
hypothyroid	99.67±0.3	99.58±0.3	99.68±0.3	99.27±0.5 •
ionosphere	92.39±4.6	87.67±6.4 •	90.94±5.1 •	92.96±4.6
iris	94.27±6.0	94.47±6.1	95.60±5.6	95.13±6.2
kr-vs-kp	98.74±0.6	99.18±0.6 ○	99.57±0.3 ○	98.58±0.6
labor	98.67±4.5	97.47±6.5	98.97±4.1	98.63±4.6
letter	70.30±1.2	85.32±0.9 ○	93.10±0.7 ○	94.16±0.6 ○
lymph	82.25±9.9	76.63±10.4•	81.61±10.0	82.54±10.0
primary-tumor	46.27±7.4	35.53±6.5 •	40.25±7.6 •	42.45±7.6 •
rootstock	59.67±24.3	46.83±20.2•	56.67±19.0	54.17±22.0
segment	97.39±1.1	95.47±1.4 •	97.19±1.1	97.45±1.2
sick	98.67±0.5	98.48±0.6 •	98.79±0.5	98.35±0.6 •
sonar	81.47±8.4	70.29±8.4 •	77.35±8.9 •	77.84±8.2 •
soybean	93.64±2.8	90.19±3.7 •	92.77±3.0 •	92.76±2.5 •
splice	95.27±1.0	94.24±1.3 •	95.10±1.1	85.81±2.2 •
vehicle	73.71±3.7	69.16±4.7 •	74.46±3.6	74.32±3.6
vote	95.83±3.0	93.09±3.8 •	94.96±3.2 •	95.79±2.9
vowel	85.29±3.9	76.32±4.9 •	91.01±2.9 ○	93.19±2.5 ○
waveform	82.38±1.7	73.71±1.7 •	80.89±1.7 •	81.33±1.8 •
zoo	96.10±6.2	93.07±7.4 •	95.62±6.0	95.31±6.1
(Win/Tie/Loss)		(2/3/31)	(3/22/11)	(2/19/15)

TABLE 4.3: Comparison for FLT with $n_c = 30\%$, $r_{\text{leaf}} = 50$, and $\zeta = 100\%$.

Dataset	FLT	AdaBoost	Bagging	Random Forest
anneal-orig	99.66±0.6	98.84±1.1 •	99.47±0.7	99.45±0.8 •
anneal	99.68±0.6	98.83±1.1 •	99.29±0.9 •	99.42±0.8 •
audiology	82.60±11.1	77.01±12.0•	83.72±10.5	79.99±12.5
autos	85.88±7.4	74.21±10.8•	83.56±8.2 •	80.99±8.8 •
balance-scale	82.40±3.3	77.67±4.7 •	79.79±3.9 •	81.50±3.5
breast-cancer	70.08±7.8	66.96±8.4 •	70.81±6.7	72.43±5.9 ○
breast-w	95.99±2.3	93.65±2.4 •	95.35±2.6	95.98±2.3
colic-orig	97.15±2.7	95.33±4.0 •	97.04±2.6	94.45±3.7 •
colic	96.94±2.7	94.22±3.7 •	96.20±2.8	95.83±3.4 •
credit-a	85.88±3.9	81.07±4.4 •	85.79±4.1	85.96±4.1
credit-g	74.89±3.5	67.79±4.7 •	72.28±4.5 •	73.22±3.5 •
diabetes	74.91±4.6	68.84±5.3 •	74.32±4.4	73.93±4.6
glass	75.48±8.8	65.16±9.3 •	73.96±7.7	74.78±9.3
heart-c	79.90±6.8	73.58±6.4 •	79.68±7.6	80.92±7.5
heart-h	79.12±7.4	74.57±7.6 •	78.29±6.4	80.14±6.6
heart-statlog	80.41±7.4	73.22±7.7 •	79.81±6.8	81.30±6.6
hepatitis	90.19±6.4	86.58±8.5 •	89.68±6.7	90.05±7.2
hypothyroid	99.65±0.3	99.58±0.3	99.68±0.3	99.27±0.5 •
ionosphere	92.39±4.6	87.67±6.4 •	90.94±5.1 •	92.96±4.6
iris	94.27±6.0	94.47±6.1	95.60±5.6	95.13±6.2
kr-vs-kp	99.30±0.4	99.18±0.6	99.57±0.3 ○	98.58±0.6 •
labor	98.67±4.5	97.47±6.5	98.97±4.1	98.63±4.6
letter	76.99±1.1	85.32±0.9 ○	93.10±0.7 ○	94.16±0.6 ○
lymph	82.25±9.9	76.63±10.4•	81.61±10.0	82.54±10.0
primary-tumor	44.51±7.3	35.53±6.5 •	40.25±7.6 •	42.45±7.6
rootstock	59.67±24.3	46.83±20.2•	56.67±19.0	54.17±22.0
segment	97.78±1.0	95.47±1.4 •	97.19±1.1 •	97.45±1.2 •
sick	98.79±0.5	98.48±0.6 •	98.79±0.5	98.35±0.6 •
sonar	81.47±8.4	70.29±8.4 •	77.35±8.9 •	77.84±8.2 •
soybean	93.54±2.8	90.19±3.7 •	92.77±3.0	92.76±2.5 •
splice	95.44±1.1	94.24±1.3 •	95.10±1.1 •	85.81±2.2 •
vehicle	74.85±3.7	69.16±4.7 •	74.46±3.6	74.32±3.6
vote	95.83±3.0	93.09±3.8 •	94.96±3.2 •	95.79±2.9
vowel	92.09±2.7	76.32±4.9 •	91.01±2.9 •	93.19±2.5 ○
waveform	82.85±1.7	73.71±1.7 •	80.89±1.7 •	81.33±1.8 •
zoo	96.10±6.2	93.07±7.4 •	95.62±6.0	95.31±6.1
(Win/Tie/Loss)		(1/4/31)	(2/22/12)	(3/19/14)

TABLE 4.4: Comparison for FLT with $n_c = 30\%$, $r_{\text{leaf}} = 100$, and $\zeta = 100\%$.

Dataset	FLT	AdaBoost	Bagging	Random Forest
anneal-orig	99.68±0.7	98.84±1.1 •	99.47±0.7 •	99.45±0.8 •
anneal	99.70±0.5	98.83±1.1 •	99.29±0.9 •	99.42±0.8 •
audiology	83.00±10.8	77.01±12.0•	83.72±10.5	79.99±12.5
autos	86.55±6.6	74.21±10.8•	83.56±8.2 •	80.99±8.8 •
balance-scale	78.78±4.0	77.67±4.7	79.79±3.9	81.50±3.5 ○
breast-cancer	70.12±7.3	66.96±8.4 •	70.81±6.7	72.43±5.9 ○
breast-w	95.96±2.4	93.65±2.4 •	95.35±2.6	95.98±2.3
colic-orig	97.15±2.7	95.33±4.0 •	97.04±2.6	94.45±3.7 •
colic	96.73±2.8	94.22±3.7 •	96.20±2.8	95.83±3.4 •
credit-a	85.61±3.8	81.07±4.4 •	85.79±4.1	85.96±4.1
credit-g	72.17±4.1	67.79±4.7 •	72.28±4.5	73.22±3.5
diabetes	74.24±4.3	68.84±5.3 •	74.32±4.4	73.93±4.6
glass	75.73±8.3	65.16±9.3 •	73.96±7.7	74.78±9.3
heart-c	79.74±6.8	73.58±6.4 •	79.68±7.6	80.92±7.5
heart-h	78.85±6.9	74.57±7.6 •	78.29±6.4	80.14±6.6
heart-statlog	80.33±7.1	73.22±7.7 •	79.81±6.8	81.30±6.6
hepatitis	90.88±6.7	86.58±8.5 •	89.68±6.7	90.05±7.2
hypothyroid	99.66±0.3	99.58±0.3	99.68±0.3	99.27±0.5 •
ionosphere	92.79±4.0	87.67±6.4 •	90.94±5.1 •	92.96±4.6
iris	94.33±5.8	94.47±6.1	95.60±5.6	95.13±6.2
kr-vs-kp	99.42±0.4	99.18±0.6 •	99.57±0.3 ○	98.58±0.6 •
labor	98.50±4.8	97.47±6.5	98.97±4.1	98.63±4.6
letter	95.20±0.5	85.32±0.9 •	93.10±0.7 •	94.16±0.6 •
lymph	82.82±9.2	76.63±10.4•	81.61±10.0	82.54±10.0
primary-tumor	42.05±7.0	35.53±6.5 •	40.25±7.6	42.45±7.6
rootstock	58.83±21.3	46.83±20.2•	56.67±19.0	54.17±22.0
segment	97.80±0.9	95.47±1.4 •	97.19±1.1 •	97.45±1.2 •
sick	98.74±0.5	98.48±0.6 •	98.79±0.5	98.35±0.6 •
sonar	81.65±7.9	70.29±8.4 •	77.35±8.9 •	77.84±8.2 •
soybean	93.33±2.8	90.19±3.7 •	92.77±3.0	92.76±2.5
splice	95.52±1.1	94.24±1.3 •	95.10±1.1 •	85.81±2.2 •
vehicle	74.82±3.6	69.16±4.7 •	74.46±3.6	74.32±3.6
vote	95.97±2.7	93.09±3.8 •	94.96±3.2 •	95.79±2.9
vowel	95.32±2.5	76.32±4.9 •	91.01±2.9 •	93.19±2.5 •
waveform	81.84±1.7	73.71±1.7 •	80.89±1.7 •	81.33±1.8 •
zoo	95.81±5.6	93.07±7.4 •	95.62±6.0	95.31±6.1
(Win/Tie/Loss)		(0/4/32)	(1/24/11)	(2/20/14)

TABLE 4.5: Comparison for FLT with $n_c = 30\%$, $r_{\text{leaf}} = \infty$, and $\zeta = 100\%$.

Bibliography

[1] K. Bache and M. Lichman. Uci machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

[2] D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers, 1999.

[3] S. Kotsiantis and Pintelas P. Kanellopoulos, D. Data preprocessing for supervised learning. *International Journal of Computer Science*, 1(2):111–117, 2006.

[4] H. J. Hu, Y. Pan, R. Harrison, and Tai P. C. Improved protein secondary structure prediction using support vector machine with a new encoding scheme and an advanced tertiary classifier. *IEEE Transaction on NanoBioscience*, 3 (4):265–271, 2004.

[5] S.F. da Silva, A.J.M. Traina, M.X. Ribeiro, J. do E.S.Batista Neto, and A.J.M. Traina. Ranking evaluation functions to improve genetic feature selection in content-based image retrieval of mammograms. In *22nd IEEE International Symposium on Computer-Based Medical Systems*, pages 1–8, 2009.

[6] B. Chakraborty. Genetic algorithm with fuzzy fitness function for feature selection. In *Proceedings of the 2002 IEEE International Symposium on Industrial Electronics*, volume 1, pages 315–319, 2002.

[7] Bir Bhanu and Yingqiang Lin. Genetic algorithm based feature selection for target detection in sar images. *Image and Vision Computing*, 21:591–608, 2003.

[8] Alvin C. Rencher. *Methods of Multivariate Analysis*. John Wiley and Sons, 2002.

[9] Ronald A. Fisher. *Statistical Methods for Research Workers*. Oliver and Boyd, 1925.

[10] Daniel Lewandowski, Roger M. Cooke, , and Radboud J. Duintjer Tebbens. Sample-based estimation of correlation ratio with polynomial approximation. *ACM Transactions on Modeling and Computer Simulation*, 5:1–16, 2007.

[11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[12] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Boston Academic Press, 1990.

[13] Saamer Singh. Multiresolution estimates of classification complexity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1534–1539, 2003.

[14] Tin Kam Ho and Mitra Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:289–300, 2002.

[15] Tin Kam Ho and Henry S. Baird. Estimating the intrinsic difficulty of a recognition problem. In *Proceedings of the 12th International Conference on Pattern Recognition*, pages 178–183, 1994.

[16] D. O. Loftsgaardne and C. P. Quesenberry. A non-parametric estimate of multivariate density function. *Annals of Mathematical Statistics*, pages 1049–1051, 1965.

[17] E. Fix. Discriminatory analysis: Nonparametric discrimination: Consistency properties. Technical Report Project 21-49-004, Report Number 4, USAF School of Aviation Medicine, Randolf Field, Texas, 1951.

[18] Emanuel Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[19] Seung-Jean Kim, Alessandro Magnani, and Stephen P. Boyd. Robust fisher discriminant analysis. In *Advances in Neural Information Processing Systems*, pages 659–666. MIT Press, 2006.

[20] Tin Kam Ho and Henry S. Baird. Pattern classification with compact distribution maps. *Computer Vision and Image Understanding*, 70:101–110, 1998.

[21] F. W. Smith. Pattern classifier design by linear programming. *IEEE Transactions on Computers*, 17:367–372, 1968.

[22] S. P. Smith and A. K. Jain. A test to determine the multivariate normality of a data set. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:757–761, 1988.

[23] Aarnoud Hoekstra and Robert P.W. Duin. On the nonlinearity of pattern classifiers. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 271–275, 1996.

[24] A. F. Kohn, L. G. M. Nakano, and M. O. E. Silva. A class discriminability measure based on feature space partitioning. *Pattern Recognition*, 29(5): 873–887, 1996.

[25] Jie Li, Guan Han, Jing Wen, and Xinbo Gao. Robust tensor subspace learning for anomaly detection. *International Journal of Machine Learning and Cybernetics*, 2:89–98, 2011.

[26] Nan Li, Gong-De Guo, Li-Fei Chen, and Si Chen. Optimal subspace classification method for complex data. *International Journal of Machine Learning and Cybernetics*, 2012.

[27] E. Bernado-Mansilla and Tin Kam Ho. Domain of competence of xcs classifier system in complexity measurement space. *Transaction on Evolutionary Computation*, 9(1):82–104, 2005.

[28] Julián Luengo and Francisco Herrera. Shared domains of competence of approximate learning models using measures of separability of classes. *Information Sciences*, 185(1):43–65, 2012.

[29] José Martínez Sotoca, Ramón Alberto Mollineda, and José Salvador Sánchez. A meta-learning framework for pattern classification by means of data complexity measures. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 10(29):31–38, 2006.

[30] Julián Luengo, Alberto Fernández, Salvador García, and Francisco Herrera. Addressing data complexity for imbalanced data sets: analysis of smote-based oversampling and evolutionary undersampling. *Soft Computing*, 15(10): 1909–1936, 2011.

[31] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. In *Proceedings of the National Institute of Sciences of India*, volume 2, pages 49–55, 1936.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

[33] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–=198, 1999.

[34] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 139–148, 2009.

[35] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall, 2012.

[36] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5: 197–227, 1990.

[37] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*. Springer, 2000.

[38] L. G. Valiant. A theory of the learnable, 1984.

[39] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):173–180, 2007.

[40] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, 1985.

[41] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[42] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51:181–207, 2003.

[43] E. K. Tang, Suganthan P. N., and Yao X. An analysis of diversity measures. *Machine Learning*, 65:247–271, 2006.

[44] Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[45] Gérard Biau. Analysis of a random forests model. *Journal of Machine Learning Research*, 13:1063–1095, 2012.

[46] Gérard Biau, Luc Devroye, and Gábor Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9: 2039–2057, 2008.

[47] Giuliano Armano and Nima Hatami. Random prototype-based oracle for selection-fusion ensembles. In *2010 20th International Conference on Pattern Recognition (ICPR)*, pages 77–80, 2010.

[48] Juan J. Rodriguez and Ludmila I. Kuncheva. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

[49] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8): 832–844, 1998.

[50] Yoav Freund and Robert E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[51] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[52] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[53] Leo Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996.

[54] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[55] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hilton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[56] Steven Richard Waterhouse. *Classification and Regression using Mixtures of Experts*. PhD thesis, Jesus College of Cambridge, 1997.

[57] Lior Rokach and Oded Maimon. *Data Mining with Decision Trees: Theory and Applications*. World Scientific, 2008.

[58] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[59] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[60] J. R. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[61] Sreerama K. Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.