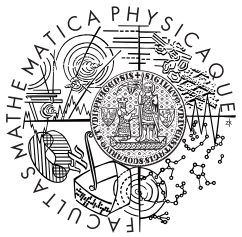
University of Cagliari



Charles University in Prague

Morpho-Colorimetric and Non-Parametric Analysis in Statistical Classification of Vascular Flora

Luca Frigau

Dissertation submitted for the
degree of Doctor of Philosophy in
Environmental and Applied Botany
for the University of Cagliari and in
Probability and Mathematical Statistics
for the Charles University in Prague.

Supervisors:
Prof. Francesco Mola, Ph.D.
Prof. RNDr. Jaromír Antoch, CSc.
Prof. Gianluigi Bacchetta, Ph.D.

Referees:
Prof. RNDr. Gejza Dohnal, CSc.
Prof. Dr. Adalbert F.X. Wilhelm

SECS-S/01 – BIO/03

Academic Year Final Examination 2014/2015

The results comprised in this dissertation have been obtained from 2012 to 2016 during a doctoral study under joint supervision of thesis between the University of Cagliari and the Charles University in Prague.

University of Cagliari

Class:	XXVII
Ph.D. course:	Environmental and Applied Botany
Ph.D. Board Chairman:	Prof. Gianluigi Bacchetta, Ph.D.
Department:	Environmental and Life Science

Charles University in Prague

Study program:	Mathematics
Subject field:	4-M-4 Probability and Mathematical Statistics
Department:	Probability and Mathematical Statistics
Faculty:	Mathematics and Physics
4-M-4 Board Chairman:	Prof. RNDr. Jaromír Antoch, CSc.

Statement of Honesty

I hereby declare that I have written this dissertation separately, independently and entirely with using quoted resources. I agree that the University Library shall make it available to borrowers under rules of the Library.

Cagliari, February 9th, 2016.

Luca Frigau

Permission of using published codes and data

All the functions present in Appendix A as well as the data, that will be freely available at the depository of Charles University in Prague, can be used by everybody who needed. If you use them in publications please cite this dissertation.



REGIONE AUTONOMA DELLA SARDEGNA


Luca Frigau gratefully acknowledges Sardinia Regional Government for the financial support of his Ph.D. 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.)

Acknowledgments

To begin with, I would like to thank prof. Francesco Mola, who has been my guide in these years teaching me a lot, both from the academic point of view, and from the human one. He believed in me, giving me confidence and freedom to develop my ideas pointing the way forward. In addition, I am deeply grateful to him for giving me the opportunity to do many experiences that, during these four years, have enriched me so much.

I would like to thank to prof. Gianluigi Bacchetta, who believed in my project and gave me the opportunity of being able to develop it.

A great thanks goes to prof. Jaromír Antoch, because he devoted much time to me, and without him my dissertation would have been very different. He has been also a great teacher whom I thank for all statistical invaluable advice, for useful discussions, for teaching me so much things and for so many beers taken together.

I would like to thank prof. Adalbert F.X. Wilhelm and prof. Gejza Dohnal for having reviewed my dissertation.

I would like to thank Claudio and Massimo because their door was always open when I needed advice or a help.

Special thanks goes to David and Lucie, who welcomed me like a son to their home in Durham and made me feel part of their family. Without them my experience at Duke University would have not been so special and complete.

Un grazie speciale va a mia nonna, alle mie zie e ai miei zii, che mi sono sempre stati vicini, incoraggiandomi e sostenendomi in tutte le scelte che ho fatto.
Un grazie particolare va a mia madre, a mio padre e a mio fratello, per tutto quello che hanno fatto per me in questi ultimi quattro anni ma, soprattutto, per quello che hanno fatto negli anni precedenti che mi ha permesso di essere oggi quello che sono. Mi hanno insegnato ad essere curioso e a non fermarmi mai al traguardo raggiunto, ma ad andare sempre avanti, *Ad maiora!*

Finally, I thank Vivi, the most important person. Because during these four long years you have always been close to me, and in the darkest moments you heartened me and gave me strength to go forward. Because you have always been patient and with your love you erased the physical distance that separated us for so long time. Ti amo.

Luca

Abstract

This dissertation concerns the task of classifying objects by methods of morpho-colorimetric and non-parametric statistical analysis. The study can be applied in botany where manual classification of seeds is still a common practice. It is labor-intensive, subjective, and suffers from contradiction, as well as it is a time-consuming task even for highly specialized botanists. Starting from this problem, automated, consistent, and efficient algorithms of classification of seeds have been developed allowing the researcher to have a valid support for reducing drastically time for classification and, at the same time, an exploratory tool that detects latent patterns in data that human eye cannot identify.

Firstly an approach called Background Subtraction, that enhances the quality of segmentation process output of an image, has been proposed. From RGB images it allows to get more precise binary images which need of a reduced intervention of manual correction. Then for each object data concern size, texture, color and shape have been extracted. These have been used as input for classification process, in which four classifiers have been performed: Linear Discriminant Analysis (LDA), Classification And Regression Trees (CART), Support Vector Machines (SVM) and Naïve Bayes (NB). In order to enhance the classification accuracy an approach of classifier combination, indicated as C_A, has been developed. It consists in spliting the complex problem of classifying among D classes into $D-1$ sub problems less complex than the original one, each of them classifying between only two classes. Combining the four classifiers considered, C_A allowed to reduce of 25% the misclassification error obtained by the best of the four classifiers. Finally, approach aimed at evaluating the reliability of a classification rule has been proposed.

The algorithms proposed are developed and optimized for botanical seeds, but they are suitable to a larger class of morphological classification problems. In order to make these algorithms usable and executable, functions have been created in R language and published.

Sommario

Questa tesi affronta il problema di classificare oggetti attraverso metodologie di analisi statistiche morfocolorimetriche e non parametriche. Lo studio trova immediata applicazione in campo botanico in cui la classificazione manuale dei semi è ancora una pratica comune. Partendo da questo problema, si sono sviluppati algoritmi di classificazione automatica dei semi consistenti ed efficienti, che permettono al ricercatore di avere un supporto che riduce drasticamente i tempi necessari per la classificazione e, allo stesso tempo, uno strumento esplorativo che offre la possibilità di portare alla luce pattern nascosti nei dati che l'occhio umano non può individuare.

Per prima cosa si è sviluppato un approccio chiamato Background Subtraction, che migliora notevolmente la qualità dell'output del processo di segmentazione di un'immagine, dando la possibilità di ottenere dalle immagini RGB immagini binarie più precise, che necessitano quindi di un ridotto intervento di correzione manuale. Successivamente per ogni oggetto sono stati estratti i dati che riguardano la dimensione, la texture interna, il colore, e la forma. Questi dati sono stati utilizzati come input nel processo di classificazione che ha previsto l'utilizzo di quattro classificatori: Linear Discriminant Analysis (LDA), Classification And Regression Trees (CART), Support Vector Machines (SVM) e Naïve Bayes (NB). Per migliorare l'accuratezza della classificazione si è sviluppato un approccio di combinazione di classificatori, indicato come C_A, che divide il problema complesso di classificare tra D classi, in $D-1$ sotto-problemi meno complessi di quello originale, ognuno dei quali che classifica tra due classi. Combinando i quattro classificatori considerati, C_A ha permesso di ridurre del 25% l'errore di classificazione ottenuto dal migliore dei quattro classificatori. Infine, si è sviluppato un modello per misurare l'affidabilità dei classificatori e renderne più facile il confronto.

Gli algoritmi sono stati sviluppati e ottimizzati per i semi botanici, nonostante ciò questi sono adatti ad una più ampia classe di problemi di classificazione morfologica. Per rendere utilizzabili ed eseguibili questi algoritmi sono state scritte e pubblicate le relative funzioni in linguaggio R.

Abstrakt

Tato dizertace se zabývá statistickými aspekty klasifikace botanických objektu, v daném případě naskenovaných obrazu semen rostlin. Soustřeďuje se především na kvalitu klasifikace a na automatizaci celého procesu klasifikace. Snahou je též odstranit rozpory zpusobené lidskými chybami.

V první kapitole se práce soustřeďuje na kroky potřebné k získání statistických dat z naskenovaných obrazu. Jedná se o postupy umožňující zvýšení kontrastu obrazu, detekci obrysu, či odstranění šumu.

V druhé kapitole je pozornost soustředěna na nástroje moderní morfometrie a teoretické koncepty analýzy tvaru. Soustřeďujeme se především na koncept význačných značek a jejich matematické transformace, jež umožňují data popsat ať již pomocí Kendalových či Booksteinových souřadnic. Vedle toho jsou použity postupy Fourierovy analýzy, neboť umožňují velmi úsporně popsat geometrickou informaci o tvarech zkoumaných objektu.

Ve třetí kapitole je prezentován originální přístup umožňující kombinaci tzv. klasifikačních stromu. Tento přístup reaguje na to, že v daném kontextu je třeba klasifikovat do velkého počtu tříd (řádově desítky až stovky). Navržený přístup kombinuje dichotomické dělení a využívá predikční kvality jednotlivých klasifikátoru, které jsou kombinovány. Praktické výpočty na reálných botanických datech ukázaly, že zvolený přístup zlepšuje jak predikční vlastnosti, tak spolehlivost navrženého klasifikátoru, a činí jej pro danou situaci vhodným.

Ve čtvrté kapitole jsou prezentovány originální výsledky týkající se ohodnocení spolehlivosti navrženého klasifikačního pravidla. Pro tento účel je klasifikátor opětovně trénován na tzv. bootstrapových datech. Dále je použit beta regresní model umožňující získat váhy pro jednotlivé bootrapem získané klasifikátory. Na jejich základě je poté konstruován index spolehlivosti. Navržený přístup je testován na reálných datech.

V páté kapitola, a mnohem podrobněji v Apendixu A, jsou popsány funkce v jazyce R umožňující provádět klasifikaci podle postupu popsaných v předešlých kapitolách a ohodnotit jejich spolehlivost.

Poslední kapitola je aplikační a ukazuje jak navržené postupu fungují při klasifikaci reálných botanických dat. Jedná se o data z germaplasmové banky semen shromážděná v Cagliari. Výsledky jsou porovnány s jinými přístupy dříve navrženými v literatuře pro klasifikaci podobných objektu. Je též diskutována eficience zvoleného přístupu.

Contents

Introduction

Extinctions have always been a part of Earth's history. Pimm et al. (1995) demonstrated an unambiguous evidence of human impact on floras and fauna extinction. Their rates improved 100 to 1000 times their pre-human levels. He considered the human population exponential increase as a concrete threat for the biodiversity. For stopping the biodiversity loss, or at any rate for slowing down it, in 1992 the United Nations Environment Programme (UNEP) promoted the adoption of the agreed text of the Convention on Biological Diversity (CBD). Its aim was to promote the sustainable development and the conservation of biological diversity. Within the framework of the CBD, the Parties adopted in 2002 the Global Strategy for Plant Conservation (GSPC) committing to deliver 16 specific targets for halting the continuing loss of plant diversity, by 2010. In 2010 the Parties signed the Updated GSPC 2011–2020 with new targets to deliver by 2020. In Europe the two most important efforts for plant conservation are the "Council Directive 92/43/EEC of 21 May 1992 on the conservation of natural habitats and of wild fauna and flora" and Natura 2000 network, which protect both animals and plant species considered of European importance.

To achieve those targets regarding the halt of loss of plant diversity, two strategies are possible: *in situ* and *ex situ* plant conservation[1]. These two strategies are complementary, the conservation of plant biodiversity can be achieved through an integrated approach balancing in situ and ex situ conservation strategies (Laliberté, 1997). On one side *in situ* approach allows natural selection to act, which cannot be recreated *ex situ*, and to achieve the ultimate aim of biodiversity conservation, i.e. the maintenance of wild species in their natural habitat. At the same time, it is impossible to halt habitat destructions and so it is necessary to preserve *ex situ* threatened species before they become extinct. Furthermore latter approach provide opportunity to study their biology and to understand the threats for future successful species restoration and reintroduction. Although *in situ* conservation strategy is considered the best one for preserving plant diversity (Burrascano et al., 2009), its measures are more expensive than *ex situ* ones. Li and Pritchard (2009) estimated costs for *ex situ* conservation as little as 1% of those needed for conserving the species *in situ*, although *ex situ* conservation must address some technical challenges. Among all *ex situ* methods, the most effective is storage of plant seeds in seed banks. It permits to save large amounts of genetic material in a small space and with minimum risk of genetic damage (Iriondo and Perez, 1999).

For this reason, in the last two decades, both consideration and application of *ex situ* conservation approach have enhanced. Thereby several seed banks and other structures have been

[1] *In situ* conservation consists in protecting threatened plant species in their natural habitat, whereas *ex situ* conservation in protecting them outside their natural habitat.

established. Due to the increasing number of seeds gathered, more attention has been focused on classification of accessions in entry. Manual classification of seeds is still a common practice. It is labor-intensive, subjective, and suffers from contradiction, as well as it is a time-consuming task even for highly specialized botanists, and the increasing number of seeds to classify is making unbearable time spent for classification. A possible solution for this problem is to develop a reliable and computerized method that is able to make real-time classification of seeds. Although it is not possible that the computer algorithm learns knowledges of botany or plant biology, it may detect latent patterns in data.

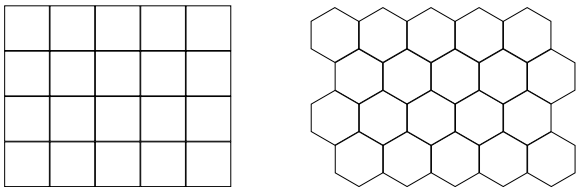
The objective of this dissertation is to provide an automated, consistent, and efficient method of morphological classification of seeds through extracting information directly from their digital images. The method proposed is developed and optimized for botanical seeds, but it is suitable to a larger class of morphological classification problems.

Chapter 1

Image Processing

The concept of "image" is quite clear to everybody, because we deal with it everyday. In mathematics, image can be modeled by a continuous function of two variables $f(x, y)$ where (x, y) are coordinates in a plane, or maybe even three variables (x, y, t) where t is time (see, among others, Shapiro and Stockman (2001) and Sonka et al. (2014)). Nevertheless, image processing often deals with static images, and for this reason we consider time t as constant and, consequently, $f(x, y)$ as the continuous function that models image. If image is grayscale, then $f(x, y) \to [0, 1]$ is a scalar function, whereas if it is expressed in color mode, its dimension is three or four. Depending on combination which primary colors are used, it is possible to distinguish different color spaces, the most common being RGB and CMYK. In the RGB the value of a particular color is expressed as a vector of three elements, i.e. $f(x, y) \to (R_i, G_i, B_i)$, where $(R_i, G_i, B_i) \in [0, 1]^3$ and R, G and B represent the intensity respectively of Red, Green and Blue color channels, whereas $i = (x, y)$ indicates pixel. Instead, in CMYK the values are expressed as a vector of four elements, i.e. $f(x, y) \to (C_i, M_i, Y_i, K_i)$, where $(C_i, M_i, Y_i, K_i) \in [0, 1]^4$ and C, M, Y and K represent the intensity respectively of Cyan, Magenta, Yellow, and Key (i.e. Black) color channels (Hunt, 2005).

An image to be processed has to be represented using an appropriate discrete data structure, for instance, a matrix. As a result, it is necessary to apply a process that transforms the continuous function $f(x, y)$ to a discrete one. To do that image is first captured by digitalization, which samples the continuous function $f(x, y)$ into a $M \times N$ matrix. These sampling points (i.e. pixels) are ordered in the plane, following a geometric relation (called grid), usually square or hexagonal (Figure 1.7).



(a) (b)

Figure 1.1: *Square and hexagonal grid.*

Then to each continuous sample is assigned an integer value by image quantization, that is, the continuous range of the image function $f(x, y)$ is split into K intervals. Consequently the finer the sampling (i.e., the larger M and N) and quantization (the larger K), the better the approximation of the continuous image function $f(x, y)$ is achieved.

In order to extract information contained on images for further statistical analyses, it is necessary to transform them into input for the methods. This operation is very important because all results can be strongly influenced by the data input accuracy. The data manipulation typically consists of three key stages described in detail below:

1. Image pre-processing.
   (a) Color to grayscale contrast enhancement image conversion.
2. Image segmentation
   (a) Background subtraction.
   (b) Recognition of objects in images.
   (c) Quality enhancing of objects.
3. Extraction of data from identified objects.

1.1 Image pre-processing

Before handling information contained in images, it is necessary to perform some pre-processing operations, i.e. dealing typically with operations with images at the lowest level of abstraction, both input and output are intensity images. They do not increase image information content but usually decrease it. Nevertheless, pre-processing is very important because it helps to remove information that is not relevant to the specific image analysis task. As a result they enhance the quality of the image removing undesired distortions or improving image features important for further processing.Several methods are included in pre-processing, such as image resampling, grayscale contrast enhancement, noise removal and manual correction.

1.1.1 Color to grayscale contrast enhancement image conversion

The first operation is to convert an image expressed by a RGB color space to grayscale contrast enhancement. In the RGB color space the value of a particular color is expressed as a vector of three elements (R_i, G_i, B_i) where R, G and B represent the intensity respectively of Red, Green and Blue color channels, whereas i indicates the pixel. In other words the conversion is an operation which transforms the three element vector (R_i, G_i, B_i) to one element vector T_i for each image pixel. The conversion is valuable because an image with a single dimension guarantees a simpler application of segmentation methods, but as consequent it provides a loss of information. Hence several methods of color to grayscale conversion have been proposed in literature with the aim of saving as much as possible information during reduction process. Often the new dimension obtained by reduction process, corresponds to luminance channel and conveys only luminance information, ignoring chromatic one. In this way important information are not taken into account, because even if luminance predominates human visual processing, it is possible it does not catch the perceived color contrasts. Ĉadík (2008) designed a study for evaluating seven state-of-art color to grayscale

conversions by two subjective perceptual experiments. The seven conversion methods chosen are listed below:

- The first method consists in taking a luminance channel as a gray representation, neglecting the chrominance channels. It is an approach very simple and widely used. For doing that, the channel Y of the CIE XYZ color space has been used (Fairchild, 2013). The CIE XYZ is a color space defined by the International Commission on Illumination (CIE) and made up of three components: X is a linear combination of cone response curves chosen to be nonnegative, Y is luminance and Z is quasi-equal to blue stimulation.

- The second method is that proposed by Bala and Eschbach (2004). They deal with loss of information in case of taking only luminance channel or a derivative thereof as a gray representation. The problem is in distinction between two different colors of similar luminance, which might be important if the two colors are spatially adjacent. Their method locally preserves distinction between adjacent colors by introducing high-frequency chrominance information (i.e. the chrominance information that is changing rapidly on a short distance scale) into the luminance channel. This information is added only in those regions that containing high-frequency chrominance information by a spatial high-pass filter.

- The third method is known as *Decolorize*. Proposed by Grundland and Dodgson (2005), it features for contrast enhancing in color to grayscale conversion by increasing luminance to reflect chromatic differences. It is explained more in detail hereinafter.

- The fourth method, implemented by Gooch et al. (2005), is called *Color2Gray*. In defining pixel gray values, they are iteratively adjusted to minimize local contrasts between all the pixel pairs. To reduce its high computational complexity, it is possible to limit pixel comparison or to use an approach developed by Mantiuk et al. (2006), which speeds up the algorithm.

- The fifth method has been proposed by Rasche et al. (2005). Its aim is to preserve contrast while maintaining consistent luminance. The optimal conversion is gained by minimizing an error function formulate by authors, which is based on matching the gray differences to the corresponding color differences.

- The sixth method is by Neumann et al. (2007). They presented two local, gradient based, grayscale conversion techniques: one is based on the CIELab color space, and the other on the Coloroid color system. The CIELab is a color space made up of three components: the first represents the lightness of the color, the second its position between red/magenta and green, and the last one its position between yellow and blue. The Coloroid is a color space in which colors are fundamentally specified by luminosity, saturation and hue. In this method, first they obtain an inconsistent gradient field from a color image applying one of the techniques. In the second step they select the closest consistent gradient field, which is later corrected using a gradient inconsistency correction method. Finally, it is integrated to produce a grayscale image.

- The seventh method is implemented by Smith et al. (2008). It applies a two-step approach: first a global one and then a local one. In the first step gray values are globally assigned by Helmholtz-Kohlrausch effect. In the second one only regions that do not represent sufficiently original chromatic contrast are modified.

For comparison Ĉadík (2008) applied a psychophysical technique of paired comparisons known as the two-alternatives forced choice (2AFC) experiment paradigm (David, 1963). It consists of two experiments. In the first one two grayscale images were shown at a time to observer together with the color original in the middle. To observer was asked: *"Your task is to select the grayscale image that better matches the colors of the original color image"*. It was done to measure reproducing accuracy. In the second experiment two grayscale images were shown at a time to observer without any reference. In this case the question asked to observer was: *"Your task is to select the preferred grayscale image from the presented pair"*. The aim of this second experiment is to measure preference of observer among all grayscale images without any reference. Decolorize method achieved the highest preference score and the second highest accuracy score, as well as its overall score has been the best one. For this reason we decided to apply it in our image analysis.

The Decolorize algorithm input is an RGB image $(R_i, G_i, B_i) \in [0,1]^3$, whereas its output is a grayscale image $T_i \in [0,1]$. It is controlled by three parameters: the degree of image enhancement $\lambda \in [0,1]$; the typical size of relevant image features in pixels $\sigma \in [1, min(\text{picture dimensions})]$; the proportion of image pixels assumed to be outliers $\eta \in [0, 0.5)$. The algorithm consists of five steps, which are explained in detailed below:

I The color image is converted into a color opponent color space.

II The color differences are measured using a Gaussian sampling.

III The chrominance projection axis is found by predominant component analysis.

IV The luminance and chrominance information are merged.

V The dynamic range is adjusted using the saturation information.

I. The color image is converted into a color opponent color space.

In the first step the vector (R_i, G_i, B_i) is transformed into the vector (Y_i, P_i, Q_i) using the following linear transformation proposed by the authors (i.e. Grundland and Dodgson (2005))

$$\begin{bmatrix} Y_i \\ P_i \\ Q_i \end{bmatrix} = \begin{bmatrix} 0.2989 & 0.5870 & 0.1140 \\ 0.5000 & 0.5000 & -1.0000 \\ 1.0000 & -1.0000 & 0.0000 \end{bmatrix} \begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix} \tag{1.1}$$

The output vector is made up of an achromatic luminance channel $Y_i \in [0,1]$ and two chromatic opponent color channels: the yellow-blue $P_i \in [-1,1]$ and the red-green $Q_i \in [-1,1]$ channels. The perpendicular chromatic axes P_i and Q_i support an easy calculation of hue $H_i \in [-1,1]$ and saturation $S_i \in [0, 1.1180]$

$$H_i = \frac{1}{\pi} tan^{-1}\left(\frac{Q_i}{P_i}\right) \quad \text{and} \quad S_i = \sqrt{P_i^2 + Q_i^2} \tag{1.2}$$

The luminance channel Y_i provides the default color to grayscale image. It conforms to the NTSC-Rec.601 standard luminance axis, which has a length of axis $Y_{axis} = 0.6686$.

II. The color differences are measured using a Gaussian sampling.

In order to analyze the distribution of color contrasts between image features, the color differences between pixels are considered. In this step each pixel X_i is paired with another pixel X_i'. In that pairing process, since a fixed size neighborhood is unable to catch contrasts in image features of different sizes, a randomized scheme is applied: sampling by Gaussian pairing. It consists in choosing X_i' randomly according to a displacement vector from an isotropic bivariate Gaussian. In other words, for each X_i the horizontal and vertical size neighborhood for locating X_i' are each drawn from a univariate Gaussian distribution with mean equal to 0 and variance to $(2/\pi)\sigma^2$, getting σ as the expected distance between paired pixels. σ is the typical size of relevant color contrast features, to which it is often assigned the value $\sqrt{2}\min(\text{picture dimensions})$.

III. The chrominance projection axis is found by predominant component analysis.

The chrominance projection axis is the color axis that represents the chromatic contrasts lost when the luminance channel supplies the color to grayscale mapping. To achieve this goal the authors applied predominant component analysis. Unlike principal component analysis which optimizes the variability of observations, predominant component analysis optimizes the differences between observations. The predominant chromatic axis aims to capture the color contrast information that is lost in the luminance channel. First of all, it is necessary to calculate the color deference between the pixels X_i and X_i', paired in the previous step

$$\begin{aligned} &\Delta R_i = R_i - R_i' \quad \Delta G_i = G_i - G_i', \quad \Delta B_i = B_i - B_i', \\ &\Delta Y_i = Y_i - Y_i' \quad \Delta P_i = P_i - P_i', \quad \Delta Q_i = Q_i - Q_i' \end{aligned} \tag{1.3}$$

For defining the importance of enhancing color contrasts, contrast loss ratio $c_i \in [0, 1]$ is calculated. It measures the relative loss of contrast incurred when luminance differences ΔY_i are used to represent the RGB color differences ΔD_i

$$c_i = \begin{cases} \dfrac{\Delta D_i - (1/Y_{axis})|\Delta Y_i|}{\Delta D_i} & \text{if } \Delta D_i \neq 0 \\ 0 & \text{if } \Delta D_i = 0 \end{cases} \tag{1.4}$$

where

$$\Delta D_i = \sqrt{\Delta R_i^2 + \Delta G_i^2 + \Delta B_i^2} \tag{1.5}$$

In the PQ chrominance plane, the predominant axis (Δp, Δq) of chromatic contrast is determined through a weighted sum of the oriented chromatic contrasts (ΔP_i, ΔQ_i) of the paired pixels

$$\Delta p = \sum o_i c_i \Delta P_i \quad \text{and} \quad \Delta q = \sum o_i c_i \Delta Q_i \tag{1.6}$$

where $o_i = sign(\Delta Y_i)$ expresses the orientation of each color contrast. It is decided by the ordering of luminance values.

IV. The luminance and chrominance information are merged.

In this step luminance and chrominance information are merged. Projecting the chromatic data (P_i, Q_i) onto the predominant chromatic axis (Δp, Δq) information are combined into the predominant chromatic data values K_i

$$K_i = P_i \Delta p + Q_i \Delta q \tag{1.7}$$

Later K_i is scaled to ignore the extreme values due to η level of image noise. The extreme values are detected by a linear time selection algorithm, Ψ_η and $\Psi_{1-\eta}$. It is a computational efficient algorithm that removes lower or upper quantiles of distribution, here indicated by η and $1-\eta$. The scaling produces the predominant chromatic channel C_i

$$C_i = \frac{K_i}{\Psi_{1-\eta}(|K|)} \tag{1.8}$$

The predominant chromatic channel C_i and the luminance channel Y_i are linearly combined with the degree of image enhancement λ to produce the enhanced luminance U_i

$$U_i = Y_i + \lambda C_i \tag{1.9}$$

It consists in an image-dependent linear combination of the original color (R_i, G_i, B_i), which maps linear color gradients to linear luminance gradients. Although the value of λ is discretional, the authors suggest to set it at 0.3.

V. The dynamic range is adjusted using the saturation information.

In order to avoid the effects of image noise, the dynamic range $[U_{min}, U_{max}]$ is defined

$$\begin{aligned} U_{min} &= \Psi_\eta(U) \\ U_{max} &= \Psi_{1-\eta}(U) \end{aligned} \tag{1.10}$$

To take into account the desired degree λ of contrast enhancement, it is necessary to define a correct dynamic range $[V_{min}, V_{max}]$

$$\begin{aligned} V_{min} &= \lambda Y_{min} + (1-\lambda)\Psi_\eta(Y) \\ V_{max} &= \lambda Y_{max} + (1-\lambda)\Psi_{1-\eta}(Y) \end{aligned} \tag{1.11}$$

The enhanced luminance U_i is linearly rescaled to obtain the corrected luminance V_i to fit the correct dynamic range $[V_{min}, V_{max}]$

$$V_i = V_{min} + \frac{(V_{max} - V_{min})}{(U_{max} - U_{min})}(U_i - U_{min}) \tag{1.12}$$

To be considered acceptable the gray level T_i, it has to be included in the range $[Y_{min}, Y_{max}]$ of luminance values. To ensure that achromatic pixels retain their luminance as their gray level, it is necessary that $|T_i - Y_i| \leqslant \lambda(Y_{max}/S_{max})S_i$. This condition permits to achieve the desired degree λ of contrast enhancement. Hence it is possible to calculate a range $[E_i, F_i]$ in which T_i has to be included

$$\begin{aligned} E_i &= \max\left(Y_{min}, Y_i - \lambda\frac{Y_{max}}{S_{max}}S_i\right) \\ F_i &= \min\left(Y_{max}, Y_i + \lambda\frac{Y_{max}}{S_{max}}S_i\right) \end{aligned} \tag{1.13}$$

Finally the grayscale values $T_i \in [E_i, F_i] \subseteq [Y_{min}, Y_{max}]$ are obtained by clipping the corrected luminance V_i to conform to the saturation dependent bounds

$$T_i = \text{median}(E_i, V_i, F_i) \tag{1.14}$$

1.2 Image segmentation

In data manipulation process, one of the most important phases is image segmentation. Its aim is to partition an image into different "objects". Although it is trivial (almost always) to the human vision, object recognition is still one of the most challenging problems in image processing, image understanding and artificial intelligence (Chan and Shen, 2005).

1.2.1 Background subtraction

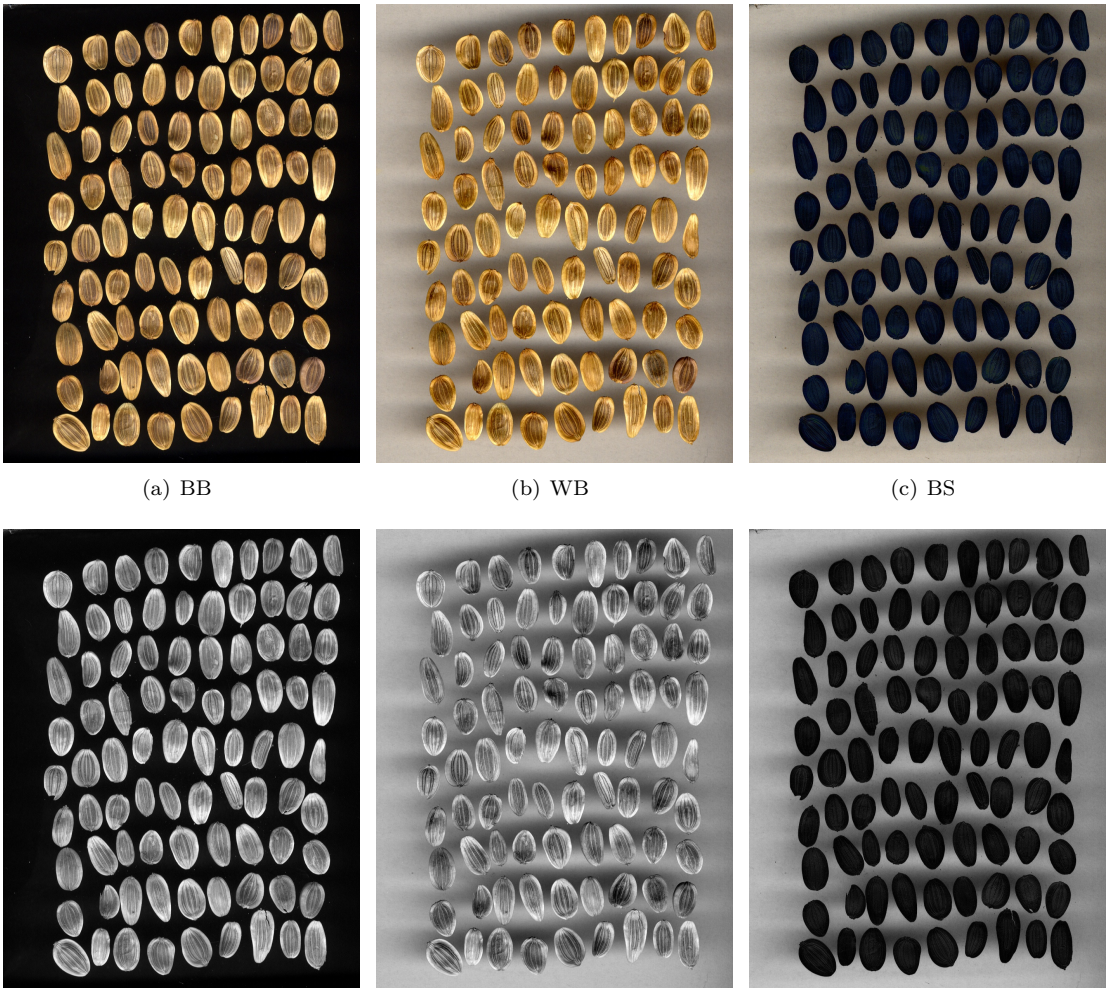
Normally, all information available for image segmentation process is hold in a single image, but often this information is not sufficient to obtain a good segmentation output. In fact, it is possible that objects convey within different information, which makes segmentation difficult even if background can be chosen freely. An example of how an object can convey non-homogeneous information is shown in Figure 1.2. It refers to a seed whose color on the left part is bright, whereas that in the right part is much darker. This means that on the left part the pixel intensity values are higher than those on the right part. This increases the complexity of image segmentation process, since it separates foreground from background as a function of a thresholding value of pixel intensities. Consequently if, as in the example, foreground objects present within both low and high intensity values, and background middle ones, it will be difficult to have a thresholding value able to separate correctly background from foreground. In order to solve this problem more information is needed. New information can be obtained by adding to segmentation process an additional image having the same foreground but with different background compared to original one. In literature it is not common to carry out that process using two images that differentiate themselves just for background because, due to time and memory storage issues, the image to analyze is just one. Despite that, nowadays memory storage capacity is increasing day by day, whereas the problem of time analysis is relative because it depends on the situation at hand.

The proposed approach to image segmentation is based on the following intuition. Since we want to take advantages of difference between the two backgrounds, it is in our opinion useful to enlarge that difference by choosing white and black as background colors of the two images. In order to use information added by the second image, it is possible to apply an approach a lot alike to background subtraction. Background subtraction (hence BS) is an approach widely used for detecting moving objects from a video. It consists of subtracting each image that arranges the video to its background image, that is an image with no moving objects (Piccardi, 2004), as a result, non-zero differences represent moving objects. In image subtraction the absolute difference between pixel intensities of the first image to those of the second one is performed. Whereas in BS what changes in images is foreground (i.e. objects), here it is the contrary: what changes is background. Therefore if subtraction is applied before segmentation process to the two images which differ just in the background, non-zero differences will represent background instead of foreground, and vice versa for zero ones. If an image is taken twice, usually it is very hard to have the pixel intensity

Figure 1.2: *Example of different information (i.e. pixel intensities) conveyed by objects.*

values of the first image identical to the correspondent ones of the second image, because some little changes in lighting often occur. As a result, it is very likely that the absolute difference between foreground pixels of the two images will assume tiny non-zero values. However it is anyway possible to distinguish between background and foreground. In fact we know that background changes from white to black, i.e. from high (close to 1) to low (close to 0) intensity values, so that their absolute difference will provide values with high intensity. Instead, foreground absolute difference will provide values close to zero. This situation is a perfect starting point of the image segmentation process, because the difference between background and foreground pixel intensities are now more pronounced then that considering one image only.

Let consider as example the images in Figure 1.3. If a person must choose which image to use for segmentation between 1.3(a) and 1.3(b), he definitely will opt for the first one, because there is a clearly better contrast between background and foreground. Although an image with black background (hence BB) can be the right one in a situation characterized by a foreground with low intensity values, that is bright color, it will be the wrong one in the contrary case, that is if the foreground intensity values are high. In order to remove the need of human choosing, the BS is a good approach for automatizing the process. In Figure 1.3(c) it is possible to see the output of the BS operation between the two images mentioned above. Comparing the three images the best option is clearly to have a BB whereas, at first, the other two seem to be almost at the same level as segmentation capacity, because in both there is some noise caused by the seed shadows. In order to figure out better the situation, we should compare them using images with a single dimension, because a segmentation process uses as input images with that characteristic. As a result we transform all of them from RGB images $(R_i, G_i, B_i) \in [0,1]^3$ to grayscale images $T_i \in [0,1]$. At this point Figures 1.3(d), 1.3(e) and 1.3(f) seem to confirm that the BB one is the best one, whereas the difference between the other two is now more marked. Nevertheless it is not still clear how the BS one is good. To go deeper in the analysis and to show better how the intensity values are distributed between background and foreground, four level plots have been produced (Figure 1.4). They consist in contour plots with the areas between the contours filled in



(a) BB (b) WB (c) BS

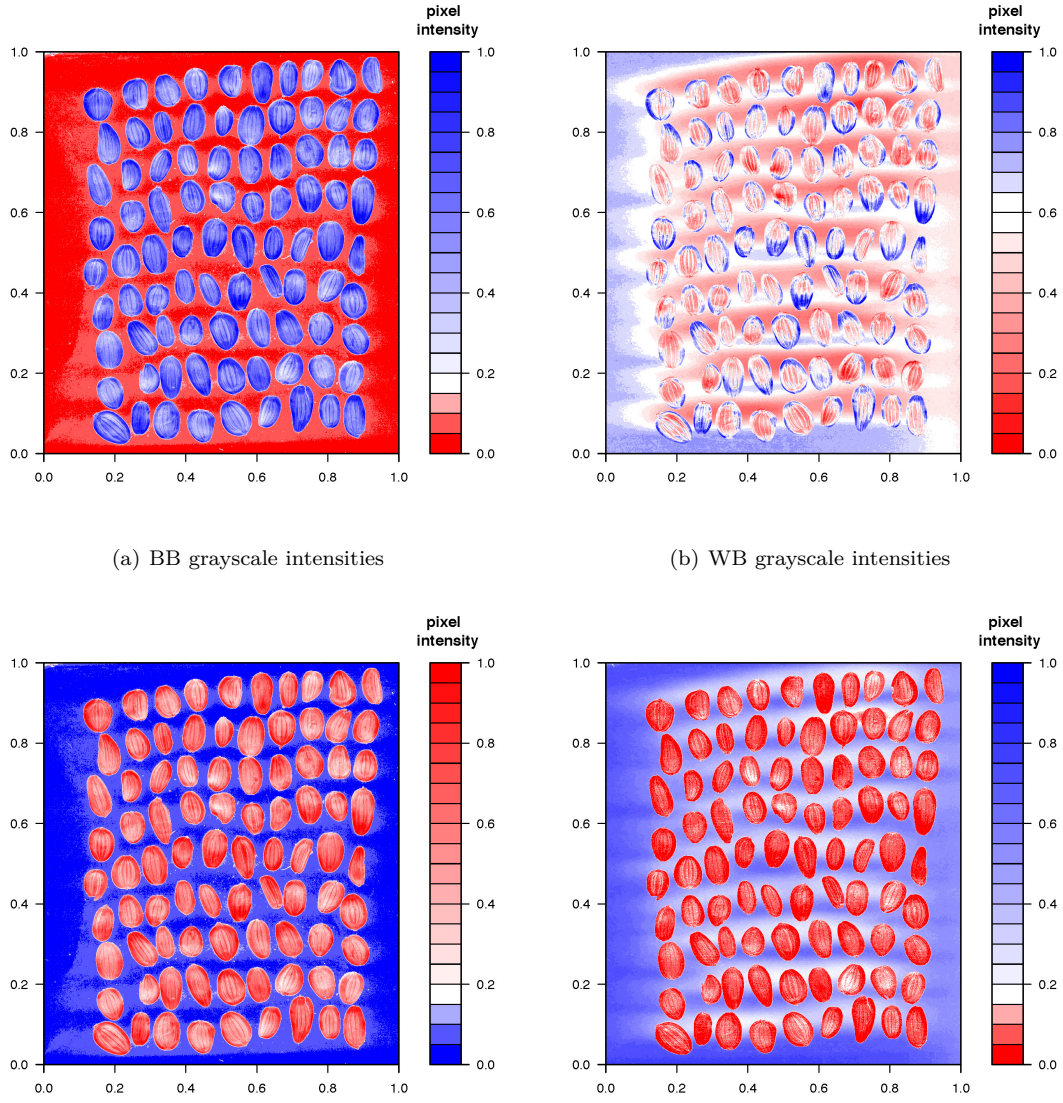
(d) BB grayscale (e) WB grayscale (f) BS grayscale

Figure 1.3: *Accession scanned changing (by column) background and (by row) color scale. In the columns are shown images with, respectively, black background (BB), white background (WB) and background subtraction (BS). In the first row the images are in RGB scale, whereas in the second one in grayscale.*

solid colors. Figure 1.4(b) shows that the white background (hence WB) image is absolute useless for segmentation process, in fact neither a global nor a local thresholding that segments the image in a acceptable way exists. On the contrary through Figure 1.4(a) it is possible to note that for the BB one a global thresholding would work good, and a local one likely even better. In order to get a comparison between the BB image and that of BS, the color scale of the former has been inverted so as to have background in blue and foreground in red. The comparison can be done through Figures 1.4(c) and 1.4(d). Although the BB one is definitely better, even the BS one allows performing a good segmentation process. In fact all its foreground object borders have intensity values lower than 0.15 and the majority even lower than 0.10. As regards background pixel values, almost all of them is higher than 0.15, only few pixels has lower values. Starting from this situation even global thresholding will be able to do a successful segmentation. Anyway, as described below, a local thresholding method allows performing a better segmentation and avoiding problems concern parts where background pixel values are higher, because thresholding value is not fixed, but changes locally. Finally it is possible to state that BS method provides good results in segmentation allowing to automate the process when foreground color of images is not known *a priori* or constant.

In order to validate these statements and to check how these approaches work in more difficult situations, a further comparison is made between the BS approach and the standard methods, i.e. using a single image as input. In other words, the segmentation process has been run three times: the first by using as input the image obtained from the BS procedure, the second time with the BB image and in the last case with the WB image. Finally, in order to compare the differences in using Local and Global thresholding approaches, the three segmentation processes have been run by applying both Sauvola's method and Otsu's method (see Section 1.2.2 for details). Let us start defining the input of the segmentation process. The first task is the definition of the objects (i.e. the foreground). In order to arise complexity of the image segmentation process, we have decided to consider six seeds characterized by different inner pixel intensities. They have been scanned twice using a BB and a WB, then the absolute difference between the pixel intensities of these two images is performed, obtaining the BS one. In the next step they were converted to grayscale contrast enhancement images, obtaining the top images illustrated in Figure 1.5. The results of the segmentation processes are shown in the second and third row of Figure 1.5, where binary images show the foreground objects identified by each method. It stands out that the application of BS operation provides us the best results for both Otsu's and Sauvola's methods, but it is simple to note that the best one is achieved by the local method. Since the output obtained using the Sauvola's method on BS image (image in the bottom-right part of Figure 1.5) can be clearly considered as the best one, it would be interesting to measure how much the other outputs are similar to it. To do that, we calculated how many pixels have been classified in the same way. The worst results are achieved when using the WB as input image. Only the 44.65% and 59.38% of image pixels have been classified, respectively for Otsu's and Sauvola's methods, as the best output. The inefficiency is due to the similarity between intensity of seed color and background, and to the presence of shadows that obstructs a correct definition of the thresholds. A better result is achieved by using the BB as input image. Here, the percentages of pixels classified as the best output go up to 95.58% and 68.44%, respectively for Otsu's and Sauvola's methods. The main reason is the stronger contrast between seed color and background. In this case we can note that a global threshold method works better than a local one, in contrast with what has been observed in the WB situation. The motivation of this result is that global approaches tend to suffer a lot for the presence of shadows, inasmuch just one threshold value is calculated for all pixels.

In order to assess if a classifier is able to validate the results provided by a segmentation method,



(a) BB grayscale intensities

(b) WB grayscale intensities

(c) BB grayscale intensities (scale color inverted)

(d) BS grayscale intensities

Figure 1.4: *Visualization of the pixel intensity values of BB, WB and BS, after conversion to grayscale, colored through a palette.*

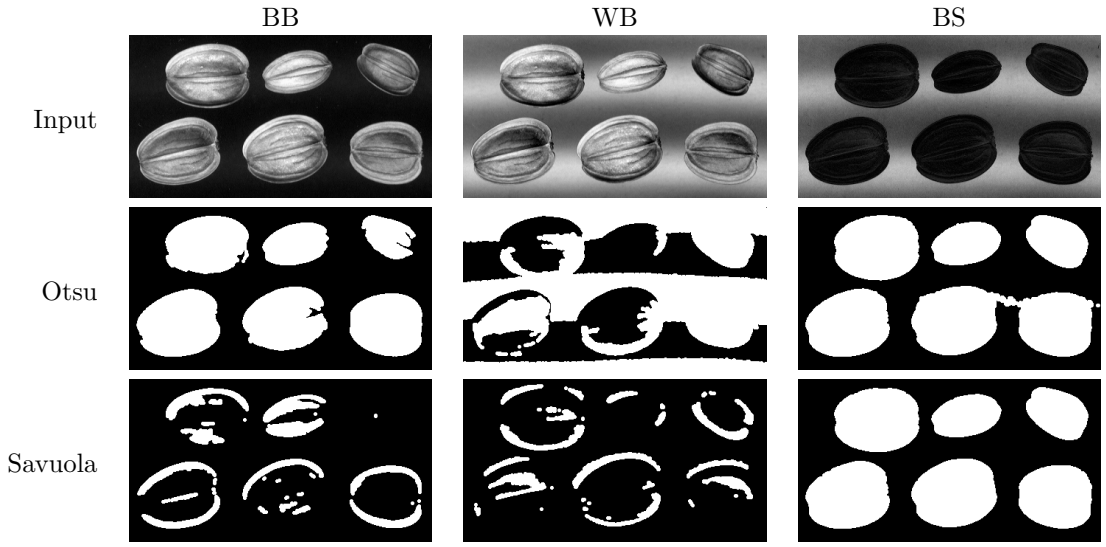


Figure 1.5: *The 9 figures, arranged in a matrix* 3×3*, show the different combination achieved by applying two different segmentation approaches to three different input images. In the first row, the input images are placed, whereas in the last two rows the outputs achieved by applying, respectively, the methods of Otsu and Sauvola are shown. Looking at the table column by column allows us to consider the different input images: the BB one (first column), the WB one (second column) and the BS one (third column).*

a classification experiment has been performed. The basic idea motivating this experiment is the following: the classification of single pixels into a background or foreground category obtained from a segmentation method is used as (binary) response variable by a classifier and observations (pixels) are classified into one of the two categories on the basis of the RGB intensities. A good performance of the classifier is an indication of reliability of the image segmentation procedure used to obtain the observed categories of the response variable. To accomplish this goal, Decision Trees implementing the FAST algorithm provided by Mola and Siciliano (1997) are used as classifiers. To run the validation experiment, we split the original data (91,052 pixels) into a training set and a test set. The first is composed of a proportion of 5% of the original data. This low value is chosen in order to increase the complexity of the classification experiment and, in this way, to further validate the effectiveness of the segmentation method. The validation experiment involves 6 alternative settings: for each one we consider the pixels as background or foreground taking into account individually the output of the segmentation methods presented above (2nd and 3rd row of Figure 1.5) and reported in the first column of Figure 1.6, where the borders of foreground objects are projected in green into the respective original RGB images. Schematically, the features of the different experimental settings are described from the second to the fourth column of Table 1.1: for instance, in the first experimental setting the input image is the one presenting a BB (Figure 1.6, 1st row-1st column) which is segmented by applying the Savuola's method. The output of this segmentation process, that is, the classification into background or foreground categories obtained for each pixel, allows us to define the response variable of the classification experiment which is run by using, in turn, the RGB intensities deriving by the BB, the WB and the BS approach applied on the same image. Thus, for each setting we have 3 classification experiments so that the total number of experiments is 18. A classification tree is grown for each experiment on the training data and a final tree is selected through pruning with 10-fold cross-validation. Next, the pruned tree is used to predict the response class (background or foreground) for test set observations. The fifth column of Table 1.1 reports the error rates on test set observations produced by each pruned classification tree in each experiment. The same results are shown graphically in Figure 1.6, from the 2nd to the 4th column. Here, the green points in each image represent the misclassified pixels.

Results of the classification experiments should be analyzed on a twofold perspective: first, it is natural to assume that in each setting the best classification is the one obtained when the origin of the RGB intensities and that of the input image are the same: this means, for example, that in the first setting we expect that the minimum error rate is the one obtained when using the RGB intensities deriving from the BB image since the latter is the image processed, in this setting, with the Savuola's segmentation method. The second perspective involves the investigation of the best performing classification tree, in order to understand in which experiment the classification tree is able to better classify the two categories of the response variable on the basis of the RGB intensities and consequently to validate the outcome of a segmentation method: in this respect, Table 1.1 shows that there is no question in reporting that the lowest values of the error rate are those obtained when the input image is the one pre-processed with the BS method and the RGB intensities are those obtained from the same method.

In order to simultaneously consider both the above mentioned perspectives, a coherence index is defined in the last column of Table 1.1. It is equivalent to a rating indicator with three possible categories. The first category is $(-)$: it refers to a situation in which there is no equivalence between the pre-processing method used before image segmentation is performed (Input image) and the method defining the origin of the RGB intensities which provides the lowest error rate. This is the case of the first setting, where it was expected that the origin of the RGB intensities deriving from

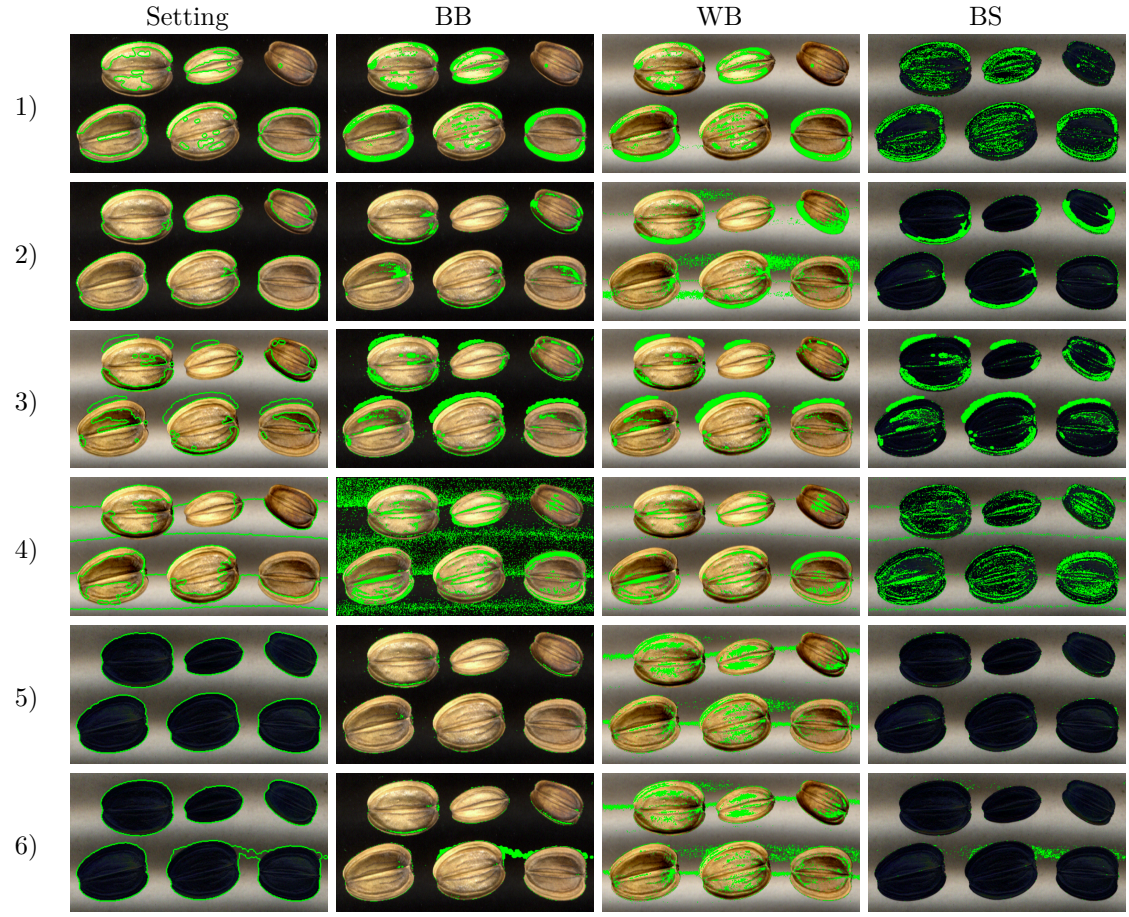


Figure 1.6: *The 24 figures, arranged in a matrix* 6×4*, show the different predictions achieved by applying classification trees algorithm to three different input images. In the first column the 6 possible settings are shown. They correspond to the 6 possible ways considering pixels as background or foreground, originated from the results of the segmentation processes above (2nd and 3rd row of Figure 1.5), where the borders of foreground objects are projected in green into the original RGB images. In the setting 1 is considered the result of the segmentation process that used as input the BB image and as method the Sauvola's one; the second one the BB image and Otsu's method; the third one, the WB image and Sauvola's method; the fourth one, the WB image and Otsu's method; the fifth one, the BS image and Sauvola's method; the sixth one, the BS image and Otsu's method. Instead, the other columns show the graphical output of the predictions, where the green points represent the misclassified pixels.*

Setting	Input image	Method	Origin of RGB intensities	Error rate (%)	Coherence
1	BB	Savuola	BB	9.69	(−)
			WB	9.40	
			BS	11.38	
2	BB	Otsu	BB	3.14	(+)
			WB	9.81	
			BS	4.32	
3	WB	Savuola	BB	8.58	(+)
			WB	7.42	
			BS	10.15	
4	WB	Otsu	BB	20.52	(+)
			WB	6.74	
			BS	14.62	
5	BS	Savuola	BB	11.40	(++)
			WB	8.70	
			BS	**0.53**	
6	BS	Otsu	BB	2.16	(++)
			WB	9.05	
			BS	**0.98**	

Table 1.1: *Settings and results of the validation experiments.*

the BB method leads to the best performing classification tree but the best result is achieved in the WB case. The second category of the coherence index (+) indicates equivalence between the pre-processing method and the origin of the RGB intensities. This is the case of Settings 2 to 4 where such an equivalence exists. The third category of coherence is (++) and refers to a situation in which the pre-processing method corresponds to the origin of the RGB intensities (similar to the (+) case) but, at the same time, the lowest error rate (best performing classification tree) is less than half with respect to the error rate produced by the second best classification tree: this is the case of Settings 5 and 6 where BS method provides an error rate which is less than half if compared to the one deriving from the WB method (Setting 5) and the BB one (Setting 6).

The BS approach provided, in the example presented above, the best results in terms of quality of segmentation for both Otsu's method and Sauvola's method. In the segmentation of images involving botanic seeds, the main problems occurred with a single image as input have been the shadows and the non-homogeneous intensity values of foreground pixels. Both problems have been overcome by the BS approach. Another important result achievable by BS is the automation of the segmentation process. In fact the absolute difference between the pixel intensities of images with the same foreground, allows us to obtain a new "artificial" image always characterized by tiny non-zero values in correspondence of foreground pixels, independently from original values of foreground pixels. As a result if we use BS approach choosing as background colors of the two images white and black, we expect to obtain a good result, as in the example presented above, independently from foreground pixel intensity values and their inner homogeneity.

1.2.2 Recognition of objects in images

In literature several image segmentation techniques exist, but there is not a single method that can be considered preferable for all images. According to Fu and Mui (1981), the segmentation techniques can be divided into three classes:

- Characteristic feature thresholding.
- Edge detection.
- Region extraction.

The characteristic feature thresholding represents the group of the most used and oldest segmentation methods. The edge detection is a group of segmentation methods based on information about edges. In the first step it detects object edges by operators using discontinuities in gray level, color or texture. In the second step the edges are combined in chains enhancing their quality.

The aim of region extraction, instead, is to construct regions directly. Gray level thresholding is one of the most used techniques for image segmentation. Thresholding can be interpreted as the transformation of an image g to a binary image o.

$$o(x,y) = \begin{cases} 0 & \text{for } g(x,y) < T \\ 1 & \text{for } g(x,y) \geqslant T \end{cases} \tag{1.15}$$

where T is the threshold value, $o(x,y) = 1$ for foreground pixels and $o(x,y) = 0$ for background pixels (Sonka et al., 2014). The most critical task of this method is selection of a correct threshold, which is essential for a successful segmentation. In this technic it is possible to use global or local information, and as a consequence to distinguish between Global and Local thresholding.

Global thresholding consists in finding a single threshold value for whole image. It is very fast, but its results are not good if illumination over the image is not uniform. In seed images with white background it is common to have seed shadows. Although image subtraction provides background pixels featured by high intensity values, shadows in white background image could compromised it giving to background pixel some values not so high which will be close to foreground pixel values. To overcome this problem it is preferred to use Local thresholding, which is characterized by calculating a threshold value for each pixel using information of their neighbor pixels.

In local thresholding the aim is to calculate a threshold $t(x,y)$ for each pixel, in order to assign the value of 0 (background) or 1 (foreground). Letting $g(x,y)$ as the grayscale intensity value of pixel at location (x,y) and $o(x,y)$ its value after thresholding

$$o(x,y) = \begin{cases} 0 & \text{if } g(x,y) < t(x,y) \\ 1 & \text{otherwise} \end{cases} \tag{1.16}$$

Badekas and Papamarkos (2005) studied seven binarization algorithms, and found the Otsu's method (Otsu, 1975) and the Sauvola's method (Sauvola and Pietikäinen, 2000) as the two best performing ones.

Otsu suggested to calculate global threshold value T after analyzing the gray level distribution between the two classes, indicated by letters F (foreground) and B (background). Let us assume that the image is represented using L gray levels $\{0, 1, \ldots, L-1\}$, denote by n_k number of pixels with gray level k and put $N = \sum_k n_k$. Then we can describe the probability distribution of gray

levels as $\{k, p_k\}_{k=0}^{L-1}$ with $p_k = n_k/N$, and to calculate for a fixed value T, $0 < T < L$, probability that a pixel belongs to foreground (F) or background (B) class as

$$\pi_F = \sum_{k=0}^{T} p_k \quad \text{and} \quad \pi_B = \sum_{k=T+1}^{L-1} p_k. \tag{1.17}$$

Now it is possible to calculate means and variances of the gray level of each class and of the whole image as

$$m_F = \sum_{k=0}^{T} k p_k, \quad m_B = \sum_{k=T+1}^{L-1} k p_k \quad \text{and} \quad m = \pi_F m_F + \pi_B v m_B, \tag{1.18}$$

$$\sigma_F^2 = \sum_{k=0}^{T} \big(k - m_F\big)^2 p_k \quad \text{and} \quad \sigma_B^2 = \sum_{k=T+1}^{L-1} \big(k - m_B\big)^2 p_k. \tag{1.19}$$

From here we get for a fixed value of T the between and within variances as

$$\sigma_{between}^2(T) = \pi_F(m_F - m)^2 + \pi_B(m_B - m)^2 \quad \text{and} \quad \sigma_{within}^2(T) = \pi_F \sigma_F^2 + \pi_B \sigma_B^2. \tag{1.20}$$

Finally, searched threshold value of T is obtained by maximizing

$$\eta(T) = \frac{\sigma_{between}^2(T)}{\sigma_{within}^2(T)}, \quad \text{i.e.} \quad T = \underset{0<T<L}{\operatorname{argmax}}\, \eta(T). \tag{1.21}$$

Sauvola's method (Sauvola and Pietikäinen, 2000), instead, calculates $t(x, y)$ by using the mean $m(x, y)$ and standard deviation $s(x, y)$ of intensity values included in a $W \times W$ window centered in the pixel (x, y)

$$t(x, y) = m(x, y) \left[1 + k \left(\frac{s(x, y)}{R} - 1\right)\right] \tag{1.22}$$

where R is the maximum value of the standard deviation and k is a parameter which assumes positive values in the range $[0.2, 0.5]$ and controls the value of the threshold in the local window. Badekas and Papamarkos (2005) found that $k = 0.34$ gives the best results in their study. The main shortcoming of Sauvola's algorithm is its high computational complexity. In fact computing $m(x, y)$ and $s(x, y)$ produces a computational complexity of $O(W^2N^2)$ for an $N \times N$ image and a $W \times W$ window. For solving that Sauvola and Pietikäinen (2000) proposed to calculate a $t(x, y)$ only for every nth pixel, and estimate the others by interpolation. A better solution has been proposed by Shafait et al. (2008) solving directly the computational problem. They proposed of computing $m(x, y)$ and $s(x, y)$ using integral images. Viola and Jones (2004) defined the integral image of an image g as a new image in which the intensity value of each pixel is equal to the sum of all intensity values of pixels above and to the left of it, inclusive the pixel itself. Consequently the intensity at position (x, y) is

$$I(x, y) = \sum_{i=1}^{x} \sum_{j=1}^{y} g(i, j) \tag{1.23}$$

From that is possible to compute efficiently both $m(x, y)$ and $s(x, y)$

$$\begin{aligned} m(x,y) =& (I\,(x + w/2, y + w/2) + I\,(x - w/2, y - w/2) \\ & - I\,(x + w/2, y - w/2) - I\,(x - w/2, y + w/2))/w^2 \\ s^2(x,y) =& \frac{1}{w^2} \sum_{i=x-w/2}^{x+w/2} \sum_{j=y-w/2}^{y+w/2} g^2(i,j) - m^2(x,y) \end{aligned} \tag{1.24}$$

The computation of integral image allows to compute local means and variances in a computationally efficient way, independent of the local window size. In fact, the computational complexity plummets from $O(W^2N^2)$ to $O(N^2)$.

1.2.3 Quality enhancing of objects

During the thresholding some background pixel could have been categorized as foreground one and vice versa. For this reason it is necessary to apply some correction. They are performed by mathematical morphology. It is a theory which provides a number of useful tools for image analysis, and it is based on the assumption that an image consists of structures which may be handled by set theory (Glasbey and Horgan, 1995). In binary images it is possible to apply mathematical morphology to enhance their quality, affecting their form, structure or shape. Since we want to apply operations to foreground pixels, we take it to be the set of interest, and the background pixels set as the complement. A basic concept of morphological operations is structuring element. It is the tool used by morphological operators for transforming binary image objects. It can be interpreted as a shape that interacts with all image pixels changing, on some condition, their setting. For instance, let the structuring element S be the simple 2×2 set shown in Figure 1.7(a), we can place S at any pixel in an image using the reference pixel, whose position defines where the structuring element has been placed (rotation of the structuring element is not allowed). The choice of this reference pixel is often arbitrary, often the centre pixel is chosen if the set is symmetric.

Two main morphological operators are *erosion* and *dilation* introduced in Serra (1982). Erosion is the most basic one. Let A a binary image and S a structuring element. If the reference pixel of S is placed at (i, j), we denote it as $S_{(i,j)}$. The erosion of A by S is defined as the set of all pixel locations for which S placed at that pixel is contained within A (Definition 1.2.1 (Serra, 1982)).

Definition 1.2.1 *The erosion of a set (binary image) A by the structure element S is given by the set operation*

$$A \ominus S = \{(i, j) : S_{(i,j)} \subset A\}$$

For instance, let us consider Figure 1.7(a). S is defined by a block of 4 pixels, using the top left corner as reference pixel, and A is a binary image. The result of $A \ominus S$ is shown in Figure 1.7(b). In order to figure out how the erosion may be performed on an image we first note that if $S_{(i,j)}$ consists of the 4 pixels (i, j), $(i + 1, j)$, $(i, j + 1)$ and $(i + 1, j + 1)$, that is

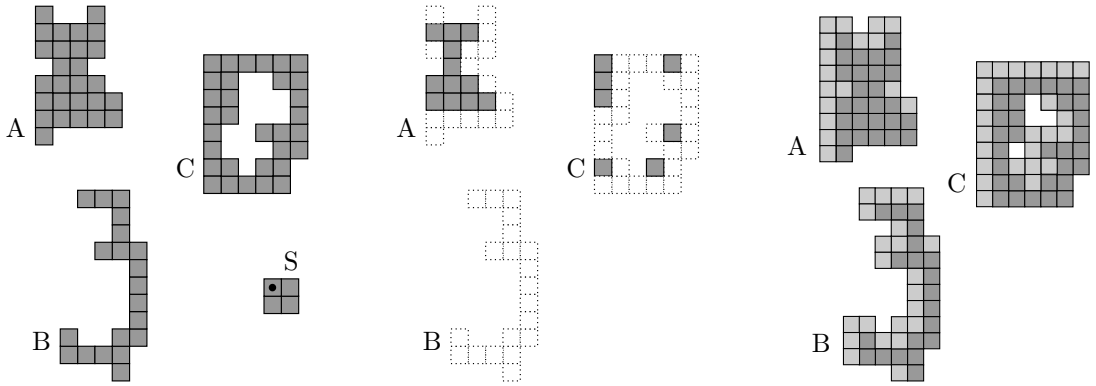
$$S_{(i,j)} = \{(i, j), (i + 1, j), (i, j + 1), (i + 1, j + 1)\} \tag{1.25}$$

If (and only if) all of these are "foreground" pixels, the pixel (i, j), in the eroded image, will be "foreground".

A complementary operation to that of erosion is *dilation* (Figure 1.7(c)). If A^c denotes the complement of A^c, then the dilation of a set A by a set S, denoted $A \oplus S$, is defined by $A \oplus S = (A^c \ominus S)^c$ (Glasbey and Horgan, 1995). In a direct way, instead, it can be defined as the set of all pixel locations for which the intersection of S placed at that pixel contained with A is not equal to the null set (Definition 1.2.2 (Serra, 1982)).

Definition 1.2.2 *The dilation of a set (binary image) A by the structure element S is given by the set operation*

$$A \oplus S = \{(i, j) : S_{(i,j)} \cap A \neq \emptyset\}$$



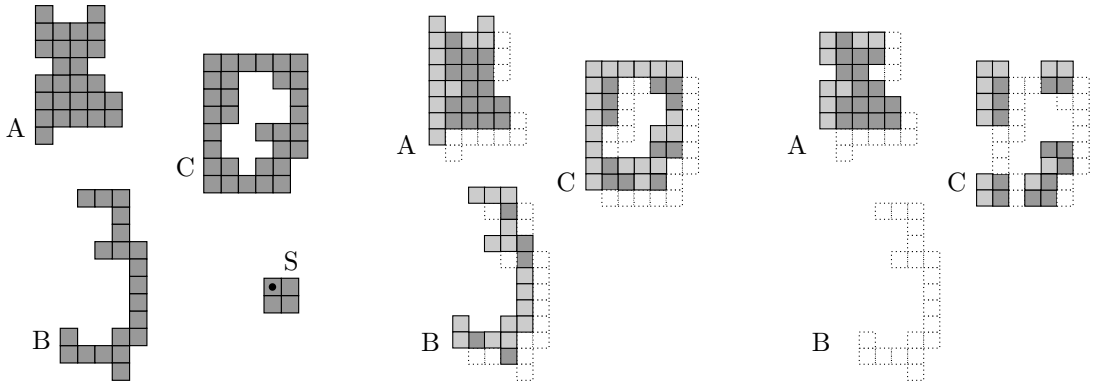
(a) Original shapes (b) After erosion (c) After dilation

Figure 1.7: *In Figure 1.7(a) three sets A, B and C with different shapes are drawn together with a structuring element S, where $\bullet$ in the top left pixel indicates that we have selected this as the reference pixel. The Figure 1.7(b) and Figure 1.7(c) show the output, respectively, after erosion and dilation with the structuring element S. In light gray it is drawn what was not part of the shape but after operation is become a new part of it. Instead in dashed line it is drawn what was part of the shape but not more after operation.*

Two very performed operations are *opening* and *closing*, which are often denoted, respectively, by $\psi_S(A)$ and $\phi_S(A)$. They are defined as

$$\begin{aligned} \psi_S(A) &= (A \ominus S) \oplus S^c \\ \phi_S(A) &= (A \oplus S) \ominus S^c \end{aligned} \tag{1.26}$$

so that opening consists of an erosion followed by a dilation, and closing a dilation followed by an erosion. An example is illustrated in Figure 1.8. They are complementary, as a result applying one A is equivalent to applying the other to A^c.



(a) Original shapes (b) After opening (c) After closing

Figure 1.8: *In Figure 1.8(a) three sets A, B and C with different shapes are drawn together with a structuring element S, where • in the top left pixel indicates that we have selected this as the reference pixel. The Figure 1.8(b) and Figure 1.8(c) show the output, respectively, after opening and closing with the structuring element S. In light gray it is drawn what was not part of the shape but after operation is become a new part of it. Instead in dashed line it is drawn what was part of the shape but not more after operation.*

1.3 Extraction of data from identified objects

Once binary image is treated, its objects are formally detached and, in the last key stage of data manipulation, several statistical indexes can be computed in order to describe them. This type of information concerns with linear measurements and other indexes that describe objects in an unidimensional way, consequently complex information conveyed by objects is overly summarized. Although more modern tools are available today, these statistics are still used in morphological analysis, since they are often relevant and significant.

We focused on 32 indexes, six of which concern the *size* of objects, and the other 26 that describe their *texture*. Let us start describing the former ones. Their comprehension is definitely immediate, since each one handled them already once in his life. We give a formal definition through the equations used for calculated them. Let an image $G(x, y)$ with $0 \leqslant x \leqslant N_x - 1$ and $0 \leqslant y \leqslant N_y - 1$, F_k the set of foreground its pixels (i, j) that define an object k, and consider $I_A(\cdot)$ the indicator function of A.

Area size

$$area_k = \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} I_{F_k}((i,j)) = \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} \omega \tag{1.27}$$

Perimeter

$$perimeter_k = \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} I_{\{1,\ldots,8\}} \left(\sum_{r=-1}^{1} \sum_{v=-1}^{1} I_{F_k}((i+r, j+v))\omega \right) = \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} \gamma \tag{1.28}$$

Mean radius

$$radius.mean_k = \frac{\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}\gamma\sqrt{(i-c_{x,k})^2+(j-c_{y,k})^2}}{\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}\gamma} \tag{1.29}$$

where

$$c_{x,k} = \frac{\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}(i\omega)}{\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}\omega} \quad \text{and} \quad c_{y,k} = \frac{\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}(j\omega)}{\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}\omega} \tag{1.30}$$

Standard deviation of the mean radius

$$radius.sd_k = \sqrt{\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}\left(\gamma\sqrt{(i-c_{x,k})^2+(j-c_{y,k})^2} - \text{radius.mean}_k\right)^2} \tag{1.31}$$

Minimum radius

$$radius.min_k = \min_{i,j}\gamma\sqrt{(i-c_{x,k})^2+(j-c_{y,k})^2} \tag{1.32}$$

Maximum radius

$$radius.max_k = \max_{i,j}\gamma\sqrt{(i-c_{x,k})^2+(j-c_{y,k})^2} \tag{1.33}$$

In order to define qualities of texture, spatial distribution of gray values have to be considered. Haralick et al. (1973) proposed two steps for texture features extraction: the first is computing the co-occurrence matrix, which summarizes information about the relationship between two neighboring pixels, and the second step is calculating texture features on the base of the co-occurrence matrix. Let us consider a grayscale image as a matrix $G(x, y)$, with $0 \leqslant x \leqslant N_x - 1$ and $0 \leqslant y \leqslant N_y - 1$, where each matrix element, that is, each pixel value, is an integer over the range $[0, N_g - 1]$. N_g is the number of gray levels in the image, and usually it is a power of 2. The $N_g \times N_g$ Gray-Level Co-occurrence Matrix (GLCM) is a matrix $P_{d,\theta}$ for a displacement vector $d(dx, dy)$ and direction θ is defined as follows. The element (i, j) of $P_{d,\theta}$ is the number of times that the pair of gray levels i and j occurs considering the distance d between i and j following the direction θ.

$$P_{d,\theta}(i,j) = \#\left\{((r,s),(t,v)) : G(r,s) = i, G(t,v) = j\right\} \tag{1.34}$$

Where (r, s), $(t, v) \in N_x \times N_y$ $(t, v) = (r + dx, s + dy)$. As shown in Figure 1.9, it is possible to consider up to eight directions: two horizontal (0 and 180), two vertical (90 and 270), two left diagonal (135 and 315) and two right diagonal (45 and 225). In most cases all eight directions are

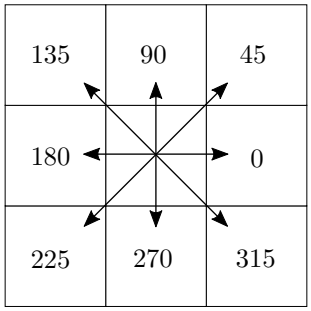


Figure 1.9: *Eight directions of adjacency.*

considered all together, that is $\theta = \{0, 45, 90, 135, 180, 225, 270, 315\}$. For instance, let us consider the 5×5 grayscale image G shown in Figure 1.10, where $N_g = 4$ such that $[0, 3]$ is the grayscale pixel values range. Let us calculate the $P_{d,\theta}$ for G considering a distance $d = 1$, we obtain the result shown in Figure 1.10(c) if the direction is right horizontal, that is $\theta = 0$, whereas that in Figure 1.10(d) if all directions are considered. Let us explain how a single value of a GLCM is calculated in practice. For example, the count of three in element $[0, 0]$ of Figure 1.10(c) was found by looking for a 0 followed by another 0 (given $d = 1$) in each row from left to right (inasmuch $\theta = 0$). The three cases are $((3, 3), (3, 4)), ((4, 1), (4, 2)), ((4, 2), (4, 3))$ and $((4, 3), (4, 4))$. Since the frequencies of GLCM depend on parameters distance and direction, in order to be able to compare different GLCM, it is necessary to normalize them. The normalization of $P_{d,\theta}(i, j)$ is performed dividing each of its entry by R, obtaining $p_{d,\theta}(i, j)$, which represents an estimate of the probability of two pixels appearing in a spatial relationship in which one pixel has intensity i and the other one has intensity j.

$$p_{d,\theta}(i,j) = \frac{P_{d,\theta}(i,j)}{R}, \qquad R = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} P_{d,\theta}(i,j) \tag{1.35}$$

From the GLCM, Haralick et al. (1973) proposed several useful texture features.

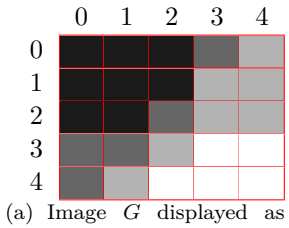
Angular Second Moment Feature The angular second-moment feature f_1 is a measure of homogeneity of the image.

$$f_1 = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} p_{d,\theta}(i,j)^2 \tag{1.36}$$

In a homogeneous image there are very few dominant gray-tone transitions. f_1 is high when image has very good homogeneity or when pixels are very similar.

Contrast Feature Contrast measures the amount of local variations present in the image, that is the variations of intensity or those of gray-level between the reference pixel and its neighbor.

$$f_2 = \sum_{n=0}^{N_g-1} n^2 \left\{ \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} p_{d,\theta}(i,j) \right\}, \text{ where } n = |i - j| \tag{1.37}$$



(a) Image G displayed as shades of gray

	0	1	2	3	4
0	3	3	3	2	1
1	3	3	3	1	1
2	3	3	2	1	1
3	2	2	1	0	0
4	2	1	0	0	0

(b) Gray levels of G pixels

	0	1	2	3
0	3	0	0	0
1	2	2	0	0
2	0	4	1	0
3	0	1	2	5

(c) $P_{1,0}$

	0	1	2	3
0	16	8	1	0
1	8	18	10	4
2	1	10	8	9
3	0	4	9	35

(d) $P_{1,\xi}$

Figure 1.10: *In 1.10(a) it is displayed the* 5×5 *image* G *in grayscale, whereas 1.10(b) indicates the gray levels for each pixel. The other Figures represent two Gray-Level Co-occurrence Matrices for* G*, applying a distance* $d = 1$ *and two different directions. In 1.10(c) the right horizontal direction (*$\theta = 0$*) is considered, whereas in 1.10(d) all the eight possible ones (*$\theta = \xi$*, where* $\xi = \{0, 45, 90, 135, 180, 225, 270, 315\}$*).*

To each cell it is assigned a weight equal to n^2. If $i = j$ the weight is 0, since the values of cells on diagonal represent the pixels totally similar to their neighbor. Instead, since the larger is the value $|i - j|$ the larger is the difference between pixels and their neighbor, the weight increases exponentially as the value $|i - j|$ increases.

Correlation Feature Correlation feature defines if a linear dependency of gray level values exists in the co-occurrence matrix, that is how a reference pixel is related to its neighbor, 0 is uncorrelated, 1 is perfectly correlated.

$$f_3 = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} p_{d,\theta}(i,j) \frac{(i-\mu_x)(j-\mu_y)}{\sigma_x \sigma_y} \tag{1.38}$$

where μ_x, μ_y, σ_x, σ_y are the means and standard deviations of p_x and p_y (the marginal distributions).

$$\begin{gathered} \mu_x = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} i p_{d,\theta}(i,j), \quad \mu_y = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} j p_{d,\theta}(i,j), \\ \sigma_x = \sqrt{\sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} (i-\mu_x)^2 p_{d,\theta}(i,j)} \quad \text{and} \quad \sigma_y = \sqrt{\sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} (j-\mu_y)^2 p_{d,\theta}(i,j)} \end{gathered} \tag{1.39}$$

Variance Feature Variance feature measures the dispersion of the difference between the reference

and the neighbor pixels

$$f_4 = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} (i-\mu)^2 p_{d,\theta}(i,j) \tag{1.40}$$

Inverse Difference Moment Feature This feature measures the local homogeneity of an image calculating how the elements of the GLCM elements are close to the diagonal.

$$f_5 = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} \frac{1}{1+(i-j)^2} p_{d,\theta}(i,j) \tag{1.41}$$

In this case the weight is the inverse of the Contrast weight. For the diagonal elements the weight is equal to 1, whereas it weight decreases exponentially away from the diagonal.

Sum Average Feature

$$f_6 = \sum_{i=0}^{2(N_g-1)} i p_{x+y}(i) \tag{1.42}$$

where

$$p_{x+y}(k) = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} p_{d,\theta}(i,j), \quad k = i+j = \{0,1,2,\ldots,2(N_g-1)\} \tag{1.43}$$

Sum Variance Feature

$$f_7 = \sum_{i=0}^{2(N_g-1)} (i-f_6)^2 p_{x+y}(i) \tag{1.44}$$

Sum Entropy Feature

$$f_8 = -\sum_{i=0}^{2(N_g-1)} p_{x+y}(i) \log(p_{x+y}(i)) \tag{1.45}$$

Entropy Feature The entropy can be interpreted as amount of irremediable chaos or disorder

$$f_9 = -\sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} p_{d,\theta}(i,j) \log(p_{d,\theta}(i,j)) \tag{1.46}$$

Difference Variance Feature

$$f_{10} = \sum_{i=0}^{N_g-1} \left(i - f'_{10}\right)^2 p_{x-y} \tag{1.47}$$

where

$$p_{x-y}(k) = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} p_{d,\theta}(i,j), \quad k = |i-j| = \{0,1,2,\ldots,(N_g-1)\} \tag{1.48}$$

Difference Entropy Feature

$$f_{11} = -\sum_{i=0}^{N_g-1} p_{x-y}(i)\log(p_{x-y}(i)) \tag{1.49}$$

Information Measures of Correlation 1 and 2 Features

$$\begin{aligned} f_{12} &= \frac{HXY - HXY1}{\max(HX, HY)} \\ f_{13} &= \left(1 - e^{-2(HXY2 - HXY)}\right)^{1/2} \end{aligned} \tag{1.50}$$

where

$$\begin{aligned} p_x(i) &= \sum_{j=0}^{N_g-1} p_{d,\theta}(i,j) \\ p_y(j) &= \sum_{i=0}^{N_g-1} p_{d,\theta}(i,j) \\ HX &= -\sum_{i=0}^{N_g-1} p_x(i)\log(p_x(i)) \\ HY &= -\sum_{i=0}^{N_g-1} p_y(i)\log(p_y(i)) \\ HXY &= -\sum_{i=0}^{N_g-1}\sum_{j=0}^{N_g-1} p_{d,\theta}(i,j)\log(p_{d,\theta}(i,j)) \\ HXY1 &= -\sum_{i=0}^{N_g-1}\sum_{j=0}^{N_g-1} p_{d,\theta}(i,j)\log(p_x(i)p_y(j)) \\ HXY2 &= -\sum_{i=0}^{N_g-1}\sum_{j=0}^{N_g-1} p_x(i)p_y(j)\log(p_x(i)p_y(j)) \end{aligned} \tag{1.51}$$

Maximal Correlation Coefficient Feature

$$f_{14} = (\text{second largest eigenvalue of } Q)^{1/2} \tag{1.52}$$

where

$$Q(i,j) = \sum_k \frac{p(i,k)p(j,k)}{p_x(i)p_y(k)} \tag{1.53}$$

From each of these 14 features, the mean and range have to be computed, resulting in 28 features. Nevertheless we computed just 26, since the maximal correlation coefficient was not calculated due to computational instability.

Chapter 2

Modern morphometrics and shape analysis

In the traditional morphometric analysis data are gathered without regard for allometry (i.e. shape variation). The shape is described just by *ad-hoc* measured distances such as length and width, which are compared in an uni- or multivariate framework. Although they are very often used in literature, there exists a lot of biases and weaknesses connected to them. First information is collected unequally by region as well as by orientation. Furthermore most elements tend to be aligned with one of axes, so a lot of information in the data is redundant whereas other information is missing. For these reasons, their usefulness is limited in solving real biological problem (Strauss and Bookstein, 1982). On the other hand, the modern morphometrics considers shape as a whole, taking into account all the geometrical relationships of the input data. Furthermore it allows a visualization and a geometrical interpretation of variation or change of the whole configuration. Among main approaches belong study of landmark configurations and the outline analysis. Both allow shape reconstruction from their numerical signature, i.e. preserve the geometrical information. This is very important because it permits to define some functional links between the shape and its variation.

The concept of "shape" is quite clear to everybody, since it is one of the first concepts a human being deals with in his life. Normally we refers to it as the appearance of an object. Nevertheless if we want to give a better definition, we can consider that given by Kendall (1977)

Definition 2.0.1 *Shape is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object.*

In other words to objects are characterized by the same shape if they can be translated, rescaled and rotated to each other so that they match exactly. In order to describe an object's shape often a name of second more familiar shape is used, for instance, it is common to say "to have a nose like a potato". The problem of this way to describe a shape is its subjectivity, so that it obviously cannot be apply in serious application. A solution to that problem is to locate a set of points on each object. These points, called landmarks, are defined by Dryden and Mardia (1998) in this way

Definition 2.0.2 *A landmark is a point of correspondence on each object that matches between and within populations.*

Configuration of landmarks is specific set of point locations on a biological form or image of a form located according to some rule. Bookstein (1997) provided a landmark type classification system, in function of rule sets used to locate landmarks:

Type I Landmarks as discrete juxtapositions of tissues, i.e. mathematical points whose topological homology is provided by biologically unique patterns on the form.

Type II Landmarks as maxima of curvature or other local morphogenetic processes, i.e. mathematical points whose topological homology is provided only by geometric, not biological or histological, criteria.

Type III Landmarks as extremal points, i.e. mathematical points constructed geometrically having at least one deficient coordinate (e.g. either end of a longest diameter).

Usually the third type of landmarks is not distinguishable from pseudo-landmarks, being points defined by construction. They can be sampled spacing equally the outline, or intersecting the outline by equally spaced transverse chords along the maximum diameter, or intersecting the outline by transverse chords with equal angles from the object centroid. Although pseudo-landmarks and the third type of landmarks do not provide the same amount of biological information such as the first type, they include geometric information in regions of the object where normally landmarks are under-sampled, and extract geometric properties of outline.

2.1 Configuration Space

Dealing with shape, an important concept to introduce is the *configuration space* of a object defined by landmarks (Dryden and Mardia, 1998).

Definition 2.1.1 *The configuration is the set of landmarks on a particular object. The configuration matrix* $\mathbf{X}$ *is the* $k \times d$ *matrix of Cartesian coordinates of the* k *landmarks in* d *dimensions. The configuration space is the space of all possible landmark coordinates.*

The number of landmarks can vary from case to case. Sometimes it is practically obligated since an object can have a specific number of specimens to be taken into account. In other case, instead, the object can have only few relevant specimens, so that in addition to them other points are set as landmarks so as to enhance the geometric information considered. Naturally the choice of landmark number is characterized by a trade-off between quantity of geometric information considered and complexity of the configuration space, which is defined in $\mathbb{R}^{dk}$ space.

2.2 Shape coordinate systems

Specifying the coordinate system is essential to describe an object's shape. A suitable coordinate system for shape has to be invariant under translation, scaling and rotation of the configuration.

2.2.1 Bookstein coordinates

Let $(x_j, y_j), j = 1, \ldots, k$ be $k \geqslant 3$ landmarks in a plane ($d = 2$ dimensions), and two specimen landmarks, $\mathbf{b}_1$ and $\mathbf{b}_2$, define a baseline. Bookstein (1984, 1986) proposed to remove the effects of location, scaling and rotation setting two specimen landmarks, defining a baseline, to a fixed position respectively to the coordinates $(-1/2, 0)$ and $(1/2, 0)$. In Figure 2.1 it is shown the simplest situation, that is, a configuration set of a triangle in which the points of the baseline (x_1, y_1) and (x_2, y_2) are positioned, respectively, at $(-1/2, 0)$ and $(1/2, 0)$ and the third point so that the shape stays the same. Dryden and Mardia (1998) give the following definition of Bookstein coordinates

Definition 2.2.1 *Bookstein coordinates* $(u_j^B, v_j^B)^T$, $j = 3, \ldots, k$ *are the remaining coordinates of an object after translating, rotating and rescaling the baseline to* $(-1/2, 0)$ *and* $(1/2, 0)$ *so that*

$$\begin{aligned} u_j^B &= \frac{(x_2 - x_1)(x_j - x_1) + (y_2 - y_1)(y_j - y_1)}{D_{12}^2} - \frac{1}{2}, \\ v_j^B &= \frac{(x_2 - x_1)(y_j - y_1) - (y_2 - y_1)(x_j - x_1)}{D_{12}^2} \end{aligned} \tag{2.1}$$

where $j = 3, \ldots, k$, $D_{12}^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2 > 0$ *and* $-\infty < u_j^B, v_j^B < \infty$.

Bookstein coordinates $\mathbf{b}_j = (u_j^B, v_j^B)$ for a fixed $j = 3, \ldots, k$, are calculated by finding the scale $a > 0$, the rotation matrix $\mathbf{\Gamma}$, and the translation $\mathbf{t} = (t_1, t_2)$ such that

$$\mathbf{b}_j = a\mathbf{\Gamma}(\mathbf{x}_j - \mathbf{t}) \tag{2.2}$$

where

$$\mathbf{\Gamma} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \tag{2.3}$$

Applying the transformation to landmarks $\mathbf{b}_1$ and $\mathbf{b}_2$, we get a system of four equations in four unknowns $(\mathbf{t}, \theta, a, \mathbf{\Gamma})$

$$\begin{aligned} a\mathbf{\Gamma}\left[\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} - \mathbf{t}\right] &= \begin{pmatrix} -1/2 \\ 0 \end{pmatrix}, \\ a\mathbf{\Gamma}\left[\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} - \mathbf{t}\right] &= \begin{pmatrix} 1/2 \\ 0 \end{pmatrix} \end{aligned} \tag{2.4}$$

After solving these equations we get

$$\begin{aligned} \mathbf{t} &= \left(\frac{1}{2}(x_1 + x_2), \frac{1}{2}(y_1 + y_2)\right), \\ \theta &= \arctan\frac{y_2 - y_1}{x_2 - x_1}, \\ a &= \left((x_2 + x_1)^2 + (y_2 + y_1)^2\right)^{-1/2}, \\ \mathbf{\Gamma} &= a\begin{pmatrix} x_2 - x_1 & y_2 - y_1 \\ -(y_2 - y_1) & x_2 - x_1 \end{pmatrix} \end{aligned} \tag{2.5}$$

Substituting (x_j, y_j), $j = 1, \ldots, k$ into Equation 2.5 and Equation 2.2, we get Equation 2.1. In that way a configuration $\mathbf{X}$ is rotated, translated and scaled onto the baseline.

Bookstein coordinates can be computed even using complex arithmetic. Starting from the original complex coordinates $z_1^o, \ldots, z_k^o$, the coordinates are defined as

$$u_j^B + iv_j^B = \frac{z_j^o - z_1^o}{z_2^o - z_1^o} - \frac{1}{2} = \frac{2z_j^o - z_1^o - z_2^o}{2(z_2^o - z_1^o)}, j = 3, \ldots, k. \tag{2.6}$$

The coordinate system of Bookstein coordinates is very used both by newcomer to shape analysis and by many people experienced in shape analysis in the first stages of an analysis. Nevertheless the problem of choosing the baseline push to use other coordinate system (Dryden and Mardia, 1998).

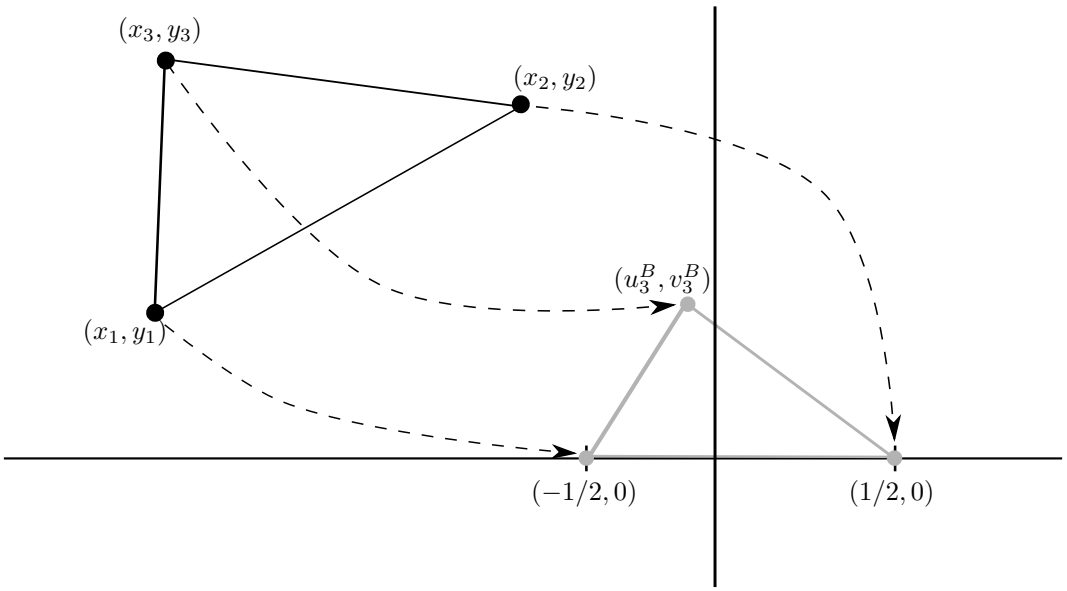


Figure 2.1: *Transformation of a configuration set of a triangle into Bookstein coordinates.*

2.2.2 Kendall coordinates

Kendall coordinates are very similar to Bookstein coordinates, what changes is that location is removed using the Helmert sub-matrix.

Let us define first the full Helmert matrix $\mathbf{H}^F$. It is a $k \times k$ orthogonal matrix with the first row equal to $1/\sqrt{k}$ and the other ones orthogonal to the first one. In order to get the Helmert sub-matrix $\mathbf{H}$ is enough to remove from $\mathbf{H}^F$ the first row. This is done to get the transformation $\mathbf{HX}$ independent from the original location of the configuration. Dryden and Mardia (1998) give the following definition of how to create a Helmert sub-matrix

Definition 2.2.2 *The jth row of the Helmert sub-matrix* $\mathbf{H}$ *is given by*

$$(h_j, \ldots, h_j, -jh_j, 0, \ldots, 0) \tag{2.7}$$

where $h_j = -(j(j+1))^{-1/2}$*, so that the jth row consists of* h_j *repeated* j *times, followed by* $-jh_j$ *and then* $k - j - 1$ *zeros,* $j = 1, \ldots, k-1$.

Consequently, for removing location from the original complex landmarks $\mathbf{z}^o = (z_1^o, \ldots, z_k^o)^T$, it is enough to pre-multiply it by Helmert sub-matrix to get $\mathbf{z}_H = \mathbf{H}\mathbf{z}^o = (z_1^o, \ldots, z_{k-1}^o)^T$.

Let us move now on the Kendall coordinates. Dryden and Mardia (1998) define those in the following way

Definition 2.2.3 *The Kendall coordinates are given by*

$$u_j^K + iv_j^K = \frac{z_j - 1}{z_1}, j = 3, \ldots, k \tag{2.8}$$

Between Kendall and Bookstein coordinates there is a simple 1-1 correspondence. For Bookstein coordinates let

$$w^B = (u_3^B + iv_3^B, \ldots, u_k^B + iv_k^B)^T \tag{2.9}$$

and for Kendall coordinates

$$w^K = (u_3^K + iv_3^K, \ldots, u_k^K + iv_k^K)^T \tag{2.10}$$

then it is possible to identify this relation

$$w^K = \sqrt{2}\mathbf{H}_1 w^B \tag{2.11}$$

where $\mathbf{H}_1$ is the lower right $(k-2) \times (k-2)$ partition of Helmert sub-matrix $\mathbf{H}$.

2.2.3 Tangent coordinates

In the shape space (see Section 2.3 for details) the metric is not Euclidean. This can become a problem in case we want to carry out traditional statistical methods, which relies on the Euclidean metric. To do that one must first project the surface of the hyperhemisphere of the shape space onto a "flat" tangent space where the metric is Euclidean. As a result, the Euclidean distance in tangent space is a good approximation to shape distances in shape space and general multivariate methods that use Euclidean metric can be performed to the points in the tangent space.
Let us define the vectorize operator in the folow way

$$\text{vec}(\mathbf{X}) = (\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_d^T)^T \tag{2.12}$$

A pole γ, usually the mean shape, is chosen on the pre-shape sphere in order to determinate the tangent plane. The pre-shape, $\mathbf{Z}$, is rotated to match γ as closely as possible by multiplying by $\hat{\mathbf{\Gamma}}$, choosing $\hat{\mathbf{\Gamma}}$ so as to minimize $\|\gamma - \mathbf{Z}\mathbf{\Gamma}\|^2$. The projection onto the tangent plane at γ gives the following tangent coordinates

$$v = (\mathbf{I}_{dk-d} - \text{vec}(\gamma)\text{vec}(\gamma)^T)\text{vec}(\mathbf{Z}\mathbf{\Gamma}) \tag{2.13}$$

2.3 Shape Space and distances

The shape of an object is the information that remains after removing location, orientation and scale effect. The location is filtered out either translating the object by adding a constant vector

$\mathbf{t} \in \mathbb{R}^d$ to coordinates of each point, or pre-multiplying its configuration with Helmert sub-matrix. The rotation is filtered out rotating the object by post-multiplying the configuration matrix $\mathbf{X}$ by a rotation matrix $\mathbf{\Gamma}$. If $d = 2$ the rotation matrix is

$$\mathbf{\Gamma}_\theta = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \tag{2.14}$$

$\mathbf{\Gamma}_\theta$ rotates clockwise the configuration matrix $\mathbf{X}$ by θ radiants. On the other hand, if $d = 3$ the rotation matrix is an orthonormal matrix given by

$$\mathbf{\Gamma} = \mathbf{\Gamma}_\phi \mathbf{\Gamma}_\omega \mathbf{\Gamma}_\theta \tag{2.15}$$

where

$$\mathbf{\Gamma}_\phi = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix}, \quad \mathbf{\Gamma}_\omega = \begin{pmatrix} \cos\omega & 0 & \sin\omega \\ 0 & 1 & 0 \\ -\sin\omega & 0 & \cos\omega \end{pmatrix}, \quad \mathbf{\Gamma}_\theta = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.16}$$

$\mathbf{\Gamma}_\phi$, $\mathbf{\Gamma}_\omega$ and $\mathbf{\Gamma}_\theta$ rotate anticlockwise the configuration matrix $\mathbf{X}$, respectively, by ϕ radiants on x-axis, by ω radiants on y-axis and by θ radiants on z-axis. Finally an isotropic scaling is obtained by multiply $\mathbf{X}$ by a number $a \in \mathbb{R}^+$.

Location, orientation and scale can be filtered out simultaneously carrying out a transformation called *Euclidean similarity transformation*, which is defined by Dryden and Mardia (1998) in the following way

Definition 2.3.1 *The Euclidean similarity transformations of a configuration matrix* $\mathbf{X}$ *are the set of translated, rotated and isotropically rescaled* $\mathbf{X}$ *i.e.*

$$\left\{ a\mathbf{X}\mathbf{\Gamma} + \mathbf{1}_k \mathbf{t}^T : a \in \mathbb{R}^+, \Gamma \in SO(d), \mathbf{t} \in \mathbb{R}^d \right\} \tag{2.17}$$

where a *is the scale,* $\mathbf{\Gamma}$ *is the rotation matrix,* $SO(d)$ *the set of all possible rotations in* d *dimensions and* $\mathbf{t}$ *a translation vector.*

Another approach, normally considered more convenient, is to remove the effect one at a time. In Figure 2.2 it is illustrated the hierarchical structure of spaces that can be obtained removing just one effect at a time.

2.3.1 Pre-form

Location is the easiest effect to remove, consequently it is the first one. In order to filter it out, it is possible to pre-multiply $\mathbf{X}$ with the Helmert sub-matrix

$$\mathbf{X}_H = \mathbf{H}\mathbf{X} \tag{2.18}$$

where $\mathbf{X}_H$ indicates the *Helmertized landmarks*. Another way is to center the configuration $\mathbf{X}$ to the origin of the Cartesian axes. The centered configuration $\mathbf{X}_c$ of $\mathbf{X} \in \mathbb{R}^{dk}$ is

$$\mathbf{X}_c = \mathbf{C}\mathbf{X} \tag{2.19}$$

where

$$\mathbf{C} = \mathbf{I}_k - \frac{1}{k}\mathbf{1}_k \mathbf{1}_k^T = \mathbf{H}^T\mathbf{H} \tag{2.20}$$

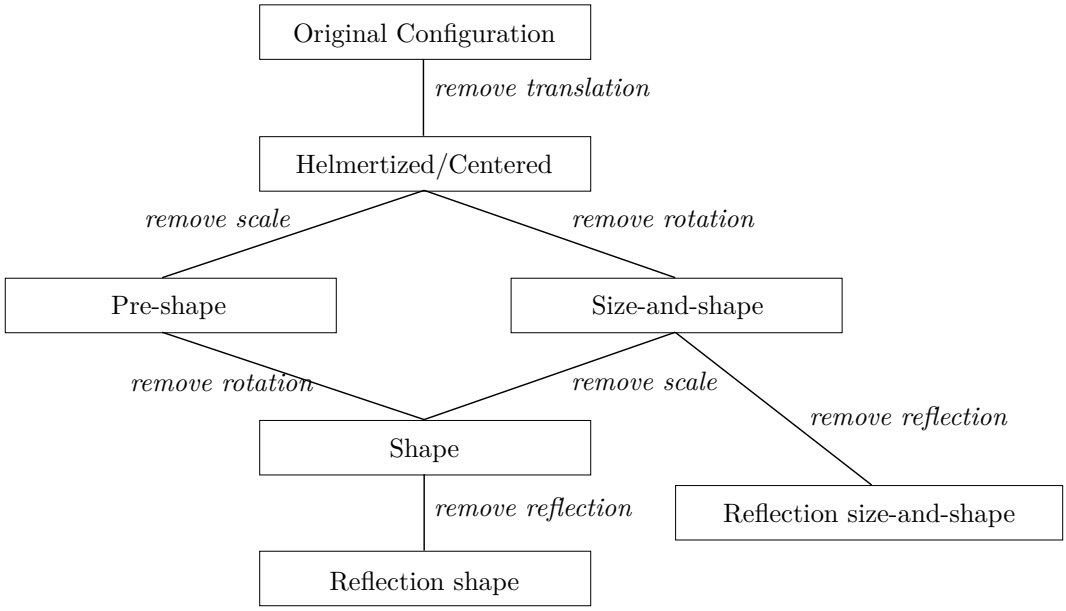


Figure 2.2: *Hierarchical structure of spaces.*

The matrices $\mathbf{X}_c$ and $\mathbf{X}_H$ are *pre-form* of $\mathbf{X}$ and are invariant under the translation of the original configuration. Furthermore they characterize pre-form space (i.e. Helmertized/Centered space) with $dk - d$ dimensions, because d dimensions are lost for location.
Finally it is possible to state that the pre-form of a configuration matrix $\mathbf{X}$ is the equivalence class of $\mathbf{X}$ with respect to the translation in $\mathbb{R}^d$.

2.3.2 Pre-shape

If scale is the second effect to remove, then we move from pre-form to *pre-shape* of a configuration matrix $\mathbf{X}$. Dryden and Mardia (1998) give the following definition of pre-shape

Definition 2.3.2 *The pre-shape of a configuration matrix* $\mathbf{X}$ *is given by*

$$\mathbf{Z} = \frac{\mathbf{X}_H}{\|\mathbf{X}_H\|} \tag{2.21}$$

which is invariant under the translation and scaling of the original configuration.

The pre-shape can be interpreted as the configuration centered and then normalized dividing it by size. In order to normalize a centered matrix $\mathbf{X}_c$ it is necessary to compute its centroid size

$$CS = \|\mathbf{X}_c\| = tr(\mathbf{X}_c\mathbf{X}_c^T) \tag{2.22}$$

Centered configuration $\mathbf{X}_c$ are rescaled to $CS = 1$, yielding centered normed configuration

$$\mathbf{X}_{cn} = \frac{\mathbf{X}_c}{\|\mathbf{X}_c\|} = \frac{\mathbf{CX}}{\|\mathbf{CX}\|} = \mathbf{H}^T\mathbf{Z} \tag{2.23}$$

since $\mathbf{C} = \mathbf{H}^T\mathbf{H}$.
The matrices $\mathbf{Z}$ and $\mathbf{X}_{cn}$ characterize pre-shape space with $dk - d - 1$ dimensions, because d dimensions are lost for location and one for uniform scale.

2.3.3 Shape

In order to filter the rotation effect out it is necessary to identify all its rotated versions of the pre-shape with each other, obtaining the shape of the configuration. A formal definition provided by Dryden and Mardia (1998) is

Definition 2.3.3 *The shape of a configuration matrix* $\mathbf{X}$ *is all the geometrical information about* $\mathbf{X}$ *that is invariant under location, rotation and isotropic scaling (Euclidean similarity transformations). The shape can be represented by the set*

$$\{\mathbf{Z}\boldsymbol{\Gamma} : \boldsymbol{\Gamma} \in SO(d)\} \tag{2.24}$$

where $SO(d)$ *is the special orthogonal group of rotations.*

In that way we removed, step by step, location, scaling and rotation information from a configuration $\mathbf{X}$, keeping just shape information. Shape characterizes a space with $dk - d - 1 - d(d-1)/2$ dimensions, because d dimensions are lost for location, one for uniform scale and $d(d-1)/2$ for rotation.

2.3.4 Size-and-shape

Sometime it happens that we want to keep both size and shape information. Consequently we have to remove just location and rotation information from original configuration. They can be removed simultaneously carrying out a transformation called *rigid-body transformation*, which is defined by Dryden and Mardia (1998) in the following way

Definition 2.3.4 *The rigid-body transformations of a configuration matrix* $\mathbf{X}$ *are the set of translated, rotated* $\mathbf{X}$ *i.e.*

$$\left\{\mathbf{X}\boldsymbol{\Gamma} + \mathbf{1}_k\mathbf{t}^T : \Gamma \in SO(d), \mathbf{t} \in \mathbb{R}^d\right\} \tag{2.25}$$

where $\boldsymbol{\Gamma}$ *is the rotation matrix,* $SO(d)$ *the set of all possible rotations in* d *dimensions and* $\mathbf{t}$ *a translation vector.*

Alternatively it is possible to filter rotation effect out from pre-form of $\mathbf{X}$. In both cases we have *size-and-shape* of $\mathbf{X}$. Formally, it is defined by Dryden and Mardia (1998) in the following way

Definition 2.3.5 *The size-and-shape of a configuration matrix* $\mathbf{X}$ *is all the geometrical information about* $\mathbf{X}$ *that is invariant under location and rotation, and this can be represented by the set*

$$\{\mathbf{X}_H\boldsymbol{\Gamma} : \boldsymbol{\Gamma} \in SO(d)\} \tag{2.26}$$

Size-and-shape is even called the *form* and characterizes a space with $dk - d - d(d-1)/2$ dimensions, because d dimensions are lost for location and $d(d-1)/2$ for rotation.

2.3.5 Reflection

Furthermore it is possible to remove reflection effect for shape or size-and-shape. Dryden and Mardia (1998) give the following definitions of *reflection shape* and *reflection size-and-shape*

Definition 2.3.6 *The reflection shape of a configuration matrix* $\mathbf{X}$ *is all the geometrical information that is invariant under similarity transformations and reflection. The reflection shape can be represented by the set*

$$\{\mathbf{ZR} : \mathbf{R} \in O(d)\} \tag{2.27}$$

where O(d) is the set of $d \times d$ *orthogonal matrices, satisfying* $\mathbf{R}^T\mathbf{R} = \mathbf{I}_d = \mathbf{R}\mathbf{R}^T$ *and* $|R| = \pm 1$*, and* $\mathbf{Z}$ *is the pre-shape.*

Definition 2.3.7 *The reflection size-and-shape of a configuration matrix* $\mathbf{X}$ *is all the geometrical information that is invariant under similarity transformations and reflection. The reflection size-and-shape can be represented by the set*

$$\{\mathbf{X}_H\mathbf{R} : \mathbf{R} \in O(d)\} \tag{2.28}$$

where O(d) is the set of $d \times d$ *orthogonal matrices and* $\mathbf{X}_H$ *are the Helmertized coordinates.*

2.3.6 Procrustes distances

In order to can state how two configurations differ, it is necessary to define the concept of distance between two shapes. Several distances exist and here we have explained three of these: *full Procrustes distance*, *partial Procrustes distance* and *Procrustes distance*.

Let us consider $\mathbf{X}_1$ and $\mathbf{X}_2$ as two configurations with pre-shapes $\mathbf{Z}_1$ and $\mathbf{Z}_2$. The full Procrustes distance between $\mathbf{X}_1$ and $\mathbf{X}_2$ is

$$d_F(\mathbf{X}_1, \mathbf{X}_2) = \inf_{\mathbf{\Gamma} \in SO(d), a \in \mathbb{R}^+} \|\mathbf{Z}_2 - a\mathbf{Z}_1\mathbf{\Gamma}\| = \left\{1 - \left(\sum_{i=1}^{d} \lambda_i\right)^2\right\}^{1/2} \tag{2.29}$$

where $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_{d-1} \geqslant |\lambda_d|$ are the square roots of the eigenvalues of $\mathbf{Z}_1^T\mathbf{Z}_2\mathbf{Z}_2^T\mathbf{Z}_1$ and the smallest value λ_d is the negative square root iff $\det(\mathbf{Z}_1^T\mathbf{Z}_2) < 0$. The rotation matrix that minimizes d_F is given by

$$\hat{\mathbf{\Gamma}} = \mathbf{U}\mathbf{V}^T \tag{2.30}$$

where $\mathbf{U}, \mathbf{V} \in SO(d)$ and $\mathbf{Z}_2^T\mathbf{Z}_1 = \mathbf{V}\mathbf{\Lambda}\mathbf{U}^T$ with $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d)$. The scale that minimizes d_F is

$$\hat{a} = \sum_{i=1}^{d} \lambda_i \tag{2.31}$$

The partial Procrustes distance between the two configurations $\mathbf{X}_1$ and $\mathbf{X}_2$ matches their pre-shapes over rotation as much as possible, without considering scale. Consequently

$$d_P(\mathbf{X}_1, \mathbf{X}_2) = \inf_{\mathbf{\Gamma} \in SO(d)} \|\mathbf{Z}_2 - \mathbf{Z}_1\mathbf{\Gamma}\| = \sqrt{2}\left(1 - \sum_{i=1}^{d} \lambda_i\right)^{1/2} \tag{2.32}$$

Finally the Procrustes distance is the closest great circle distance between $\mathbf{Z}_1$ and $\mathbf{Z}_2$ on the pre-shape sphere

$$\rho(\mathbf{X}_1, \mathbf{X}_2) = 2 \arcsin\left(\frac{d_P(\mathbf{X}_1, \mathbf{X}_2)}{2}\right) = \arccos\left(\sum_{i=1}^{d} \lambda_i\right) \tag{2.33}$$

In Figure 2.3 are shown in the section of the pre-shape sphere the relationships among the Procrustes distance and the full and partial Procrustes distances.

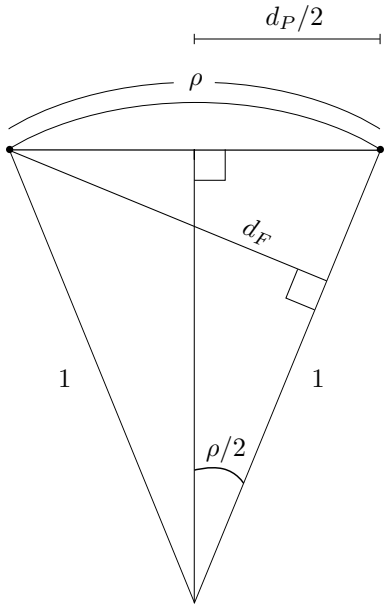


Figure 2.3: *Section of the pre-shape sphere, showing the relationship among the full Procrustes distance d_F, the partial Procrustes distance d_P, and Procrustes distance ρ.*

2.4 General Procrustes Methods

Procrustes Methods are methods based on Procrustes superimposition that analyze landmark data, especially, for estimating an average shape and studying shape variability. A Procrustes analysis is called *full* if all the similarity transformations are carried out, else as *partial*. The term *ordinary* concerns matching of one configuration onto another, whereas if at least two configurations are to be matched is used the term *generalized*.

2.4.1 Full ordinary Procrustes analysis

Full ordinary Procrustes analysis (full OPA) is used for matching two configurations with k landmarks each in d dimensions following rotation, scaling and translation. Estimation of similarity

parameters $\mathbf{t}$, $\mathbf{\Gamma}$ and a is carried out by minimizing the square Euclidean distance between

$$D^2_{OPA}(\mathbf{X}_1, \mathbf{X}_2) = \left\| \mathbf{X}_2 - a\mathbf{X}_1\mathbf{\Gamma} - \mathbf{1}_k\mathbf{t}^T \right\|^2 \tag{2.34}$$

where $a > 0$ is the scale parameter, $\mathbf{\Gamma} \in SO(d)$ is an $(d \times d)$ rotation matrix, $SO(d)$ the set of all possible rotations in d dimensions and $\mathbf{t}$ an $(d \times 1)$ translation vector.
The minimum of $D^2_{OPA}(\mathbf{X}_1, \mathbf{X}_2)$ is indicated as $OSS(\mathbf{X}_1, \mathbf{X}_2)$ which stands for Ordinary (Procrustes Sum of Squares). Assuming $\mathbf{X}_1$ and $\mathbf{X}_2$ are centered to the origin of Cartesian axes, the full ordinary Procrustes solution of $OSS(\mathbf{X}_1, \mathbf{X}_2)$ is given by $\hat{\mathbf{t}}$, $\hat{\mathbf{\Gamma}}$ and $\hat{a}$, where

$$\hat{\mathbf{t}} = 0 \tag{2.35}$$

$$\hat{\mathbf{\Gamma}} = \mathbf{U}\mathbf{V}^T \tag{2.36}$$

where

$$\mathbf{X}_2^T\mathbf{X}_1 = \|\mathbf{X}_1\|\|\mathbf{X}_2\|\mathbf{V}\mathbf{\Lambda}\mathbf{U}^T \tag{2.37}$$

with $\mathbf{U}, \mathbf{V} \in SO(d)$ and $\mathbf{\Lambda}$ a diagonal $d \times d$ matrix of positive elements except possibly its last element. Then

$$\hat{a} = \frac{trace(\mathbf{X}_2^T\mathbf{X}_1\hat{\mathbf{\Gamma}})}{trace(\mathbf{X}_1^T\mathbf{X}_1)} \tag{2.38}$$

and

$$OSS(\mathbf{X}_1, \mathbf{X}_2) = \|\mathbf{X}_2\|^2 \sin^2 \rho(\mathbf{X}_1, \mathbf{X}_2) \tag{2.39}$$

where $\rho(\mathbf{X}_1, \mathbf{X}_2)$ is the Procrustes distance. The full Procrustes fit of $\mathbf{X}_1$ onto $\mathbf{X}_2$ is

$$\mathbf{X}_1^P = \hat{a}\mathbf{X}_1\hat{\mathbf{\Gamma}} + \mathbf{1}_k\hat{\mathbf{t}}^T \tag{2.40}$$

where the superscript P indicates "Procrustes superimposition". The residual matrix after Procrustes matching is defined as

$$\mathbf{R} = \mathbf{X}_2 - \mathbf{X}_1^P \tag{2.41}$$

An important thing is that the ordinary Procrustes fit is not reversible. In other words if the figure sizes are not the same, then $OSS(\mathbf{X}_1, \mathbf{X}_2) \neq OSS(\mathbf{X}_2, \mathbf{X}_1)$. As result $\sqrt{OSS(\mathbf{X}_1, \mathbf{X}_2)}$ cannot be used as a distance measure. To solve that problem it is enough to normalize to unit size in this way

$$OSS\left(\frac{\mathbf{X}_1}{\|\mathbf{X}_1\|}, \frac{\mathbf{X}_2}{\|\mathbf{X}_2\|}\right) = 1 - \left\{\sum_{i=1}^{d} \lambda_i\right\}^2 = \sin^2 \rho(\mathbf{X}_1, \mathbf{X}_2) = d_F^2(\mathbf{X}_1, \mathbf{X}_2) \tag{2.42}$$

2.4.2 Full generalized Procrustes analysis

In the general case where we must matched $n \geqslant 2$ configurations $\mathbf{X}_1, \ldots, \mathbf{X}_n$, Full generalized Procrustes analysis (full GPA) is applied. It concerns to translate, rescale and rotate the configurations relative to each other so as to minimize a quantity proportional to the sum of squared norms of pairwise differences

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{j=i+1}^{n} \left\| \left(a_i\mathbf{X}_i\mathbf{\Gamma}_i + \mathbf{1}_k\mathbf{t}_i^T\right) - \left(a_j\mathbf{X}_j\mathbf{\Gamma}_j + \mathbf{1}_k\mathbf{t}_j^T\right) \right\|^2 \tag{2.43}$$

with the centroid size of the average configuration $S(\overline{\mathbf{X}})$ that must be equal to one, and where $\mathbf{\Gamma}_i \in SO(d)$, $a_i > 0$, $\|\mathbf{X}\| = \sqrt{trace(\mathbf{X}^T\mathbf{X})}$. The average configuration is computed

$$\overline{\mathbf{X}} = \frac{1}{n}\sum_{i=1}^{n}\left(a_i\mathbf{X}_i\mathbf{\Gamma}_i + \mathbf{1}_k\mathbf{t}_i^T\right) \tag{2.44}$$

The generalized (Procrustes) sum of squares, defined by

$$\begin{aligned} G(\mathbf{X}_1,\ldots,\mathbf{X}_n) &= \inf_{a_i,\mathbf{\Gamma}_i,\mathbf{t}_i}\frac{1}{n}\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left\|\left(a_i\mathbf{X}_i\mathbf{\Gamma}_i + \mathbf{1}_k\mathbf{t}_i^T\right) - \left(a_j\mathbf{X}_j\mathbf{\Gamma}_j + \mathbf{1}_k\mathbf{t}_j^T\right)\right\|^2 \\ &= \inf_{a_i,\mathbf{\Gamma}_i,\mathbf{t}_i}\sum_{i=1}^{n}\left\|\left(a_i\mathbf{X}_i\mathbf{\Gamma}_i + \mathbf{1}_k\mathbf{t}_i^T\right)\right\|^2 - \frac{1}{n}\sum_{j=i+1}^{n}\left\|\left(a_j\mathbf{X}_j\mathbf{\Gamma}_j + \mathbf{1}_k\mathbf{t}_j^T\right)\right\|^2 \\ &= \inf_{\mu:S(\mu)=1}\sum_{i=1}^{n}OSS(\mathbf{X}_i,\mu) \\ &= \inf_{\mu:S(\mu)=1}\sum_{i=1}^{n}\sin^2\rho(\mathbf{X}_i,\mu) \end{aligned} \tag{2.45}$$

gives the minimizing parameters $\hat{a}_i$, $\hat{\mathbf{\Gamma}}_i$ and $\hat{\mathbf{t}}_i$, from which the full Procrustes fit of each of $\mathbf{X}_i$ can be obtained

$$\mathbf{X}_i^P = \hat{a}_i\mathbf{X}_i\hat{\mathbf{\Gamma}}_i + \mathbf{1}_k\hat{\mathbf{t}}_i^T, i = 1,\ldots,n \tag{2.46}$$

Differently to OPA, GPA is symmetric for two configurations, even if the figure sizes are not the same.

2.5 Fourier analysis

A very clear explanation of how to perform Fourier analysis in morphometrics is given by Bonhomme et al. (2014), so we decided to get ideas principally from that for exposing it.

Since closed outlines can be considered as periodic functions, then they can be described by Fourier series. Fourier transforms use the Fourier series for decomposing and analyzing periodic signals into a weighted sum of simpler sinusoidal component functions. Actually, Fourier transforms cannot be directly apply for outlines since they are defined as a function of x and y coordinates in two dimensions. At least two possibilities allow to overcome this problem:

- The first option consists in expressing the outline as a function of one transformed variable. In this case we can define outline as the distance of any point on it to the centroid of the shape (i.e. "radius variation"), or as the variation of the tangent angle for any point (i.e. "tangent angle").
- The second possibility, instead, consists in decomposing x an y coordinates and expressing them as functions of the curvilinear abscissa. Here outline is defined as the variation of the (x, y) coordinates on the plane ("elliptical analysis").

In any case, not all points of which outline is made up are always used in outline analysis, since often only a sample of them is considered, the so-called pseudolandmarks. Once all points describing

outline are defined (through x and y coordinates or pseudolandmarks) a Fourier series decomposition method can be applied obtaining an harmonic sum of trigonometric functions weighted with harmonic coefficients. This provides the geometrical information contained in the outline, and can be analyzed with classical multivariate tools.

The general expression of the Fourier expansion for a periodic function $f(t)$ with $t \in \mathbb{R}$ and period T is defined as

$$f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} [a_n \cos(\omega_n t) + b_n \sin(\omega_n t)] \tag{2.47}$$

where $\omega_n = 2\pi n/T$ is the nth harmonic of the function (in radians), and where

$$a_n = \frac{2}{T}\int_{t_1}^{t_2} f(t)\cos(\omega_n t)dt \quad \text{and} \quad b_n = \frac{2}{T}\int_{t_1}^{t_2} f(t)\sin(\omega_n t)dt \tag{2.48}$$

are respectively the even and the odd Fourier coefficients (Claude, 2008).

2.5.1 Fourier radius variation

Zahn and Roskies (1972) developed a method for the analysis and synthesis of closed curves in the plane, therefore applicable to closed outline. This proposed method has two approaches: *radius variation* and *tangent angle*. Here it is explained the first one. Let the radius r equal to the distance from a given point of the outline and its centroid, it can be expressed as a periodic function of the angle θ, with k harmonics that approximate the following function

$$r(\theta) = \frac{1}{2}a_0 + \sum_{n=1}^{k} [a_n \cos(\omega_n \theta) + b_n \sin(\omega_n \theta)] \tag{2.49}$$

with

$$a_n = \frac{2}{p}\sum_{i=1}^{p} r_i \cos(n\theta_i), \quad b_n = \frac{2}{p}\sum_{i=1}^{p} r_i \sin(n\theta_i) \quad \text{and} \quad a_0 = \sqrt{\frac{2}{p}}\sum_{i=1}^{p} r_i \tag{2.50}$$

where p is the number of the points of outline and ω the pulse. Instead, the harmonic coefficients a_n and b_n synthesize outline information and are used as input for multivariate analyses.
This method is not good for complex outline, especially when a given radius intercepts the outline twice, that, is, when the outline presents convexities and concavities.

2.5.2 Fourier tangent angle

The second approach proposed by Zahn and Roskies (1972) is tangent angle. This is very useful for overcoming the problem of outlines with important concavities. It fits the cumulative change in the angle of a tangent vector ($\phi(t)$), as a function of the cumulative curvilinear distance t along the outline. A closed outline scaled to 2π, $\phi(t)$ can be described by $\phi(t) = \theta(t) - \theta(0) - t$, where t is the distance along the outline, $\theta(t)$ the angle of the tangent vector at t and $\theta(0)$ the angle of the tangent vector taken for the first point, which is removed in case of coefficients are normalized. For each harmonic two coefficients are estimated in the following way

$$a_n = \frac{2}{p}\sum_{i=1}^{p} \phi(t)\cos(n\theta_i), \quad b_n = \frac{2}{p}\sum_{i=1}^{p} \phi(t)\sin(n\theta_i) \quad \text{and} \quad a_0 = \sqrt{\frac{2}{p}}\sum_{i=1}^{p} \phi(t) \tag{2.51}$$

2.5.3 Elliptic Fourier analysis

The third method, developed by Giardina and Kuhl (1977) and Kuhl and Giardina (1982), fits the coordinates x and y separately. Even this one it is very useful for fitting curves to complex closed outlines, its two main advantages, respect to other Fourier-based approaches, are shown by Rohlf and Archie (1984) and Crampton (1995). The first one is that outline points can be even not equally spaced, whereas the second one is that it is possible remove size and location effect. Furthermore this approach is very efficient for reducing the number of variables of the original dataset. Let us consider the period of the signal as the perimeter T of the closed outline, $\omega = 2\pi/T$ as the pulse, and t varying from 0 to T as the curvilinear abscissa. Then it is possible to express $x(t)$ and $y(t)$ in the following way

$$\begin{aligned} x(t) &= \frac{a_0}{2} \sum_{n=1}^{+\infty} [a_n \cos(n\omega t) + b_n \sin(n\omega t)] \\ y(t) &= \frac{c_0}{2} \sum_{n=1}^{+\infty} [c_n \cos(n\omega t) + d_n \sin(n\omega t)] \end{aligned} \tag{2.52}$$

with

$$\begin{aligned} a_n &= \frac{2}{T} \int_0^T x(t) \cos(n\omega t) dt \quad \text{and} \quad b_n = \frac{2}{T} \int_0^T x(t) \sin(n\omega t) dt \\ c_n &= \frac{2}{T} \int_0^T y(t) \cos(n\omega t) dt \quad \text{and} \quad d_n = \frac{2}{T} \int_0^T y(t) \sin(n\omega t) dt \end{aligned} \tag{2.53}$$

Given that the outline is made up a finite number of k points, it is possible to calculate discrete estimators for every harmonic coefficient of the nth rank:

$$\begin{aligned} a_n &= \frac{T}{2\pi^2 n^2} \sum_{p=1}^{k} \frac{\Delta x_p}{\Delta t_p} \left(\cos \frac{2\pi n t_p}{T} - \cos \frac{2\pi n t_{p-1}}{T} \right) \\ b_n &= \frac{T}{2\pi^2 n^2} \sum_{p=1}^{k} \frac{\Delta x_p}{\Delta t_p} \left(\sin \frac{2\pi n t_p}{T} - \sin \frac{2\pi n t_{p-1}}{T} \right) \end{aligned} \tag{2.54}$$

$\Delta x_1 = x_1 - x_k$ and c_n and d_n are computed in a similar way. a_0 and c_0, that correspond to the estimate of the coordinates of the centroid of the original outline, are equal to

$$a_0 = \frac{2}{T} \sum_{i=1}^{p} x_i \quad \text{and} \quad c_0 = \frac{2}{T} \sum_{i=1}^{p} y_i \tag{2.55}$$

Through that method four coefficients are got per harmonic, but it is expected that less harmonics are necessary to describe the outline respect to former methods. The first harmonic, which defines the best-fitting ellipse, can be used for normalizing the harmonic coefficients and so that they can be invariant to size, rotation and starting position of the outline trace. This approach is known in literature as the normalized elliptic Fourier (Rohlf and Archie, 1984). It consists in calculating a new set of Fourier coefficients A_n, B_n, C_n, D_n that can be used for multivariate analyses. The equation to get them is

$$\begin{pmatrix} A_n & B_n \\ C_n & D_n \end{pmatrix} = \frac{1}{\lambda} \begin{pmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{pmatrix} \begin{pmatrix} a_n & b_n \\ c_n & d_n \end{pmatrix} \begin{pmatrix} \cos n\theta & -\sin n\theta \\ \sin n\theta & \cos n\theta \end{pmatrix} \tag{2.56}$$

where λ concerns the scale and is the magnitude of the semi-major axis of the ellipse defined by the first harmonic, the second right term is the rotation matrix corresponding to the first ellipse orientation with a ψ angle, the third one is the original harmonic coefficients, and the last one is the rotation of the starting point to the end of the ellipse, with a rotation angle of θ. These parameters can be calculated in that way (Ferson et al., 1985)

$$\lambda = \sqrt{a^{*2} + c^{*2}}, \quad \psi = 0.5 \arctan \frac{2 \times (a_1 b_1 + c_1 d_1)}{a_1^2 + c_1^2 - b_1^2 - d_1^2} \quad \text{and} \quad \theta = \arctan(c^*/a^*) \tag{2.57}$$

with $a^{*2} = a_1 \cos\psi + b_1 \sin\psi$ and $c^{*2} = c_1 \cos\psi + d_1 \sin\psi$. These standardizations are very useful since they allow to carry out comparisons among outlines and analyses of outline shape variation. Furthermore it is possible to apply traditional multivariate statistics to normalized elliptic Fourier coefficients. After standardization the first harmonic has three constant coefficients $A_1 = 1$, $B_1 = C_1 = 0$, whereas the remaining term D_1 is associated with the harmonic eccentricity.

Several approaches to evaluate the number of harmonics exist. Among quantitative ones Crampton (1995) proposed of examining the average deviation from the original outline (that one reconstructs using $n/2$ harmonics) as a function of the number of harmonics used to reconstruct the outline. The examination of the spectrum of harmonic Fourier power is very important for defining the number of needed harmonics. The power, which can be considered as a measure of shape information, is linked to harmonic amplitude by a proportional relation. In fact the first harmonics have a large power, whereas the last harmonics a very small power. The power of a harmonic n is computed through this formula (Claude, 2008)

$$\text{Harmonic Power}_n = \frac{A_n^2 + B_n^2 + C_n^2 + D_n^2}{2} \tag{2.58}$$

In order to choose the number of harmonics needed, it is enough to consider the cumulative sum of the harmonic power, and choose those harmonics that explain a desired amount of shape information. Although most of the shape "information" is contained in the first harmonic, it is possible that it will not be relevant shape information, especially when one wants to investigate differences in complex outlines.

Chapter 3

Classifier combinations

The main aim of combination of expert opinion is to improve the decision preciseness and, consequently, to reduce the probability to commit errors. In the combination of expert opinions it is possible to distinguish between behavioral and mathematical approaches Clemen and Winkler (1999). The behavioral methods consist in discussions among experts, which are combined through a process in which human beings are involved. On the other hand, the mathematical combinations concern the building of models which are combined through mathematical, statistical and logical rules. In the second half of the last century the awareness of importance of combining expert opinions through the development of practical models with a transparent mathematical foundation increased (Cooke, 1991). Later, the attention to this approach raised more and more thanks to the continuos development of computer systems.

Tulyakov et al. (2008) provided a complete definition of the classifier combination and a categorization of different typologies. In the classifier combination the experts are sort of "classifiers" that are combined by a generic "secondary" classifier. It uses as inputs the outputs of the base classifiers, and the combination algorithm is generated by training them. If we denote the score assigned to class i by the base classifier j as s_i^j, then the combination rule is some function f such that

$$S_i = f(\{s_i^j\}_{j=1,\dots,M}) \tag{3.1}$$

The function f is usually simple function such as sum, weighted sum, max et cetera, whereas the "secondary" classifier can assume a particular arg max f form, or it can be a proper classifier. In the matter of different typologies of classifier combination, Tulyakov et al. (2008) grouped them on the basis of some aspects of combined classifiers such as the number, the operating level, the kind of output and the complexity type.

3.1 A tree approach of classifier combination

In a classification problem it is well known that with large number of classes the computational complexity is high. One possible solution could be to split the complex problem of classifying among C classes into $C-1$ sub problems, less complex than the original one, each of them classifying between only two classes. The main idea is to create from data a binary tree of classes with $C-1$

nodes, place a classification rule in each node, and let new samples spread through the tree as models created decide.

3.1.1 Tree building

The binary tree of classes is built adopting an agglomerative (bottom-up) approach. Starting from the situation when each class represents a distinct group, at each step the two classes which are the most similar are concatenated. The classes concatenated at any step are considered as new group of classes (i.e. a superclass) at the next step. The process stops when all classes joint together into one single group.

Let $\mathbf{Z} = (z_{ij})$, $i = 1, \ldots, n$, $j = 1, \ldots, p$ and $z_{ij} \in \mathbb{R}$, denote a matrix containing measurements of p different features of n observations. The p features can be differentiated into several groups according to the methods through which they have been generated. In fact, applying different methods to the same data, they could produce outputs which provide different information. The partition of the features into groups of features is important because those came from different methods could provide complementary information and require to be analyzed separately. Furthermore, some features could be redundant to other ones, thus it is not necessary that both are included in the same analysis. For these reasons, each classification process uses as inputs only features coming from the same group.

The identification of the groups allows us to make order and to organize better the assignation of the right features to each classifier as well as to facilitate their description. Let us define now these groups: In the matrix $\mathbf{Z}$ the columns express the features, and $H = \{\eta_1, \ldots, \eta_p\}$ the set where each element η_j represents the feature of the jth column of the matrix. Consider $\Phi = \{\phi_1, \ldots, \phi_D\}$ with $D \ll p$, as the set of the feature types ϕ_i into which individual features can be gathered. In order to map the features to a specific group, a function $f_1 : H \to \Phi$ is defined. It assigns the feature η_j to a group ϕ_i according to its characteristics. Each group of features can be defined as $T_i = f_1^{-1}(\phi_i) = \{\eta \in H : f_1(\eta) = \phi_i\}$.

Let us show an example of how the p features of $\mathbf{Z}$ can be gathered into D groups ϕ_i (of course, after rearranging the features)

$$\mathbf{Z} = \begin{array}{c} \overbrace{}^{\phi_1}\ \overbrace{}^{\phi_2}\ \cdots\ \overbrace{\phantom{\eta_j\ \eta_{j+1}}}^{\phi_i}\ \cdots\ \overbrace{\phantom{\eta_{p-2}\ \eta_{p-1}\ \eta_p}}^{\phi_D} \\ \begin{array}{ccccccccccccc} \eta_1 & \eta_2 & \eta_3 & \eta_4 & \eta_5 & \cdots & \eta_j & \eta_{j+1} & \cdots & \eta_{p-2} & \eta_{p-1} & \eta_p \end{array} \\ \begin{pmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} & \cdots & z_{1j} & z_{1(j+1)} & \cdots & z_{1(p-2)} & z_{1(p-1)} & z_{1p} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ z_{i1} & z_{i2} & z_{i3} & z_{i4} & z_{i5} & \cdots & z_{ij} & z_{i(j+1)} & \cdots & z_{i(p-2)} & z_{i(p-1)} & z_{ip} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ z_{n1} & z_{n2} & z_{n3} & z_{n4} & z_{n5} & \cdots & z_{nj} & z_{n(j+1)} & \cdots & z_{n(p-2)} & z_{n(p-1)} & z_{np} \end{pmatrix} \end{array} \tag{3.2}$$

Let us now concentrate on the observations. The matrix $\mathbf{Z}$ is made up of n row vectors $\mathbf{z}_i = (z_{i1}, \ldots, z_{ip})$, with $i = 1, \ldots, n$, that contains all information of the p features of the observation i. In that way we can consider $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ as a set of all row vectors. Denote $\mathcal{Z} = \{\mathbf{z} \in \mathbb{R}^p\}$ as the set of $\mathbf{z}_i$, i.e. composed by all possible entities of $\mathbf{z}_i$, such that $\mathbf{Z} \subset \mathcal{Z}$. The observations $\mathbf{z}_i$ can be classified into classes according to their characteristics, so that each class should contain similar observations.

Let $\Omega = \{\omega_1, \ldots, \omega_C\}$ be set of the classes ω_i into which elements $\mathbf{z}_i$ can be classified. In order to classify each observation into a class, we define a function $f_2 : \mathcal{Z} \to \Omega$ as a classification rule that maps each element of $\mathcal{Z}$ to some class ω_i. Thus it is possible to identify a column vector $V = \{f_2(\mathbf{z}_1), \ldots, f_2(\mathbf{z}_n)\} = \{v_1, \ldots, v_n\}$ of assigned classes. The main idea of this approach is to compare classes ω_i two by two in order to identify the pair of most similar classes. To do that we need to generate the $L = \binom{C}{2}$ subsets, each containing the observations connected to two classes. Let the two classes are $\omega_a, \omega_b \in \Omega$. Given the classification rule f_2, then the subset connected to them is $(f_2^{-1}(\omega_a) \cup f_2^{-1}(\omega_b)) \cap \mathbf{Z}$.

Let us generalize the concept: Define $\Psi = \{(\omega_a, \omega_b) : \omega_a, \omega_b \in \Omega, 1 \leq a < b \leq C\} = \{r_1, \ldots, r_L\}$ as the set of all possible combinations of the pairs $(\omega_a, \omega_b) = r_e$, with $e = 1, \ldots, L$, and $\mathbf{U}_{r_e}$ as the subset of $\mathbf{Z}$ containing the observations connected to the pair r_e. The subsets are formally generated as $\mathbf{U}_{r_e} = \{z \in \mathbf{Z} : f_2(z) \in r_e\}$, so that each one is made up of all observations $\mathbf{z}_i \in \mathbf{Z}$ that have the connected classes belonging to a pair r_e. The subsets $\mathbf{U}_{r_e}$ are used to calculate, through several classifiers, how much the two classes of the connected pairs are similar each other. The criteria that defines the measure of similarity between classes is based on a very simple idea: *If a classifier has some problem to discriminate between two classes, that is its accuracy proportion is low, then they can be considered as similar, else as different.* In order to take into account this criteria we need to apply a classification process to each subset $\mathbf{U}_{r_e}$ to find out its goodness rate. To do that it is necessary first to define which classifiers will be used in the processes, then to select the more appropriate features for each classifier and finally to partition the data into a training and a testing set, in order to train the model on one part and test it on unseen data.

Denote $\Theta = \{k_1, \ldots, k_m\}$ the set of m classifiers we decided to apply in our model. We define *the classification process* (k, ϕ) *as the application of a classifier* k *to features belonging to a specific group* ϕ. Because we have m classifiers and D groups of features, it is possible to identify $S = m \times D$ classification processes. Their set is the output of the Cartesian product $\Theta \times \Phi = \{(k, \phi) : k \in \Theta, \phi \in \Phi\} = \{\delta_1, \ldots, \delta_S\}$.

Although all features come from the same data, the methods applied to mine them confer different qualities to features. This is the reason why some features can be considered appropriate to be used as input for a classifier and others instead not. In order to know which features are appropriate for which classifier, we define a set $\Pi = \{\alpha, \beta\}$ where α means "appropriate" and β "not appropriate", and for each classifier $k \in \Theta$ we create a function $f_k : H \to \Pi$, that maps the features to α or β, i.e. it identifies which features can be used as input of the classifier k. Given a selection rule f_k for each classifier, then the subset of features applied in each classification process $\delta = (k, \phi)$ is $f_k^{-1}(\alpha) \cap f_1^{-1}(\phi)$.

Hence we are able to generate from $\mathcal{Z}$ $S \times L$ subsets $\mathbf{U}_{r_e}^{\delta_h} = \{z_{ij} \in \mathcal{Z} : i \in I, j \in J\}$ with I the index set of $\mathbf{z}_i \in \mathbf{U}_{r_e}$ and J the index set of $\eta_j \in (f_k^{-1}(\alpha) \cap f_1^{-1}(\phi))$. In order to be able to use $\mathbf{U}_{r_e}^{\delta_h}$ in the classification process δ_h, it is partitioned into a training set $\mathbf{X}_{r_e}^{\delta_h}$ and a testing set $\mathbf{Y}_{r_e}^{\delta_h}$. The observations are split randomly, assigning a proportion ξ of observations to $\mathbf{X}_{r_e}^{\delta_h}$ and consequentially $1 - \xi$ to $\mathbf{Y}_{r_e}^{\delta_h}$ so that $\mathbf{U}_{r_e}^{\delta_h} = \mathbf{X}_{r_e}^{\delta_h} \cup \mathbf{Y}_{r_e}^{\delta_h}$ and $\mathbf{X}_{r_e}^{\delta_h} \cap \mathbf{Y}_{r_e}^{\delta_h} = \emptyset$.

Once these steps are concluded the classification process δ_h is applied on the training set, and the classification goodness is evaluated through the testing set. Among all indexes for measuring classification goodness, we decided to use the accuracy proportion of the confusion matrix. It consists in *summing all diagonal elements of the confusion matrix stemmed by classification process, and dividing it by the sum of all elements.* In Figure 3.1 we show an example for calculating the accuracy proportion value starting from the confusion matrix stemmed by classification process. We can use the accuracy proportion for calculating the measure of similarity between classes of

Confusion matrix

	ω_a	ω_b
$\hat{\omega}_a$	a_1	a_2
$\hat{\omega}_b$	a_3	a_4

$$\text{Accuracy proportion} = \frac{a_1 + a_4}{a_1 + a_2 + a_3 + a_4}$$

Figure 3.1: *Calculating of accuracy proportion from a confusion matrix.*

r_e. Performing all classification processes we obtain an three-dimensional array $\mathbf{A} = (a_{ijh})$, $i, j = 1, \ldots, C, h = 1, \ldots, S$ and $a_{ijh} \in [0, 1]$ where a_{ijh} represents accuracy proportion calculated between two classes i and j running the classification process h. Let us consider only the two dimensions i and j, and obtain a triangular matrices $\mathbf{A}_h$ with elements equal to zero for all a_{ijh} with $1 \leq i < j \leq C$.

$$\mathbf{A}_h = \begin{pmatrix} 0 & \ldots & \ldots & \ldots & \ldots & 0 \\ a_{21h} & \ddots & & & & \vdots \\ \vdots & \ddots & 0 & & & \vdots \\ a_{i1h} & \ldots & a_{ijh} & \ddots & & \vdots \\ \vdots & \ddots & \vdots & \ddots & \ddots & \vdots \\ a_{C1h} & \ldots & a_{Cjh} & \ldots & a_{C(C-1)h} & 0 \end{pmatrix} \tag{3.3}$$

The measure of similarity between classes i and j could be defined by $S^{-1} \sum_{h=1}^{S} a_{ijh}$, but thus all classifiers, even the worst, would contribute to calculate the measure. In order to avoid of taking into account classifiers that perform not well, a $\lambda \in (0.5, 1)$ is defined as the lowest value of accuracy proportion under which a classifier cannot be considered good and so it is removed. This step is very important because the classifiers selected will be used for building the final tree model. In order for removing the classifiers with a value of accuracy proportion lower than λ, all elements a_{ijh} are multiplied by an indicator function $1_{[\lambda,1]}(a_{ijh})$. Furthermore, to make possible to get the average through the simple sum of the S matrices, each element is divided by $\sum_{h=1}^{S} 1_{[\lambda,1]}(a_{ijh})$ to normalize them. The output of these two operations is the new matrix $\mathbf{A}_h^{\lambda}$

$$\mathbf{A}_h^{\lambda} = \begin{pmatrix} 0 & \ldots & \ldots & \ldots & \ldots & 0 \\ \frac{a_{21h} 1_{[\lambda,1]}(a_{21h})}{\sum_{h=1}^{m} 1_{[\lambda,1]}(a_{21h})} & \ddots & & & & \vdots \\ \vdots & \ddots & 0 & & & \vdots \\ \frac{a_{i1h} 1_{[\lambda,1]}(a_{i1h})}{\sum_{h=1}^{m} 1_{[\lambda,1]}(a_{i1h})} & \ldots & \frac{a_{ijh} 1_{[\lambda,1]}(a_{ijh})}{\sum_{h=1}^{m} 1_{[\lambda,1]}(a_{ijh})} & \ddots & & \vdots \\ \vdots & \ddots & \vdots & \ddots & \ddots & \vdots \\ \frac{a_{C1h} 1_{[\lambda,1]}(a_{C1h})}{\sum_{h=1}^{m} 1_{[\lambda,1]}(a_{C1h})} & \ldots & \frac{a_{Cjh} 1_{[\lambda,1]}(a_{Cjh})}{\sum_{h=1}^{m} 1_{[\lambda,1]}(a_{Cjh})} & \ldots & \frac{a_{C(C-1)h} 1_{[\lambda,1]}(a_{C(C-1)h})}{\sum_{h=1}^{m} 1_{[\lambda,1]}(a_{C(C-1)h})} & 0 \end{pmatrix} \tag{3.4}$$

The averages of the accuracy proportions by pair are obtained by $\sum_{h=1}^{S} \mathbf{A}_h^{\lambda}$, where each element is related to a pair (i, j). Thus in order to know the two classes $r^* = (i, j)$ to group, it is necessary

to solve

$$\underset{r\in\Psi}{\operatorname{argmin}} \left(\sum_{h=1}^{S} \mathbf{A}_h^{\lambda} \right) \tag{3.5}$$

This approach allows to identify the pair $r^* = (i, j)$ containing the elements that have been discriminated worst and so they can be considered as the most similar among all. The process for building the tree is shown in Algorithm 1.

Algorithm 1 Tree building

while $|\Omega| > 2$ **do**
  define $\Psi = \{r_1, \ldots, r_L\}$
  generate $\mathbf{U}_{r_e} = \{z \in \mathbf{Z} : f_2(z) \in r_e\}$
  generate $\mathbf{U}_{r_e}^{\delta_h} = \{z_{ij} \in \mathcal{Z} : i \in I, j \in J\}$
  partition $\mathbf{U}_{r_e}^{\delta_h}$ into $\mathbf{X}_{r_e}^{\delta_h}$ and $\mathbf{Y}_{r_e}^{\delta_h}$
  perform all classification processes and obtain S triangular matrices $\mathbf{A}_h$
  transform $\mathbf{A}_h$ into $\mathbf{A}_h^{\lambda}$
  identify the pair $r^* = \underset{r\in\Psi}{\operatorname{argmin}} \left(\sum_{h=1}^{S} \mathbf{A}_h^{\lambda} \right)$
  remove from Ω the two elements of r^* and add a new single element $\omega = \{x \in r^*\}$ to it
end while

3.1.2 Spreading through the tree

Once the binary tree model is built, it can be used to classify a new sample $\mathbf{q} \in \mathbb{R}^p$ taking advantage of running classification processes with just two classes A and B at a time. The sample $\mathbf{q}$ spreads through the tree until it arrives to a leaf (final node), which corresponds to a specific class. At each node $\mathbf{q}$ is processed by specific classification rules and, according to results it continues the spreading to left or to right branch, that is, it can be classified as belonging to either classes.

Let $A \cup B = \Omega : A \cap B = \emptyset$ and $\Psi = \{(A, B)\}$ as the set gathering all possible classes to which the new sample $\mathbf{q}$ can be classified at each node. Every time $\mathbf{q}$ arrives at one node a classification process is applied defining if $\mathbf{q}$ can be classified as A or B. Denote p_{Ah} as the probability that the sample $\mathbf{q} \in A$ running a classification process δ_h on $\mathbf{X}_{\Psi}^{\delta_h}$. We can indicate $\mathbf{P}$ as the matrix of the probability generated by all classifiers of Θ that $\mathbf{q} \in A$ and $\mathbf{q} \in B$

$$\mathbf{P} = \begin{pmatrix} p_{A1} & \cdots & p_{Ah} & \cdots & p_{AS} \\ p_{B1} & \cdots & p_{Bh} & \cdots & p_{BS} \end{pmatrix} \tag{3.6}$$

In order to decided to which class $\mathbf{q}$ is classified, we can follow two criteria, called respectively the average and the maximum.

The average criteria defines the class of $\mathbf{q}$ by the maximum of the average of the probabilities produced by all classifiers weighted by their accuracy proportions measured during the tree building. Starting from $\mathbf{P}$, both the probabilities of classes A and B are summarized by a weighted average. The vector of weights w is defined in such a way

$$w^T = \begin{pmatrix} (a_{\Psi 1} - \lambda)1_{[\lambda,1]}(a_{\Psi 1}) & \ldots & (a_{\Psi h} - \lambda)1_{[\lambda,1]}(a_{\Psi h}) & \ldots & (a_{\Psi S} - \lambda)1_{[\lambda,1]}(a_{\Psi S}) \end{pmatrix} \tag{3.7}$$

As during binary tree model building, even in this stage we have to remove the classifiers that perform not well. We used the same benchmark so that all elements of w are multiplied by an indicator function $1_{[\lambda,1]}(a_{\Psi h})$. Thus all classifiers with an accuracy proportion lower than λ are not included in the analysis because considered not reliable. Furthermore for giving more weight to the classifiers with an accuracy proportion higher, a quantity of λ is subtracted from each weight $a_{\Psi h}$. Then they will be normalize through their sum that is equal to $\mathbf{1}_m^T w$, where $\mathbf{1}_m = (1, \ldots, 1)^T$ is a column vector of length m. The probabilities of $\mathbf{q}$ to belonging to A or B are equal to $\boldsymbol{\zeta} = \mathbf{P}w/\mathbf{1}_m^T w = (\zeta_A, \zeta_B)^T$.

The maximum criteria, instead, defines the class of $\mathbf{q}$ by the maximum of the probabilities produced by the classifier with the best accuracy proportions measured during the tree building for that specific node. Let us denote a^* as the best accuracy proportions measured during the tree building. The best classifier for a specific node is selected using this vector

$$u^T = \begin{pmatrix} a_{\Psi 1} 1_{[a^*]}(a_{\Psi 1}) & \ldots & a_{\Psi h} 1_{[a^*]}(a_{\Psi h}) & \ldots & a_{\Psi S} 1_{[a^*]}(a_{\Psi S}) \end{pmatrix} \tag{3.8}$$

The probabilities of $\mathbf{q}$ to belonging to A or B are equal to $\boldsymbol{\zeta} = \mathbf{P}u = (\zeta_A, \zeta_B)^T$.

Finally, for both criteria, the class ω^* in which $\mathbf{q}$ has to be classified is defined by

$$\omega^* = \underset{i \in \{A,B\}}{\operatorname{argmax}} \zeta_i \tag{3.9}$$

The process for spreading through the tree is shown in Algorithm 2.

Algorithm 2 Classification

1: **while** $|\Omega| > 1$ **do**
2: define $\Psi = \{(A, B)\}$
3: generate the training set $\mathbf{X}_\Psi^{\delta_h}$
4: apply the S classification processes to $\mathbf{X}_\Psi^{\delta_h}$ to get $\mathbf{P}$
5: calculate $\boldsymbol{\zeta}$ according to the criteria applied
6: identify the class $\omega^* = \underset{i \in \{A,B\}}{\operatorname{argmax}} \zeta_i$
7: classify $\mathbf{q}$ as ω^* and pass to next node
8: **end while**

3.2 Simulations

In order to get easier the understanding of the working principles of the tree approach of classifier combination proposed in this chapter, we carry out an illustrative example explaining step by step all operations needed. We decided to use a dataset available on UCI Machine Learning Repository so as to help to replicate this experiment. Among all datasets we decided to use *Statlog (Image Segmentation) Data Set* (Lichman, 2013) since its default task is classification and the number of classes of response variable is neither too much small or too much large. In fact a small number of classes does not allow to appreciate the working principles of the approach, whereas a large number of classes weighs down the illustrative example.

The dataset has been created by Vision Group (University of Massachusetts) and donated from November 1990. It is an image segmentation database, that is, it contains instance attribute information concerning different objects. The instances have been drawn randomly from a database of 7 outdoor images (brickface, sky, foliage, cement, window, path and grass), and the images have been handsegmented to create a classification for every pixel. Each instance is a 3×3 pixels region. The number of instances is 2310, whereas the attributes are 19 continuous variables. We report in Table 3.1 their description given by the authors.

No.	Label	Description
1	region-centroid-col	the column of the center pixel of the region.
2	region-centroid-row	the row of the center pixel of the region.
3	region-pixel-count	the number of pixels in a region = 9.
4	short-line-density-5	the results of a line extraction algorithm that counts how many lines of length 5 (any orientation) with low contrast, less than or equal to 5, go through the region.
5	short-line-density-2	same as short-line-density-5 but counts lines of high contrast, greater than 5.
6	vedge-mean	measure the contrast of horizontally adjacent pixels in the region. There are 6, the mean and standard deviation are given. This attribute is used as a vertical edge detector.
7	vegde-sd	(see 6)
8	hedge-mean	measures the contrast of vertically adjacent pixels. Used for horizontal line detection.
9	hedge-sd	(see 8).
10	intensity-mean	the average over the region of $(R + G + B)/3$
11	rawred-mean	the average over the region of the R value.
12	rawblue-mean	the average over the region of the B value.
13	rawgreen-mean	the average over the region of the G value.
14	exred-mean	measure the excess red: $(2R - (G + B))$
15	exblue-mean	measure the excess blue: $(2B - (G + R))$
16	exgreen-mean	measure the excess green: $(2G - (R + B))$
17	value-mean	3-d nonlinear transformation of RGB. (Algorithm can be found in Foley et al. (1982))
18	saturatoin-mean	(see 17)
19	hue-mean	(see 17)

Table 3.1: *Description of the 19 attributes information of Statlog (Image Segmentation) Data Set.*

3.2.1 The model structure

In order to carry out the process for building the tree, the first thing to do is to decide which classifiers to use and all parameters we need. To execute this experiment three classifiers are taken into account: Classification And Regression Trees (CART), Support Vector Machines (SVM) and

Naïve Bayes (NB). For simplicity we consider all attributes as a unique feature type. About the parameters, we consider $\lambda = 0.5$ so as to exclude from the analysis just the any classifiers that predict worse than randomly[1], whereas we split the dataset into training and test set assigning to them, respectively, the 80% and the 20% of the instances. Now we can build the tree following the steps indicated in Algorithm 1. The first thing to do is to check if $|\Omega| > 2$, that is, if the number of classes is larger than 2. Since we know that the number of classes is 7, we can start with the first step.

Step 1

Let us define $\Psi = \{r_1, \ldots, r_{21}\}$ (all 21 possible combinations between classes) and create the respective 21 subsets $\mathbf{U}_{r_e}$, each one including all instances with the response variable with a class included in the combination r_e. Since we consider all attributes as of the same type, we do not have to perform more any partition on the data, but we can directly split it into training and test set following the proportion cited above. Now we can apply the three classifiers to training set and define their accuracy using the test set, removing those values smaller than 0.5, i.e. λ (in this case none), and averaging the three values of each pair. The results are shown in Table 3.2. In the first step we group the pair r_{15} creating a superclass "foliage–window", that substitutes for the two classes "foliage" and "window". The last thing to do is to check if now the condition $|\Omega| > 2$ is yet satisfied. Since $|\Omega| = 6$, we can go on with the second step.

Step 2

Let us update Ψ to $\{r_1, \ldots, r_{15}\}$ and create the respective 15 subsets $\mathbf{U}_{r_e}$. Then, as in previous step, we split data into training and test set following the same proportions. Now we reapply the three classifiers to training set and define their accuracy using the test set, removing those values smaller than 0.5, i.e. λ (even in this case none), and averaging the three values of each pair. The results are shown in Table 3.3. In the second step we group the pair r_9 creating a superclass "cement–foliage–window", that substitutes for the two classes "cement" and "foliage–window". The last thing to do is to check if now the condition $|\Omega| > 2$ is yet satisfied. Since $|\Omega| = 5$, we can go on with the third step.

Step 3, Step 4, Step 5 and Step 6

In the next four steps the same process as the two previous ones is repeated. As it is possible to see from Table 3.4, the superclasses created are, respectively, "brickface–cement–foliage–window", "grass–sky", "brickface–cement–foliage–window–grass–sky" and "path–brickface–cement–foliage–window–grass–sky". The final output of the model is the tree structure (Figure 3.2) that illustrates which classes are grouped at each step.

3.2.2 Model accuracy

The tree model that we have just built, allows us to classify new samples. Let us try to use the instances of the testing set to check the accuracy of the tree model. Let us denote $\mathbf{q}_1$ as the first

[1] Since each prediction is made between two classes, if we do not have any information and we decide the prediction randomly, it is like to decide by the flip of a coin, which has, in this case, a probability of right prediction equal to 0.5.

Tree building						
Step 1						
	Class 1	Class 2	CART	NB	SVM	Average
r_1	brickface	cement	1.0000	0.9621	0.8864	0.9495
r_2	brickface	foliage	0.9697	0.9091	0.9015	0.9268
r_3	brickface	grass	1.0000	1.0000	0.8864	0.9621
r_4	brickface	path	1.0000	1.0000	0.8712	0.9571
r_5	brickface	sky	1.0000	1.0000	0.8939	0.9646
r_6	brickface	window	0.9773	0.8712	0.9773	0.9419
r_7	cement	foliage	0.9621	0.9545	0.7879	0.9015
r_8	cement	grass	1.0000	1.0000	0.7197	0.9066
r_9	cement	path	0.9773	0.9394	0.8182	0.9116
r_{10}	cement	sky	1.0000	0.9924	0.7500	0.9141
r_{11}	cement	window	0.9697	0.9091	0.7879	0.8889
r_{12}	foliage	grass	1.0000	1.0000	0.7879	0.9293
r_{13}	foliage	path	1.0000	0.9924	0.8485	0.9470
r_{14}	foliage	sky	1.0000	1.0000	0.7879	0.9293
r_{15}	foliage	window	0.9545	0.5303	0.7803	0.7551
r_{16}	grass	path	1.0000	1.0000	0.8182	0.9394
r_{17}	grass	sky	1.0000	1.0000	0.7576	0.9192
r_{18}	grass	window	1.0000	0.9848	0.7955	0.9268
r_{19}	path	sky	1.0000	1.0000	0.8258	0.9419
r_{20}	path	window	1.0000	0.9924	0.8712	0.9545
r_{21}	sky	window	1.0000	1.0000	0.7955	0.9318

Table 3.2: *The summary results of step 1. The first two columns indicate the class pair. The third, fourth and fifth columns report the accuracy values of the three classifiers obtained on test set for all 21 combinations of class pairs. The last column, instead, is the average of the three accuracy values. In yellow it is highlighted the pair of the two classes to group with the lowest average value.*

instance. We know it belongs to class "foliage", and we want to know to which class our model will classify it. In order to classify that instance, it has to spread through the tree until it arrives to a final node, that is, until it is assigned to a non-superclass. At each node $\mathbf{q}_1$ is processed by all three classification rules we carried out and, according to results it continues the spreading to left or to right branch according to probability to belong to one class or to the other one. Firstly, we have to set the criteria parameter, that is, the rule that combines the different output of the three classifiers. In this case we chose the average criteria. Now we follow the steps indicated in Algorithm 2. The first thing to do is to check if the node of the tree where $\mathbf{q}_1$ is positioned is a final node. Since it is not a final node, we can start with the first step.

Step 1

At the beginning, we define the two classes of the node between whom the model has to classify. At this step they are "path" and "brickface–cement–foliage–window–grass–sky". We train the three classifiers with training set, and use the three models produced to calculate the probability that

Tree building						
Step 2						
	Class 1	Class 2	CART	NB	SVM	Average
r_1	brickface	cement	1.0000	0.9621	0.8864	0.9495
r_2	brickface	grass	1.0000	1.0000	0.8864	0.9621
r_3	brickface	path	1.0000	1.0000	0.8712	0.9571
r_4	brickface	sky	1.0000	1.0000	0.8939	0.9646
r_5	brickface	foliage–window	0.9848	0.7980	0.8384	0.8737
r_6	cement	grass	1.0000	1.0000	0.7197	0.9066
r_7	cement	path	0.9773	0.9394	0.8182	0.9116
r_8	cement	sky	1.0000	0.9924	0.7500	0.9141
r_9	cement	foliage–window	0.9798	0.9091	0.7172	0.8687
r_{10}	grass	path	1.0000	1.0000	0.8182	0.9394
r_{11}	grass	sky	1.0000	1.0000	0.7576	0.9192
r_{12}	grass	foliage–window	1.0000	1.0000	0.7071	0.9024
r_{13}	path	sky	1.0000	1.0000	0.8258	0.9419
r_{14}	path	foliage–window	1.0000	0.9848	0.8384	0.9411
r_{15}	sky	foliage–window	1.0000	1.0000	0.7374	0.9125

Table 3.3: *The summary results of step 2. The first two columns indicate the class pair. The third, fourth and fifth columns report the accuracy values of the three classifiers obtained on test set for all 15 combinations of class pairs. The last column, instead, is the average of the three accuracy values. In yellow it is highlighted the pair of the two classes to group with the lowest average value.*

$\mathbf{q}_1 \in$ "path" and $\mathbf{q}_1 \in$ "brickface–cement–foliage–window–grass–sky", that is, we get the matrix $\mathbf{P}$. Then those probabilities have to be weighted through the respective accuracy proportions obtained during the tree building, and normalized, that is, we have to calculate $\boldsymbol{\zeta}$. Finally, $\mathbf{q}_1$ is assigned to class that maximizes $\boldsymbol{\zeta}$. These operations have to be carry out until $\mathbf{q}_1$ is assigned to a non-superclass.

Table 3.5 shows all the results of the steps carried out until $\mathbf{q}_1$ has been classified to a non-superclass. Firstly we can note that $\mathbf{q}_1$ has been classified correctly as "foliage". Another interesting result is that the probability of the correct class of $\boldsymbol{\zeta}$ decreases at each step. This is an expected result since two classes are grouped if the prediction capability of classifiers is the lowest among all. Consequently, at final steps we find classes very similar each others that create more problems to classifiers, whereas at the beginning steps the classes are very heterogeneous between them so that classifiers have less problem. Furthermore from Step 5 it is possible to note an advantage of the tree model proposed. In spite of NB classified perfectly in the first four steps, in the last one it completely missed the correct class, but the other two classifiers straighten the prediction to the right way. This is an example that allows us to show how the principle of combine more information into a single model can improve the goodness of overall prediction.

If we test all the instances of the test set, we will get an overall accuracy rate equal to 0.9654. In order to calculate how much is the credit for that result of the tree model proposed by us, we compared it with the results obtained using the three classifiers taken individually and reported them in the second column of Table 3.6. The best one is CART since it got the highest accuracy (0.9026), the second one is NB (0.8095) and the last one SVM (0.6299). The differences from the

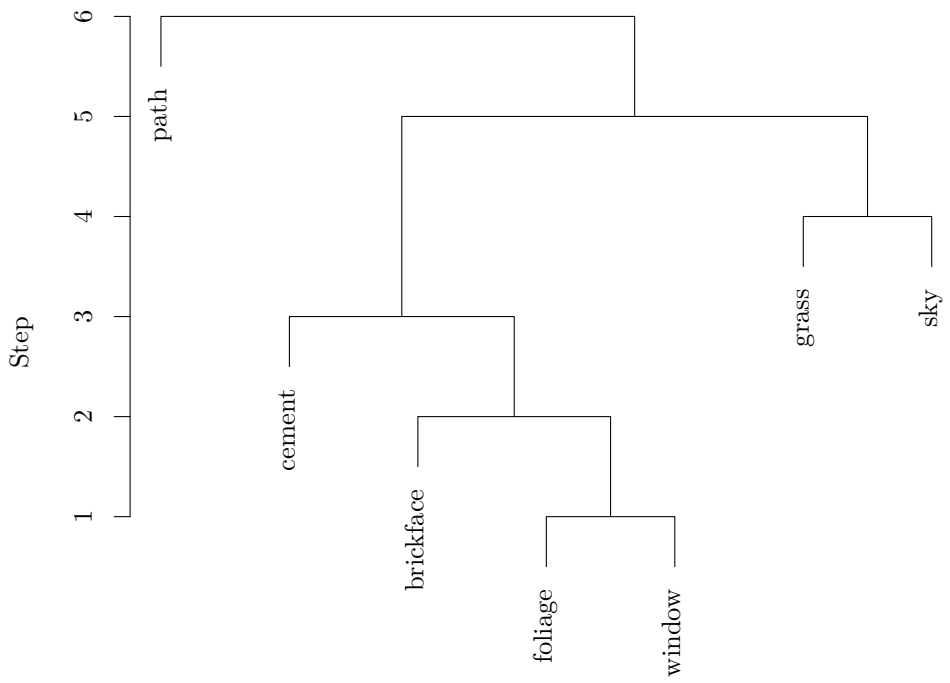


Figure 3.2: *The tree structure of the model.*

results of the classifier taken individually and that of the tree model are 0.0628, 0.1559 and 0.3355, it means we have reduced the error, respectively, of 64%, 82% and 91%. For removing the effect of adding more information into the model by combining different classifiers, and consequently filtering the effect of split the complex problem of classifying among C classes into $C-1$ sub problems, less complex than the original one, each of them classifying between only two classes, we tried to use that tree approach for the three classifiers taken individually, that is, without combining them with other classifiers. The results are reported in the third column of Table 3.6. It is possible to note that both CART and SVM improved their performance, whereas NB worsens it. It means that this approach do not improve automatically performance every time, but only in according to kind of data and type of classifiers. In order to measure the value of combining different classifiers in this tree approach, it is enough to look at the difference between the performances of the tree approach for the three classifiers taken individually and that of the tree approach for the three classifiers combined. The differences are 0.0195 for CART, 0.2057 for NB and 0.2901 for SVM, it means we have reduced the error, respectively, of 36%, 86% and 89%.

Concluding we can state that the tree approach proposed can be a useful tools for enhancing the goodness of prediction, although this is not true for every situation, but according to kind of data and type of classifiers.

Tree building

Step 3

	Class 1	Class 2	CART	NB	SVM	Average
r_1	brickface	grass	1.0000	1.0000	0.8864	0.9621
r_2	brickface	path	1.0000	1.0000	0.8712	0.9571
r_3	brickface	sky	1.0000	1.0000	0.8939	0.9646
r_4	brickface	cement–foliage–window	0.9848	0.7992	0.8561	0.8801
r_5	grass	path	1.0000	1.0000	0.8182	0.9394
r_6	grass	sky	1.0000	1.0000	0.7576	0.9192
r_7	grass	cement–foliage–window	1.0000	1.0000	0.7727	0.9242
r_8	path	sky	1.0000	1.0000	0.8258	0.9419
r_9	path	cement–foliage–window	0.9962	0.9432	0.8561	0.9318
r_{10}	sky	cement–foliage–window	1.0000	0.9924	0.7917	0.9280

Step 4

	Class 1	Class 2	CART	NB	SVM	Average
r_1	grass	path	1.0000	1.0000	0.8182	0.9394
r_2	grass	sky	1.0000	1.0000	0.7576	0.9192
r_3	grass	brickface–cement--foliage–window	1.0000	1.0000	0.8182	0.9394
r_4	path	sky	1.0000	1.0000	0.8258	0.9419
r_5	path	brickface–cement--foliage–window	0.9970	0.9455	0.8818	0.9414
r_6	sky	brickface–cement--foliage–window	1.0000	0.9909	0.8303	0.9404

Step 5

	Class 1	Class 2	CART	NB	SVM	Average
r_1	path	brickface–cement--foliage–window	0.9970	0.9455	0.8818	0.9414
r_2	path	grass–sky	0.9747	1.0000	0.8182	0.9310
r_3	brickface–cement--foliage–window	grass–sky	1.0000	0.8687	0.7222	0.8636

Step 6

	Class 1	Class 2	CART	NB	SVM	Average
r_1	path	brickface–cement–foliage--window–grass–sky	0.9935	0.9740	0.9026	0.9567

Table 3.4: *The summary results of the last four steps of tree building. The first two columns indicate the class pair. The third, fourth and fifth columns report the accuracy values of the three classifiers obtained on test set for the combinations of class pairs. The last column, instead, is the average of the three accuracy values. In yellow it is highlighted the pair of the two classes to group with the lowest average value.*

Model accuracy				
Step 1				
Class	CART	NB	SVM	ζ
path	0.0000	0.0000	0.0000	0.0000
brickface–cement–foliage--window–grass–sky	1.0000	1.0000	1.0000	1.0000
Step 2				
Class	CART	NB	SVM	ζ
brickface–cement–foliage–window	0.9972	1.0000	1.0000	0.9987
grass–sky	0.0028	0.0000	0.0000	0.0013
Step 3				
Class	CART	NB	SVM	ζ
brickface	0.0097	0.0000	0.0000	0.0041
cement–foliage–window	0.9903	1.0000	1.0000	0.9959
Step 4				
Class	CART	NB	SVM	ζ
cement	0.0171	0.0000	0.0000	0.0074
foliage–window	0.9829	1.0000	1.0000	0.9926
Step 5				
Class	CART	NB	SVM	ζ
foliage	1.0000	0.0007	1.0000	0.9604
window	0.0000	0.9993	0.0000	0.0396

Table 3.5: *The summary results of all model accuracy steps. The first column indicates the class pair. The second, third and fourth columns report the probabilities, produced by the three classifiers, that* $\mathbf{q}_1$ *belongs to the two classes. In the last column the weighted and normalized probabilities are reported. In green it is highlighted the class in which* $\mathbf{q}_1$ *has been classified at each step.*

	Individually	Individually tree approach	Combined tree approach
CART	0.9026	0.9459	
NB	0.8095	0.7597	0.9654
SVM	0.6299	0.6753	

Table 3.6: *Results of three classifiers (CART, NB and SVM) carried out in three different ways: individually (second column), using the tree approach but taken individually, that is, without combining them with other classifiers (third column) and combining them using the tree approach.*

Chapter 4

Assessing the reliability of a multi-class classifier

In a classification problem it is common practice testing a wide variety of learning algorithms by varying threshold values and by using different tuning parameters. In that way different classifiers are obtained which can be compared in order to evaluate their predictive ability. Notationally, given a classification problem on L classes observed on n cases, let $\mathbf{Q}$ be a confusion matrix (Table 4.1) resulting from a classifier k. In this framework rows of $\mathbf{Q}$ refer to the true classes, and columns of $\mathbf{Q}$ to the predicted ones. By checking rows, the elements $q_{\ell j}$ indicate how many cases have been classified in each predicted class $\hat{\ell}_j$ $(j = 1, \ldots, L)$. By checking columns, the elements $q_{i\ell}$ indicate how many cases of each predicted class have been classified as ℓ_i $(i = 1, \ldots, L)$.

Starting from the confusion matrix $\mathbf{Q}$ several measures and approaches have been proposed to evaluate classifier performance (accuracy, sensitivity, specificity, speed, cost, readability, etc.). Likewise, the confusion entropy index (Wei et al., 2010), the global performance index (Freitas et al., 2007), the entropy of a confusion matrix (Van Son, 1995), the transmitted information of the classifier (Abramson, 1963) and the relative classifier information (Sindhwani et al., 2001) are all measures that have been defined in order to compare classifiers performance on the basis of the misclassification cells obtained from confusion matrices. Among all these measures, accuracy is the most known. This measure is very plain, overlooking a lot of information about the costs of different elements of misclassification (Hand and Till, 2001).

	$\hat{x}_1$	$\hat{x}_2$	$\cdots$	$\hat{x}_n$
x_1	c_{11}	c_{12}	$\cdots$	c_{1n}
x_2	c_{21}	c_{22}	$\cdots$	c_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
x_n	c_{n1}	c_{n2}	$\cdots$	c_{nn}

Table 4.1: *The confusion matrix* $\mathbf{Q}$.

The goal of this paper is to propose a new approach that enables us to compare performances of several classifiers in the framework of multi-class learning (i.e., when a new observation has to be classified into one, and only one, of L non-overlapping classes). The output is a bivariate classifier performance index obtained from two different measures. The first one refers to a cost-sensitive weighted classification accuracy index. The second one refers to an index measuring the similarity in distribution between the n observations which have been classified in one of the L classes by a classifier and the original distribution of the n cases among the L classes. Both indices are defined in $[0, 1] \in \mathbb{R}$, so that a comparison of different classifier performance can be represented in a $[0, 1]^2$ space. Additionally, introducing a measure which is not one-dimensional allows us to study the reliability of each classifier by re-training the classifier on resampled versions of the original data and computing the convex hull of the area obtained in the 2 dimensions in which values of the bivariate classifier performance index are projected.

4.1 The bivariate classifier performance index

The bivariate classifier performance index derives from a three steps procedure to be carried out for each candidate classifier. The 3 steps can briefly identified with: 1) the model-based measurement of classification accuracy; 2) the measurement of the similarity in distribution between observed classes and predicted ones; 3) the visualization of the results of the previous steps in order to assess global classifier performance.

4.1.1 Model-based measurement of classification accuracy

Let $\pi \in [0, 1]$ be a misclassification level, so that $1 - \pi$ is the classification accuracy level. If K different classifiers are considered, K values of π can be observed and those values, defined in $[0, 1]$, can be modeled on the basis of other information related to each classifier. The model specified for π allows us to assess classifier performance through a model-based classification accuracy index.

In a regression modeling framework characterized by a continuous response variable Y defined in $[0, 1]$, data are usually transformed in order to map the domain of Y in the real line and then a standard linear regression analysis is applied. This approach has some shortcomings (Cribari-Neto and Zeileis, 2009), such as heteroskedasticity and difficulties in the interpretation of estimated parameters, which are expressed in terms of the transformed variable instead of the original one. Ferrari and Cribari-Neto (2004) proposed a regression model for continuous variables that assumes values in $[0, 1]$, called *Beta Regression Model*. The assumption of this model is that the response variable is beta-distributed, $Y \sim Beta(a, b)$ with $a, b > 0$. The authors proposed a particular parameterization of the beta density in order to obtain a regression structure for the mean of the response along with a precision parameter. They showed that, through setting $\mu = a/(a + b)$ and $\phi = a + b$, it is possible to express expectation and variance of Y as $E(Y) = \mu$ and $VAR(Y) = \mu(1 - \mu)/(1 + \phi)$, respectively. The parameter ϕ conveys a rate of precision because for larger ϕ $VAR(Y)$ decreases.

The beta regression model introduced in Ferrari and Cribari-Neto (2004) is applied in the framework of the present study in order to estimate π and, indirectly, $1 - \pi$. Specifically, the goal is to estimate a Beta regression model using a large number of simulated confusion matrices weighted by some proximity measures and misclassification costs, in order to obtain estimated regression parameters and associated π values. Weighting is very important in this framework, because it conveys essential information to the model about the different importance attributed

to possible different misclassification levels. Once the model is estimated, it is applied to the confusion matrix resulting from each classifier in order to estimate a *cost-sensitive (model-based) weighted classification index*. For a classifier k ($k = 1, \ldots, K$) and assuming $\pi_k \sim Beta(\mu_k, \phi)$, the beta regression model is defined as

$$g(\mu_k) = \sum_{i=1}^{L} \sum_{j=1}^{L} \beta_{ij} q_{ij}^k d(\ell_i, \ell_j) = \eta_k \tag{4.1}$$

where $d(\ell_i, \ell_j)$ is a cost-weighted proximity measure as defined in Equation 4.2, q_{ij}^k is the frequency of the cell of the ith row and jth column of the confusion matrix resulting from the classifier k, and β_{ij} is the model coefficient that expresses the contribution of q_{ij}^k to global misclassification of classifier k. Finally, $g(\cdot)$ is a link function. In Equation 4.1 the probit distribution is chosen for specifying the link function $g(\cdot)$, so that the expectation of π_k can be defined as $\mu_k = g^{-1}(\eta_k) = \Phi(\eta_k)$, where $\Phi(\cdot)$ is the cumulative distribution function of a standard normal distribution. As already mentioned, for estimating the β_{ij} in Equation 4.1 a large number B of confusion matrices are simulated. A proportion α with $\pi = 0$ and non-zero elements in the diagonal only, and the other proportion $1-\alpha$ with random assigned elements in order to simulate random classifications, so that $\pi = 1$. A random classified confusion matrix is quite simple to obtain. All confusion matrices stemmed by classifiers have the same marginal row frequencies. In fact, since they come from the same dataset the number of true classes is fixed for all matrices. Hence, it is sufficient to simulate matrices with uniformly distributed rows setting their marginal row frequencies equal to those of the confusion matrices resulting from the classifiers. The next step consists of excluding diagonal cells from simulated matrices, leaving just cells that convey misclassification information. Additionally, the cells of the simulated confusion matrices are weighted by some proximity measures, which are defined, for all entries q_{ij} (with $i \neq j$) corresponding to off-diagonal elements of confusion matrix, as

$$d(\ell_i, \ell_j) = \begin{cases} \dfrac{\ell_L - \ell_1}{|\ell_i - \ell_j|} w_{ij} & \text{if } x \text{ is numerical} \\ \dfrac{L-1}{|i-j|} w_{ij} & \text{if } x \text{ is ordinal} \\ w_{ij} & \text{if } x \text{ is nominal} \end{cases} \tag{4.2}$$

where w_{ij} is a weight, fixed by the researcher, that specifies the importance in terms of misclassification cost attributed to the proximity level between ℓ_i and ℓ_j. As such, weighting is motivated by the idea of adding information deriving from expert knowledge. Once the simulated matrices are weighted, the model could be fitted through them in order to derive the estimated value $\hat{\mu}_k$ of π_k for the kth classifier as

$$\hat{\mu}_k = \Phi\left(\sum_{i=1}^{L} \sum_{j=1}^{L} \hat{\beta}_{ij} q_{ij}^k d\left(\ell_i, \ell_j\right) \right) \tag{4.3}$$

4.1.2 Similarity in distribution index

One of the main problem in the framework of classifier performance measurement is the choice of the best classifier once that two (or more) classifiers present the same value of the classification

accuracy $1 - \pi$ but the latter derives from different confusion matrices. To define a classifier performance measure that also considers information about the difference in distribution among classifier confusion matrices, a *normalized similarity in distribution index* is considered. It derives from a dissimilarity index introduced by Gini and used, among others, in Rachev (1985). In general, for a L-class classification problem D, the Gini index of dissimilarity in distribution, is defined as

$$D = \sqrt{\frac{1}{L^2 - 1} \sum_{h=1}^{L^2-1} |F_h^{v_1} - F_h^{v_2}|^2} \tag{4.4}$$

where $F_h^{v_1}$ and $F_h^{v_2}$ are the cumulative frequencies in h of the vectors v_1 and v_2, whereas $\sqrt{L^2 - 1}$ is equal to the maximum value of this index, and it is used to normalize it. D is defined in $[0, 1]$ and is susceptible to change in values as long as one or more observations are assigned to the class j instead of the true class i ($i \neq j$ and $i, j \in \{1, \ldots, L\}$).

In the framework of the bivariate classifier performance index described so far, the dissimilarity in distribution index introduced in Equation 4.4 is reformulated in terms of a similarity in distribution index. To this aim, let us consider two confusion matrices, $\mathbf{Q}_{k_1}$ and $\mathbf{Q}_{k_2}$, corresponding to classifiers k_1 and k_2 respectively. They refer to a situation in which the value of classification accuracy is the same for both classifiers, even if the two confusion matrices are clearly different. Measuring similarity between $\mathbf{Q}_{k_1}$ and $\mathbf{Q}_{k_2}$ requires the comparison of each element of the two matrices with those of a common reference matrix $\mathbf{Q}_{\max}$. The latter is the matrix which refers to the situation of maximum accuracy so that all its non-zero elements are located in the diagonal, i.e., in the q_{ij} cells ($i = j$), and all predicted values correspond to observed ones. To make such a comparison, the matrices $\mathbf{Q}_{\max}$, $\mathbf{Q}_{k_1}$ and $\mathbf{Q}_{k_2}$ are transformed into vectors $v_{\max}$, v_{k_1} and v_{k_2} by writing the matrix elements in row-major order. To compute the similarity in distribution for $\mathbf{Q}_{k_1}$ and $\mathbf{Q}_{k_2}$, it is necessary to compare the distribution of v_{k_1} and v_{k_2} with that of v_{max}. Considering the difference $1 - D$, where D has been defined in Equation 4.4, for $\mathbf{Q}_{k_1}$ and $\mathbf{Q}_{k_2}$ we define a similarity in distribution index whose values are in $[0, 1]$ as

$$S_{\mathbf{Q}_{k_i}} = 1 - \sqrt{\frac{\sum_{h=1}^{L^2-1} |F_h^{v_{k_i}} - F_h^{v_{\max}}|^2}{L^2 - 1}}, \qquad \forall i = 1, 2 \tag{4.5}$$

4.1.3 Visualization

Once both values of the *cost-sensitive (model-based) weighted classification index* introduced in Section 2.1 and the *normalized similarity in distribution index* introduced in Section 2.2 are available for each classifier, their values can be projected in a $[0, 1]^2$ space in order to evaluate their performance from the perspective of both classification accuracy and similarity in distribution. The possibility of analyzing classifier performance in a two-dimensional space is very useful since it facilitates the comparison among different classifiers and allows the user to understand which of the two considered items (weighted classification and similarity in distribution) mostly influences classifier performance. Of course, the two-dimensional representation is particularly helpful when the number of considered classifiers is very large.

4.1.4 Simulated data example

Hereinafter, results obtained for the two-dimensional classification performance index on simulated data are presented. The simulation setting considers 6 confusion matrices ($\mathbf{Q}_A$ to $\mathbf{Q}_F$) deriving from classifiers A to F (see Table 4.2), with respect to a classification problem involving 4 classes, x_1 to x_4, of an ordinal response variable.

		$\hat{x}_1$	$\hat{x}_2$	$\hat{x}_3$	$\hat{x}_4$
$\mathbf{Q}_A$	x_1	20	4	2	22
	x_2	4	10	1	0
	x_3	0	3	5	0
	x_4	11	7	8	3

		$\hat{x}_1$	$\hat{x}_2$	$\hat{x}_3$	$\hat{x}_4$
$\mathbf{Q}_B$	x_1	28	3	12	5
	x_2	3	9	2	1
	x_3	2	0	4	2
	x_4	4	4	6	15

		$\hat{x}_1$	$\hat{x}_2$	$\hat{x}_3$	$\hat{x}_4$
$\mathbf{Q}_C$	x_1	12	10	12	14
	x_2	5	5	1	4
	x_3	1	2	3	2
	x_4	4	10	8	7

		$\hat{x}_1$	$\hat{x}_2$	$\hat{x}_3$	$\hat{x}_4$
$\mathbf{Q}_D$	x_1	48	0	0	0
	x_2	0	15	0	0
	x_3	0	0	8	0
	x_4	0	0	0	29

		$\hat{x}_1$	$\hat{x}_2$	$\hat{x}_3$	$\hat{x}_4$
$\mathbf{Q}_E$	x_1	4	23	16	5
	x_2	3	6	4	2
	x_3	0	2	2	4
	x_4	1	10	15	3

		$\hat{x}_1$	$\hat{x}_2$	$\hat{x}_3$	$\hat{x}_4$
$\mathbf{Q}_F$	x_1	4	4	16	24
	x_2	1	6	1	7
	x_3	6	0	2	0
	x_4	15	10	1	3

Table 4.2: *Confusion matrices from six different classifiers.*

It is possible to note that: a) the classifier D provides a perfect classification ($\pi_D = 0$); b) the classifiers E and F are characterized by the same accuracy, i.e., they present the same elements on the diagonal of the confusion matrix, but they differ with respect to off-diagonal entries; c) the confusion matrix obtained from C has quite uniformly distributed rows, as it usually happens in random classification, and d) the confusion matrices obtained from A and B refer to a situation which can be considered as intermediate between that concerning C and D.

For each classifier, the proximity measure $d(x_i, x_j)$ introduced in Section 2.1 has been defined according to Equation (4.2). In this example, we fix $w_{ij} = 1$ in order to refer to a situation in which the weight depends proportionally from the distance between observed class and predicted ones. As a result, more weights is attributed to cases which have been classified in class which is far from the original one (in our example x_1 classified as $\hat{x}_4$ or viceversa). Next step is to simulate a large number of weighted confusion matrices. In this example 1,000 matrices are simulated as follows: 500 matrices refer to classifiers with complete accuracy (i.e. the best possible classification), so that they present all non-zero elements on the diagonal and $\pi = 0$; 500 matrices refer to cases deriving from random classified elements (i.e. the worst possible classification), so that they present uniformly distributed row elements and $\pi = 1$.

To apply the beta regression model specified in Equation (4.1) information deriving from these confusion matrices has to be conveniently rearranged. Rearranging each simulated matrix in a row by row manner leads us to the $1,000 \times (4 \times 4)$ matrix represented in Table 4.3. It is possible to note that this matrix has the same row marginal frequencies. This is a common characteristic of all confusion matrices which can be derived from a classifier trained on the same dataset. The computation of a cost-sensitive (model-based) classification accuracy index as defined in Section 2.1 requires the elimination of the diagonal cells from the simulated confusion matrices since only cells that convey misclassification information are included in the beta regression model. Following this elimination, the simulated matrices with off-diagonal elements only are weighted by the proximity measures $d(x_i, x_j)$ in order to obtain the new (weighted) data matrix represented in Table 4.4. It

allows us to put into the beta regression model information about the importance attributed to the possible different misclassifications.

#	p	c_{11}	c_{12}	c_{13}	c_{14}	c_{21}	c_{22}	c_{23}	c_{24}	c_{31}	c_{32}	c_{33}	c_{34}	c_{41}	c_{42}	c_{43}	c_{44}
1	0	48	0	0	0	0	15	0	0	0	0	8	0	0	0	0	29
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
500	0	48	0	0	0	0	15	0	0	0	0	8	0	0	0	0	29
501	1	11	10	13	14	2	9	3	1	1	2	4	1	3	11	3	12
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1000	1	16	9	13	10	1	1	7	6	4	1	3	0	7	10	9	3

Table 4.3: *Simulated confusion matrices with the diagonal cells highlighted.*

#	p	c_{12}	c_{13}	c_{14}	c_{21}	c_{23}	c_{24}	c_{31}	c_{32}	c_{34}	c_{41}	c_{42}	c_{43}
1	0	0	0	0	0	0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
500	0	0	0	0	0	0	0	0	0	0	0	0	0
501	1	30	26	14	6	9	2	2	6	3	3	22	9
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1000	1	27	26	10	3	21	12	8	3	0	7	20	27

Table 4.4: *Simulated confusion matrices after weighting.*

The beta regression model is estimated using the above described weighted simulated matrices. Following the estimation of model parameters, the cost-sensitive (model-based) classification accuracy index is computed for classifiers A to F after the elimination of the diagonal cells from their confusion matrices. The index value is obtained by predicting the response value ($\hat{\pi}_k$, with $k = A, \ldots, F$) on the basis of the classifier confusion matrix entries and the estimated β_{ij}.

As for the computation of the similarity in distribution index S, the six confusion matrices have been disassembled in order to form new variables as described described in Section 2.2. Then, the index S has been computed through the Equation (4.5).

Results obtained for the two indexes are shown in Figures 4.1 and 4.2. The first plot refers to the representation of the two considered indexes in a $[0, 1]^2$ space. The second plot refers to the distribution of the predicted classes and the observed ones, and it shows information about the decomposition of the similarity in distribution index S introduced in Equation (4.5). Figure 4.1 shows that the best classifier is D. This result is not surprising since the considered classifier is the one providing a perfect classification ($\pi_D = 0$). As a consequence, for this classifier the distribution of the predicted classes corresponds to that of the observed ones so that the line in Figure 4.2 obtained for D overlaps the dotted one, which refers to the original distribution of cases among the 16 cells of the confusion matrix. Interesting considerations can be made about the other classifiers. The second best classifier is B, which presents highest values for the two considered indexes after D. This result depends on the fact that misclassified observations are not far from their true classes. Thus, the lower penalization of classifiers presenting extra-diagonal entries which

are close to the diagonal ones is a peculiarity of the proposed bivariate index which correctly uses the higher cost of misclassified observations whose predicted class is far from the observed one. This consideration is enforced by the comparison of the performance of classifiers E and F which, as previously mentioned, have the same elements on the diagonal of the confusion matrix but they differ in the distribution of the extra-diagonal ones. In particular, comparing E and F it is possible to note that in the first case extra-diagonal frequencies are more close to the diagonal ones. The weighting system introduced in Equation (4.5) causes the performance of F to be very poor in comparison to that of E.

	Similarity in distribution index $S_{\mathbf{Q}_k}$	Cost-sensitive (model-based) classification accuracy index $\hat{\pi}_k$
A	0.8588	0.1248
B	0.9175	0.8977
C	0.8599	0.2318
D	1.0000	0.9988
E	0.8524	0.3796
F	0.8030	0.0062

Table 4.5: *Results obtained for the two classifier performance indexes for the classifiers A to F.*

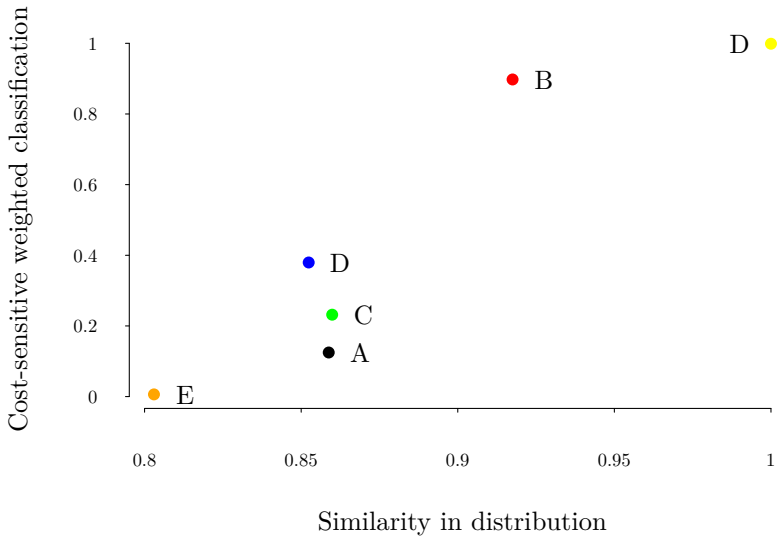


Figure 4.1: *The bivariate cost-sensitive classification index.*

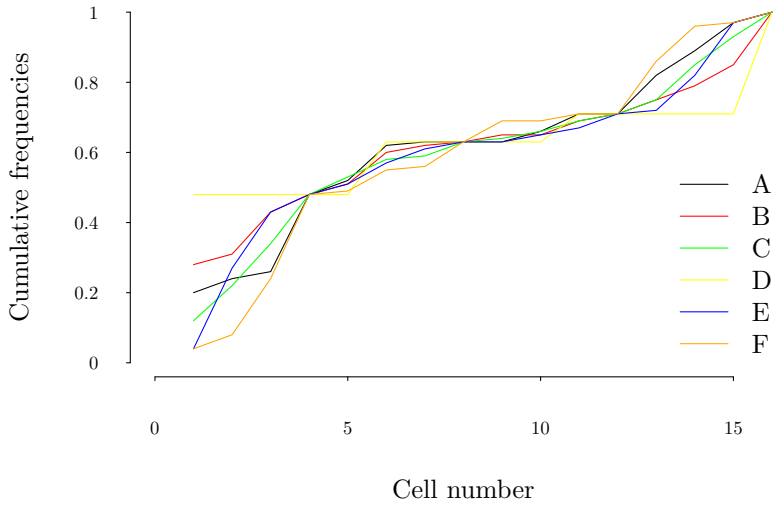


Figure 4.2: *The cumulative distribution function for the observed confusion matrices.*

4.2 Assessing reliability

Besides measuring the performance of a classifier on the basis of classification accuracy and similarity in distribution, it is very important to define its reliability. The *cost-sensitive (model-based) weighted classification index* can be used to accomplish this goal also. In fact, the measurement of the performance of a classifier can be used as a tool in order to define a measure of its reliability. To this purpose, the basic idea is that applying the same classifier to slightly modified versions of the original data, we expect that its results are rather similar, so that the closer they are to each other the more reliable the classifier can be considered. Thus, the proximity of the results obtained from the same classifier by resampling and measured by the bivariate classifier performance index of Section 4.1 is considered as a measure of classifier performance reliability. Formally, this proximity is measured by the convex hull of a set of points $\mathcal{P}$ defined in the Euclidean space obtained with respect to the two dimensions of the bivariate classifier performance index as the area of the convex set that contains $\mathcal{P}$. In order to obtain this measure of reliability three steps are necessary:

1. re-train the classifier B times on resampled versions of the original data;

2. use the resulting B confusion matrices as inputs for the two indices measuring cost-weighted accuracy and similarity in distribution;

3. measure the classifier reliability as the area of the convex hull of the set of points $\mathcal{P}$ defined by the values of two indices obtained over the B runs.

4.2.1 Real data example

In this study a dataset containing 7 variables and $n = 5,712$ cases is considered. The response variable is plant family and has 5 classes (Cyperaceae, Dipsacaceae, Fabaceae, Iridaceae, Lamiaceae). The other six variables are used as predictors and consist in measurements of colorimetric characteristics of seeds. These are the mean of hue, the saturation, the luminance as well as the Red channel, Green channel and Blue channel intensity.
In order to measure classification accuracy and reliability the original data were randomly split into two subsets: a proportion of $0.5 \times n$ defines the training set and the remaining observations the test set. The experiment involves three different classifiers: CART-like recursive partitioning (CART), Random Forests (RF) and Support Vector Machines (SVM). The choice of these classifiers is based on the consideration that CART is notably known as unstable in terms of reliability of the classification outcome whereas the other two methods are presumably more reliable and able to provide more accurate classification. The bivariate classification accuracy index and the classifier reliability measured and visualized through the convex hull are used to verify that the approach presented in sections 2 and 3 provides new insights for the analyzed dataset.

When classifying botany seeds the goal was to measure the performance and reliability of three classifiers using the approach discussed above. It is worth to remember that the *cost-sensitive (model-based) weighted classification index* is made up of two measures: 1) the model-based measurement of classification accuracy; 2) the measurement of the similarity in distribution between observed classes and predicted ones.

To obtain the cost sensitive weighted classification accuracy index as defined in Equation 4.3 it is necessary to define a proximity measure between each pair of classes of the response variable. To this purpose, observations of the training set are standardized and the proximity is measured as the normalized Euclidean distance between the centroids related to pairs of response classes. The obtained proximities for the botanic seeds data are shown in Table 4.6. Furthermore, for

	Cyperaceae	Dipsacaceae	Fabaceae	Iridaceae	Lamiaceae
Cyperaceae	0.0000	0.0272	0.0472	0.0357	0.0454
Dipsacaceae	0.0272	0.0000	0.0246	0.0528	0.0601
Fabaceae	0.0472	0.0246	0.0000	0.0596	0.0810
Iridaceae	0.0357	0.0528	0.0596	0.0000	0.0665
Lamiaceae	0.0454	0.0601	0.0810	0.0665	0.0000

Table 4.6: *Proximity measures between response classes (Training set).*

estimating the coefficients of the Beta regression model introduced in Equation 4.1, $B = 1,000$ confusion matrices were simulated, with a proportion $\alpha = 0.5$ of cases of perfect classification ($\pi = 0$) and the same proportion of cases of random classification ($\pi = 1$). The classifier (CART, SVM or RF) was trained on the training set observations and predicted classes for the test set observations were used to obtain the confusion matrices, which are the input of the Beta regression model specified in Equation 4.1.
As for the measurement of the similarity in distribution between observed classes and predicted ones, the Equation 4.4 was applied to the three confusion matrices obtained by predicting the response classes of the test set observations for the classifiers CART, RF and SVM respectively. The results of the measures are shown in Table 4.7(a).

In order to assess reliability of the three classifiers we used the approach explained in Section 4.2.

Classifier	$(1-\hat{\pi}_k)$	$\hat{S}_Q$
RF	0.842	0.954
SVM	0.811	0.948
CART	0.645	0.942

(a) Accuracy

Classifier	$\frac{1}{100}\sum_{k=1}^{100}(1-\hat{\pi}_k)$	$\frac{1}{100}\sum_{k=1}^{100}\hat{S}_{Q_k}$	$\mathcal{C}$
RF	0.795	0.951	0.171
SVM	0.804	0.948	0.169
CART	0.653	0.942	0.407

(b) Reliability

Table 4.7: *Accuracy and reliability results for the Random Forest, Support Vector Machines and CART-like recursive partitioning classifiers.*

Firstly, we re-trained each classifier on 100 resampled versions of the training set. Next, we used the 100 confusion matrices obtained from each sample as inputs for the two considered accuracy indexes. Finally, we computed the convex hull $\mathcal{C}$ of the area defined by the values of two indexes obtained over the 100 runs as a measure of reliability. The main results are shown in Table 4.7(b). As it is possible to note from both Table 4.7 and Figure 4.3, Random Forest is the best classifier in this example with respect to accuracy. In fact, it has both the highest classification accuracy (0.842) and the highest similarity in distribution (0.954). In contrast, the most reliable classifier is Support Vector Machines as it provides the smallest convex hull area ($\mathcal{C} = 0.169$). As expected, CART has to be considered as the worst one for both accuracy and reliability.

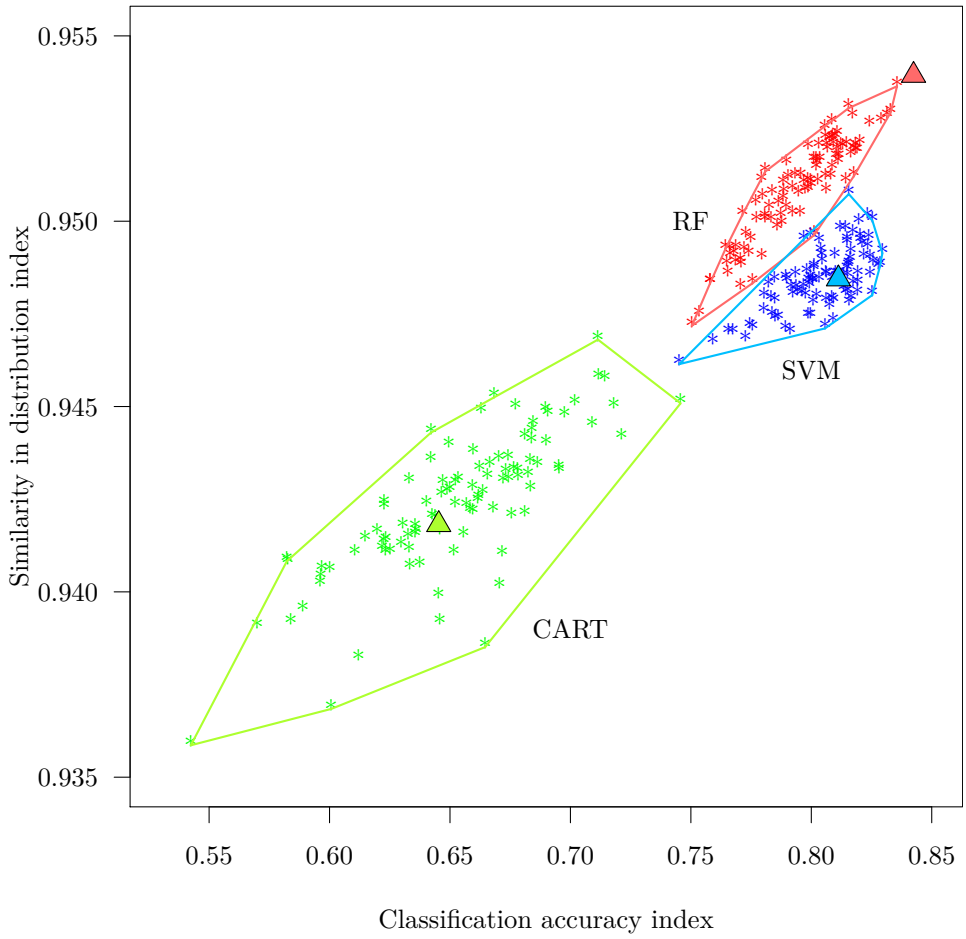


Figure 4.3: *Accuracy and reliability of the Random Forests, Support Vector Machines and CART-like recursive partitioning classifiers. The triangles correspond to the cost-sensitive (model-based) weighted classification index and the similarity in distribution index obtained from the original data, whereas the stars are values of the same indices obtained on resampled versions of the original data. Reliability is measured through the convex hull of the area defined by each set of points.*

Chapter 5

R package

5.1 Why we created an R package

In this dissertation we proposed several original theoretical approaches. The best way to check their reliability is to carry out them with real data and to look at their operation. In order to do that it would have be enough to write some unorganized lines of code in whatever programming language and run them. In spite of it, since reproducibility has always been the most important characteristic of scientific research, we wanted to guarantee possibility that any researcher in all the world could reproduce our analyses in the same way we did. To make it possible we have chosen R (R Core Team, 2015) as programming language because it is the most used language and environment for statistical computing and graphics by statistical researchers in all the world. One of the reasons of its wide spread is its availability as free software, and its simplicity of language that can be easily understood and learnt by everyone.

To be able to carry out our analyses, we wrote several R functions and to organize them in a R package. We decided to create an R package since it is really the easiest way to distribute R code and associated data. Our R package is not already finished, since we must yet test it so as to guarantee its stability, and enhance its computational efficiency. In spite of it, we are confident to release the package in a short time.

In Appendix A it is possible to find all functions developed with their detailed explanations, whereas in Section 6.3.2 a brief tutorial of how to extract information described in Section 6.3.1 using the R functions created.

All the functions present in Appendix A as well as the data, that will be freely available at the depository of Charles University in Prague, can be used by everybody who needed. If you use them in publications please cite this dissertation.

Chapter 6

Application to real data

In this chapter we try to apply all theoretical concepts previously described to real data. The main aims consist of studying how germplasm data respond to morphometrics approaches of classification, comparing the results obtained using algorithms of classification carried out individually and combined as illustrated in Chapter 3, so as to check if their performances of classification are different and, in this case, the importance of this difference.

6.1 State of art

Automatic seed classification is an important research area for two main reasons. The first one is that manual identification of seeds by highly specialized botanists is slow and has a degree of subjectivity hard to quantify. The second one is about commercial interests of agricultural industry, which has to be helped by a reliable and fast tool that permits to classify seeds with a high accuracy rate.

The first attempts to classify seeds by machine vision (Keefe and Draper, 1986; Chen et al., 1989) had deal with few geometrical measurements, such as shape factor, aspect ratio, length and area, and all images used were in grayscale mode. Later Petersen and Krutz (1992) pointed out importance of using color mode images for enhancing classification accuracy. Another important changing is the increasing number of features calculated. It is very important because more features mean more information, even if problem of redundancy is important. Several methods have been applied to remove redundancy among the features, choosing typically those with the largest discriminating power. Granitto et al. (2005) implemented standard sequential forward and backward selection algorithms, i.e. algorithms that build up a subset of features incrementally starting with the empty set (sequential forward selection) or starting with the complete set of features and remove redundant ones at each step (sequential backward selection). They considered as selection criterion the performance of a Naïve Bayes classifier using normal distributions to fit the class-conditional probabilities. Another approach was used by Zapotoczny et al. (2008), who selected the features in function of their variability calculated by Fisher's coefficient, probability of error and average correlation coefficient and mutual information, selecting features with the highest variability for each method.

One of the first methods used for seed classification was Linear Discriminant Analysis (LDA). It is a technic widely adopted in this field for many years, several authors conducted their study using it, such as Chen et al. (1989), Chtioui et al. (1996), Venora et al. (2007), Bacchetta et al. (2008) and Zapotoczny et al. (2008). Among used technics it is possible to come across some Bayesian method such as Naïve Bayes (Granitto et al., 2005), or some more classical one such as Non-Linear Discriminant Analysis (NLDA) and Principal Component Analysis (PCA) (Zapotoczny et al., 2008). Another very recurring method are Artificial Neural Networks (ANNs) (Granitto et al., 2005). Some authors verified the performance of the developed classifiers using the procedure of cross validation (Grillo et al., 2010).

6.2 Data acquisition

At University of Cagliari there exists a center for the biodiversity conservation, called *Centro per la Conservazione della Biodiversità*. In this structure is present the Sardinian Germplasm Bank (BG-SAR) (Mattana et al., 2005) that deals with conservation of germplasm. The germplasm stored in this bank has been gathered respecting internationally recognized protocols (Guarino et al., 1995; Bacchetta et al., 2006) to ensure the greatest possible representativity of the genetic differences of original populations. The seeds stored in the Sardinian Germplasm Bank amount roughly to 71,000 units.

All seeds stored in the bank have been classified by a botanist expert assigning to each one a Family, a Genus, a Species, a Variety and, in same cases, a Sub-variety as well[1]. Because of time and data storage limits data acquisition was obtained by scanning not all seeds gathered, but only a sample of 100 seeds for accession. If the original accession was lower than 100 units, all seeds were scanned. In order to guarantee the representativity of the accession and, at the same time, to minimize intraspecific changes of shape and size of seeds due to their position in fruit and fruit position in plant (Harper et al., 1970), the samples were prepared "randomly"[2].

The images had been taken before the accessions have been put into the dehydration room (15°C at the 15% of R.H.), in order to avoid each possible variation in shape and color. Data acquisition is realized by scanning the accessions in 400 DPI (Dots Per Inch) and in RGB color mode. For adjusting the color, a calibration is obtained using a Kodak Q60 Target Color Chart as reference image. Scanned images have been saved in Tagged Image File Format (TIFF). The choice of the image format is TIFF because in this way no information is lost, as well as it is designed to be independent of any particular hardware or software, and so it is possible to transport images from one application to another or from one computer to another. The drawback of TIFF image is its large size, nevertheless, the advantages overpassed disadvantages.

[1] In botanical classification exists a hierarchical structure for nomenclature of organisms. According to Miller et al. (2011) in botany *"every individual organism is treated as belonging to an indefinite number of taxa of consecutively subordinate rank, among which the rank of species is basic"*. A taxon (plural "taxa") is a group of organisms that share some characteristics, positioned in a rank of the botanical classification. Upper the rank of species is genus, and more upper family. Another important term is accession, it refers to a specific gathering activity and is the basic working unit of conservation in the germplasm bank. It is characterized by several elements such as number of plants wherein seeds are gathered, place, time.

[2] For preparing a random sample of 100 units, the botanist had put all seeds of the accession on a work bench and unsystematically, i.e. without a governing method, selected 100 seeds.

6.3 Data pre-processing

A blind application of data mining on raw data may be detrimental as it could lead to discovering meaningless patterns (Fayyad et al., 1996). To extract useful information therefore it requires referring to data mining within the KDD (Knowledge Discovery in Databases) process, defined as the extraction of useful and not known information from data (Frawley et al., 1992). The other steps in the KDD process, such as data preparation, data selection, data cleaning, and correct interpretation of the results, are mandatory to be able to extract information from data (Fayyad et al., 1996).

6.3.1 From images to data

The starting point is the image digitalized, to be more precise two images for each accession: one with black background and one with white background. As we described in Chapter 1, several methods are included in image pre-processing, such as image resampling, grayscale contrast enhancement, noise removal and manual correction. Among them, in the this application we performed just the grayscale contrast enhancement, since it helps us to get simpler the segmentation process. The others have not been performed. Although manual correction is almost always useful, we decided to do not perform it in this work because both we want to build an approach as automatic as possible removing the subjective human acts, and avoiding time-consuming. Finally we do not perform image resampling because the images resolution was appropriate if compared to seed sizes, and noise removal because not so much noise was present in the images.

The next step is image segmentation, which has the aim to partition an image into different "objects", so in this case to recognize seeds within the accession image. We performed it applying background subtraction approach, which allows us to get a better starting point for image segmentation algorithms (as detailed in Chapter 1), to Sauvola's method. The combination background subtraction approach with Sauvola's method gave us very satisfying results. In fact, since it does not exist a better way for checking the goodness of image segmentation than human eyes, we quickly compared, in that way, several original images with their corresponding binary outputs of the image segmentation, and noticed they resulted very good.

In the next step we enhanced the quality of identified objects. During the thresholding some background pixel could have been categorized as foreground one and vice versa. For this reason it is necessary to apply some correction. The first one is about pixels inside objects set as background. Their assignation is considered as an error, because logically each point enclosed by the perimeter of an object needs to be part of the object itself. Remembering that objects represent seeds, the latter sentence is always true except if seed has holes inside. This situation is impossible because no seed species studied has this characteristic as own, and if some seed was found with some hole, being a damaged seed, it would not be scanned. To solve this problem and fill holes in objects, a first important operation is made: All background pixels enclosed by foreground pixels are set as foreground. The other corrections about object borders are performed by mathematical morphology (see Chapter 1 for details). We let a structuring element with a shape of a "disc" with diameters of 5 pixels and then an operation of opening was applied. It allowed to remove objects that were smaller than the structuring element, whereas large objects remained. In such a way noise is removed and the borders are smoothed.

In the last step, once binary image has been treated, objects are formally detached from it and labeled with correspondent species (each image contains seeds of the same species), data are

extracted from identified objects. Firstly size and texture indexes are computed (see Chapter 1 for details). In order to take into account colors, for each seed it has been computed the empirical distributions of the three color channels (Red, Blue and Green). Finally outline (coordinates that correspond to projection of a border pixel of seed on a Cartesian plane) and landmarks (see Chapter 2 for details) are computed.

Let us consider in detail outline. It is defined here as the closed polygon formed by the (x, y) coordinates of pixels defining it, consequently it conveys a lot of information since considers all shape points. This is a kind of data very common in morphometrics, and several studies have used Fourier series decomposition for fitting a periodic function to describe it. In fact, if you start somewhere on the outline and follow it, you will pass again and again by the same starting point and thus periodic functions can describe this outline. We described three kind of these functions: radius variation, tangent angle and elliptical analysis (see Chapter 2 for details). As a result it is obtained a periodic function, which can be decomposed (and thus described) by Fourier series. Figure 6.1 shows how an outline can be described using the three Fourier-based methods. Although outline is considered as a continuous functions, this statement is true only in theory, inasmuch it is made up of a finite number of discrete points. For that reason a discrete equivalent of Fourier series is used in morphometrics. Among the three methods mentioned above, we decided to transform outline data applying the "elliptical analysis", since it guarantees several advantages such as the possibility to make the coefficients independent of outline position, and the normalization for size. Furthermore this approach is much more efficient than other two for reducing the number of variables of the original dataset. As regards the choice of the number of harmonics can be done following two approaches: a qualitative one and a qualitative one. The first approach consists of visualizing contour reconstructions produced by increasing the number of harmonics involved and comparing these reconstructions with the original one (Figure 6.2). This inspection usually shows that highorder harmonics record high frequency variation that one can assimilate in most cases to "noise variation". In spite of that it is necessary to consider that when differences one wants to investigate concern complex outlines residual variation can be actually useful. The main problem with this approach is that the evaluation is totally subjective. In fact concerning the Figure 6.2 it is very difficult to define which is the best number of harmonics to consider for having a good reconstruction. Furthermore this is a time-consuming activity when, such as in these situations, the number of objects to evaluate is high. On the other hand, the quantitative approach guarantees objectivity and to automate the choice. We have token into account two methods: the average deviation and the harmonic power (see Chapter 2 for details). The first one consists in examining the average deviation from the original outline, i.e. it is calculated for each point of the outline the deviations from original and reconstructed shapes, then the average of these deviations is carried out . The second one, instead, estimates the number of harmonics required for achieving a specific level of "information explained". In this study we have applied the harmonic power method.

Let us consider now in detail landmarks. They are a set of point locations on seed located according to some rule and are used for gathering information over the shape variation. In literature they are often used as starting point for Procrustes methods. In order to make these data usable for further analysis, the first thing to do is to filter the effects we are not interested out. Since we want to get information just about shape, we have to remove those about location, rotation and size performing Full ordinary Procrustes analysis (see Chapter 2 for details). In Figure 6.3 it is shown an example of two configurations of eight landmarks (each one identify by a specific color) before and after Procrustes superimposition. The two configurations have been translated, rotated and scaled until the distances between the correspondent landmarks (landmarks of the same color)

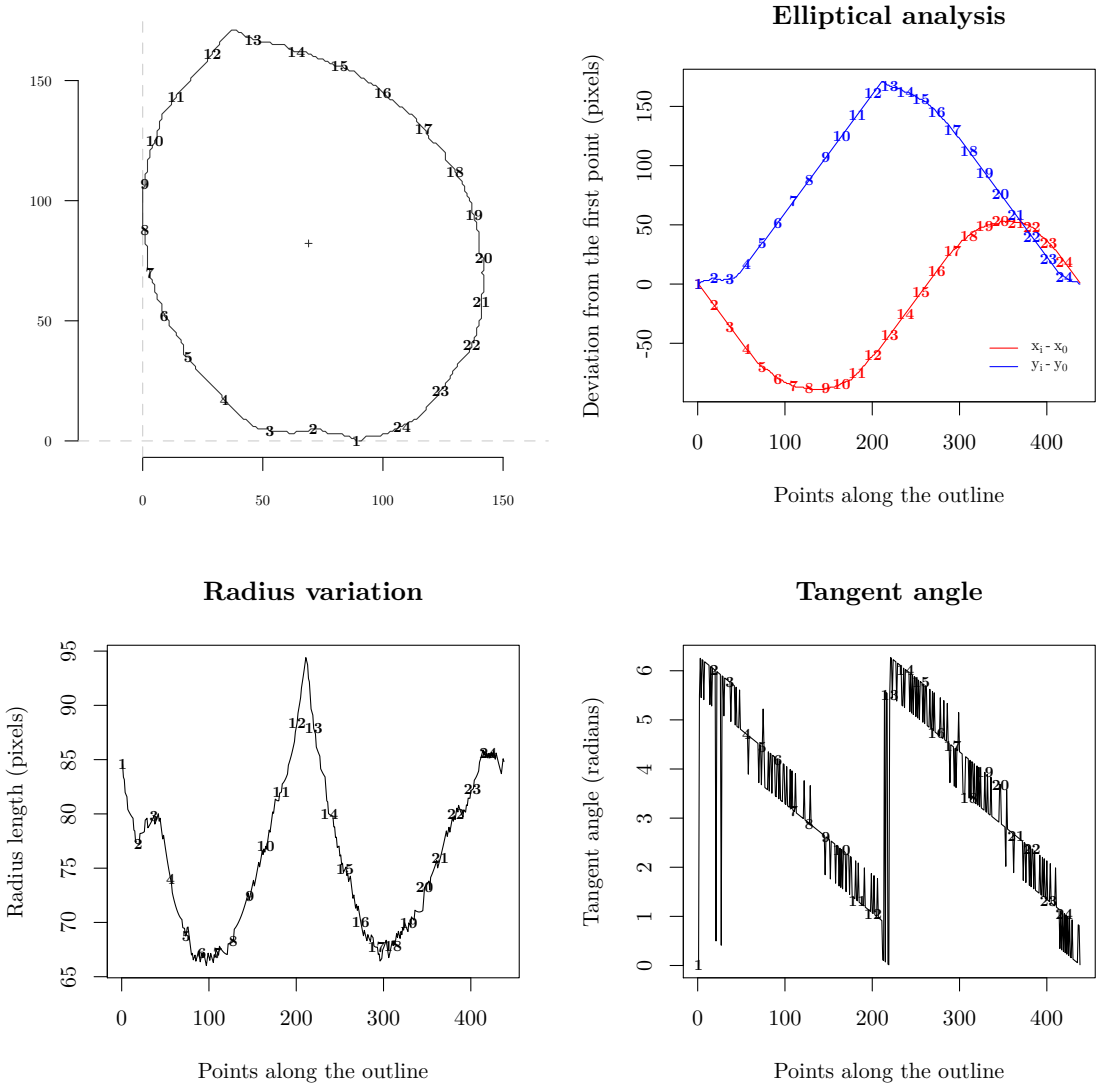


Figure 6.1: *Description of decomposition the outline shown in top-left panel using Fourier-based methods. Elliptical analysis (top-right) shows the two curves corresponding to* $x_i - x_0$ *(in red) and* $y_i - y_0$ *(in blue). Radius variation (bottom-left) illustrates the length of the radius, here considered as the distance between the center of the shape and the points along the outline. Tangent angle (bottom-right) illustrates the variation of the tangent angle along the outline.*

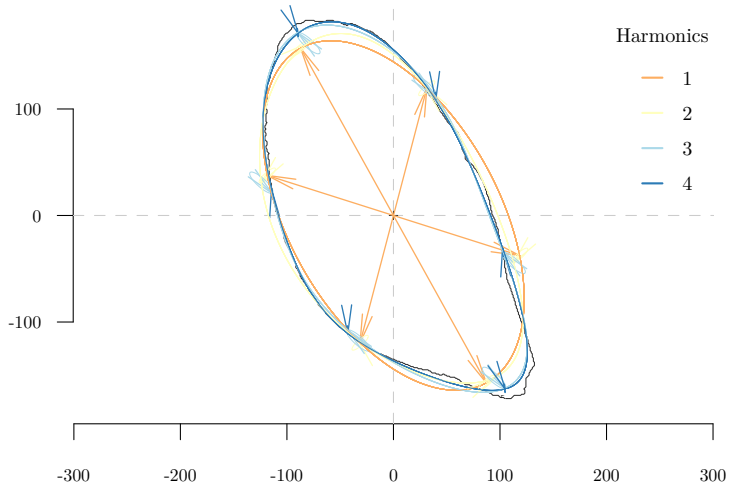


Figure 6.2: *Reconstruction of the shape using a range of harmonic number from 1 to 4. The original shape is defined by the black border. The Fourier decomposition method used is the "elliptical analysis".*

are minimized. In order to be able to use information convey by coordinates as inputs for machine learning algorithms, it is necessary to summarize them through Principal Component Analysis. In this way new variables that express the difference of landmark positions, and so of shape variation, will be created.

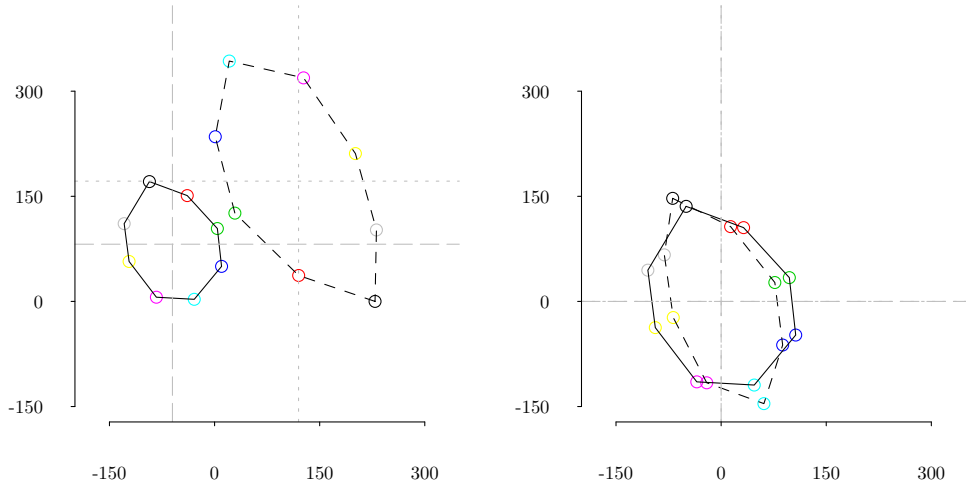
6.3.2 Extracting data from images using R functions

In this section we want to illustrate all steps to follow to extract information described in Section 6.3.1 using the R functions created by ourselves and listed in Appendix A. We explain the process step by step using, for space reasons, as example a sample of an accession concerning the two images shown in Figure 6.4. The first thing to do is to load in R the two images. We used the function `readImage()` of the package `EBImage`, which is a very good R package for image processing and analysis developed by Pau et al. (2010) of which we used even other functions in those created by ourselves.

```
# installation of EBImage package
source("https://bioconductor.org/biocLite.R")
biocLite("EBImage")

library(EBImage)

```



(a) Before Procrustes superimposition (b) After Procrustes superimposition

Figure 6.3: *Two configurations of eight landmarks before and after to be superimposed by Procrustes analysis. The crosses between gray lines indicate the centroids of the shapes. Each landmarks is characterized by a different color. It is easy to see the effects of translation (centroids coincide after superimposition), scaling (sizes are changed) and rotation.*

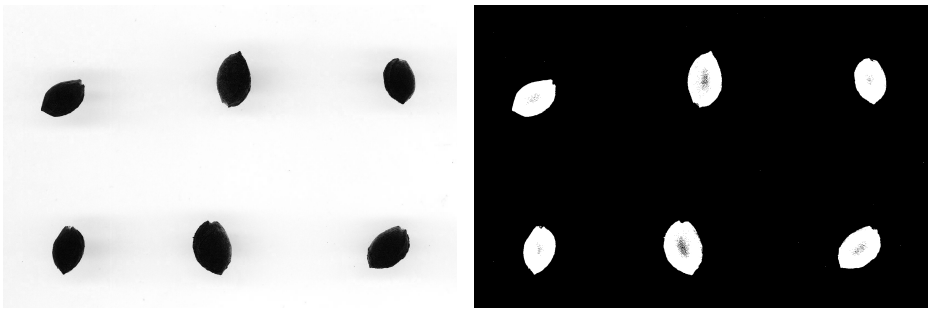
```
7  imageW <- readImage("imageWhite.jpeg")
8  imageB <- readImage("imageBlack.jpeg")
```

Successively we apply the image subtraction approach (see Chapter 1 for details). The first operation is to perform the absolute difference between pixel intensities of the first image to those of the second one. Consequently we get a new image where the values close to zero represent foreground, and those close to one background. Then we transform it into grayscale image enhancing the contrast (Figure 6.5(a)) applying the function `decolorize()`, which has been created by ourselves following the Decolorize algorithm developed by Grundland and Dodgson (2005). The next step consists in defining the threshold values for each pixel in order to be able to separate foreground from background and getting the binary image (Figure 6.5(b)). For this task we created the function `LAT()`, which performs the Sauvola's method applying the algorithm proposed by Shafait et al. (2008) (see Chapter 1 for details).

```
9  subt <- abs(imageW - imageB)
10 imageG <- decolorize(subt)
11 thresholds <- LAT(imageG, w = 100)
12 binary <- ifelse(imageG < thresholds, 1, 0)
```

(a) Black background image (b) White background image

Figure 6.4: *Example of two images corresponding to a sample of an accession used as starting point of data extraction process.*



(a) Decolorized image (b) Binary image

Figure 6.5: *The output of the grayscale contrast enhancement applying the Decolorize algorithm and the binary image applying on it Sauvola's method.*

The binary image generated is not yet ready to be used for defining foreground pixels. As it is possible to see from Figure 6.5(b), Some pixels inside objects set as background, and it has to be considered as an error, because logically each point enclosed by the perimeter of an object needs to be part of the object itself. So we use the `fillHull()` function to fill holes in the objects. Then we define a structuring element with `makeBrush()` function and using it to perform an opening operation with `opening()` function to remove objects that were smaller than the structuring element (i.e. noise) and to smooth the borders. The output is shown in Figure 6.6. The next operation consists in applying the `bwlabel()` function for connecting a label to each object identified as a connected set. All the four functions used in this stage are of the package `EBImage`.

```
13 filled <- fillHull(binary)
14 kern <- makeBrush(size = 5, sigma = 0.3, shape = "disc", angle = 45)
15 op <- opening(filled, kern)
```

```
16 labeled <- bwlabel(op)
```

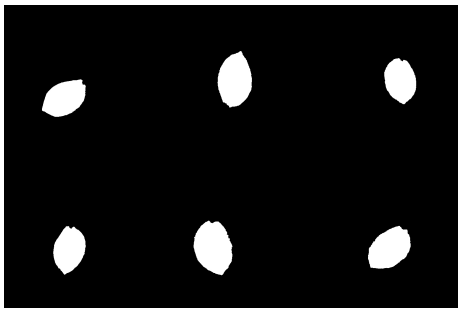


Figure 6.6: *The binary image output of the segmentation process that defines foreground pixels of the original images.*

All the operations shown so far deal with definition of foreground pixels. In order to simplify the work and avoid to write all those lines of code, we created a single function called `binary()`, that gives the same output.

```
17 labeled <- binary(imageW, imageB, w = 100)
```

Once objects are defined, it is necessary to extract information from them. For that task we created a unique function called `extinfo()` that uses as first input the output of `binary()` function. A second input consists in a single image, because Haralick's features and color intensity values have to be computed considering just one image. Since we have two images, the best solution is to average them and using the resulting image as our input. `extinfo()` provides, for each object, a series of information about its shape (outline and landmarks), size, texture and color.

```
18 averageImage <- (imageW + imageB)/2
19 raw_data <- extinfo(labeled$obj, averageImage, nlandmarks = 8, int = 8)
```

Now what we must do is to extract from outline and landmarks data information suitable to further statistical analyses. As explained in Section 6.3.1, it means to compute Fourier coefficients from outline, and superimposed coordinates from landmarks. Since several good packages that perform these operations already exist, we decided to use one of them, more precisely we used `Momocs` (Bonhomme et al., 2014). `Momocs` is S4-oriented (Chambers, 1998), so firstly it is necessary to transform landmarks and outline data, respectively, into `Ldk` and `Out` class objects. Then Elliptical Fourier transform and Full Generalized Procrustes alignment are performed. Now data are ready for

```
# installation of Momocs package
devtools::install_github("vbonhomme/Momocs")

library(Momocs)

outline <- Out(raw_data$outline)
ef <- efourier(outline)

landmarks <- Ldk(raw_data$landmarks$ldks8)
fg <- fgProcrustes(landmarks)
```

6.4 Case study: Germplasm classification analysis of *Rosaceae Prunus*

6.4.1 Data description

For this case study 1,396 seeds have been collected in Sardinia and stored at the BG-SAR. As shown in Table 6.1, the seeds concerning 1 Family, 1 Genus, 3 Species and 23 Varieties. Achieving an elevated rate of good classification can be very difficult since both the number of classes is high and the diversity of shapes is low being all seeds of a same Genus and the majority of the same Species (*Domestica*) as well. The data as resulting from Section 6.3 are the starting point of our analysis. Consequently we have available for the analysis five kinds of data: *size*, *texture*, *outline*, *landmarks* and *color*. Hence we will use italics when we want to refer them as data set and regular as features.

6.4.2 Size data

Let us start to analyze the first kind of data, the size features. They consist of six synthetic indicators (listed in Table 6.2 and described in details in Chapter 1). Since they are expressed in different measure units, in order to make comparison it is necessary to standardize[3] them.

Exploratory analysis

Let us take into account the values taken by all variables in each class. Figures 6.7 and 6.8 illustrate that *MirabolanoGiallo*, *MirabolanoRosso* and *GialloBosa* are very similar, since the four classes have values very lower than average ones for all variables. Another group can be composed by *Coru*, *CorueColumbu* and *LaconiA*, which are characterized by having values of all variables much higher than averages ones. Then we can group *Melone*, *Paradisu*, *LaconiD* and *Stanley* because they assume values very high respect to average for all variables with the exception for *radius.min*,

[3] Standardization is an operation that transform different measure units of variables into a common one. It is carried out to make possible the comparison among variables with different measure units, since it preserves relative distances. In order to carry out the standardization of a variable x, all its values x_i are subtracted by its mean $\bar{x}$ and divided by its standard deviation s_x, i.e. $z_i = (x_i - \bar{x})/s_x$ where z_i corresponds to the standardized value of x_i.

Family	Genus	Species	Variety	No. of seeds
Rosaceae	*Prunus*	*Cerasifera*	*MirabolanoGiallo*	90
			MirabolanoRosso	75
		Domestica	*Cariadoggia*	80
			Cariasina	39
			Coru	55
			CorueColumbu	80
			Croccorighedda	30
			DoreA	32
			Fara	30
			GialloBosa	30
			LaconiA	87
			LaconiD	85
			LaconiE	31
			LaconiF	70
			Melone	77
			NeroSardo	99
			Paradisu	18
			SanGiovanni	39
			SanguignaIBosa	85
			SanguignaIIBosa	60
			Sighera	89
		Salicina	*Shiro*	94
			Stanley	21

Table 6.1: *Distribution of the seeds collected by Family, Genus, Species and Variety.*

which is characterized by a value a little lower the average one. Another group can be made up of *Cariasina*, *LaconiE* and *SanguignaIBosa* that have an almost symmetrical situation respect to latter group, in fact they assume values higher than average for *radius.min* and lower for the other ones. Another group, instead, is composed of *DoreA*, *LaconiF*, *NeroSardo*, *Shiro* and *SanGiovanni* characterized by having all values either little higher or little lower than the average ones. Finally five classes left (*Sighera*, *SanguinosaIIBosa*, *Fara*, *Cariadoggia* and *Croccorighedda*) since they do not assume values similar to other classes. For this reason we decided to leave them alone.

Let us focus on the relationships between variables. From the analysis made above, it is evident as there are several classes where all indicators assume positive values (i.e. values above average) or where all indicators assume negative values (i.e. values below average). This is a first important clue that suggests us about the present of correlation between the variables. In order to have the confirmation of direction and strength of the linear relationship between all pairs of the variables, we carried out a Pearson's correlation index for each pair. Before discussing about the correlation matrix plot in Figure 6.9, it is useful to give some explanation about its reading and interpretation. The correlation matrix plots present in this chapter are made up of two parts: an upper triangle and a lower one. In the lower triangle the values carried out through the Pearson's correlation index for all the possible pairs are reported. In the upper triangle circles for all the pairs are plotted, the bigger size the higher correlation. About the different font color present in both triangles, it

Type	Label	Feature
Size	area	area size
	perimeter	perimeter
	radius.mean	mean radius
	radius.sd	standard deviation of the mean radius
	radius.min	minimum radius
	radius.max	maximum radius
	majoraxis	elliptical fit major axis
	eccentricity	elliptical eccentricity
Texture	asm.s1	Angular Second Moment (mean)
	asm.s2	Angular Second Moment (range)
	con.s1	Contrast (mean)
	con.s2	Contrast (range)
	cor.s1	Correlation (mean)
	cor.s2	Correlation (range)
	var.s1	Variance (mean)
	var.s2	Variance (range)
	idm.s1	Inverse Different Moment (mean)
	idm.s2	Inverse Different Moment (range)
	sav.s1	Sum Average (mean)
	sav.s2	Sum Average (range)
	sva.s1	Sum Variance (mean)
	sva.s2	Sum Variance (range)
	sen.s1	Sum Entropy (mean)
	sen.s2	Sum Entropy (range)
	ent.s1	Entropy (mean)
	ent.s2	Entropy (range)
	dva.s1	Difference Variance (mean)
	dva.s2	Difference Variance (range)
	den.s1	Difference Entropy (mean)
	den.s2	Difference Entropy (range)
	f12.s1	Information Measures of Correlation 1 (mean)
	f12.s2	Information Measures of Correlation 1 (range)
	f13.s1	Information Measures of Correlation 2 (mean)
	f13.s2	Information Measures of Correlation 2 (range)

Table 6.2: *List of size and texture features computed on seeds.*

Figure 6.7: *Distribution of the standardized values taken by the six size variables forf Cariadoggia, Cariasina, Coru, CorueColumbu, Croccorighedda, DoreA, Fara, GialloBosa, LaconiD, LaconiA, LaconiE and LaconiF.*

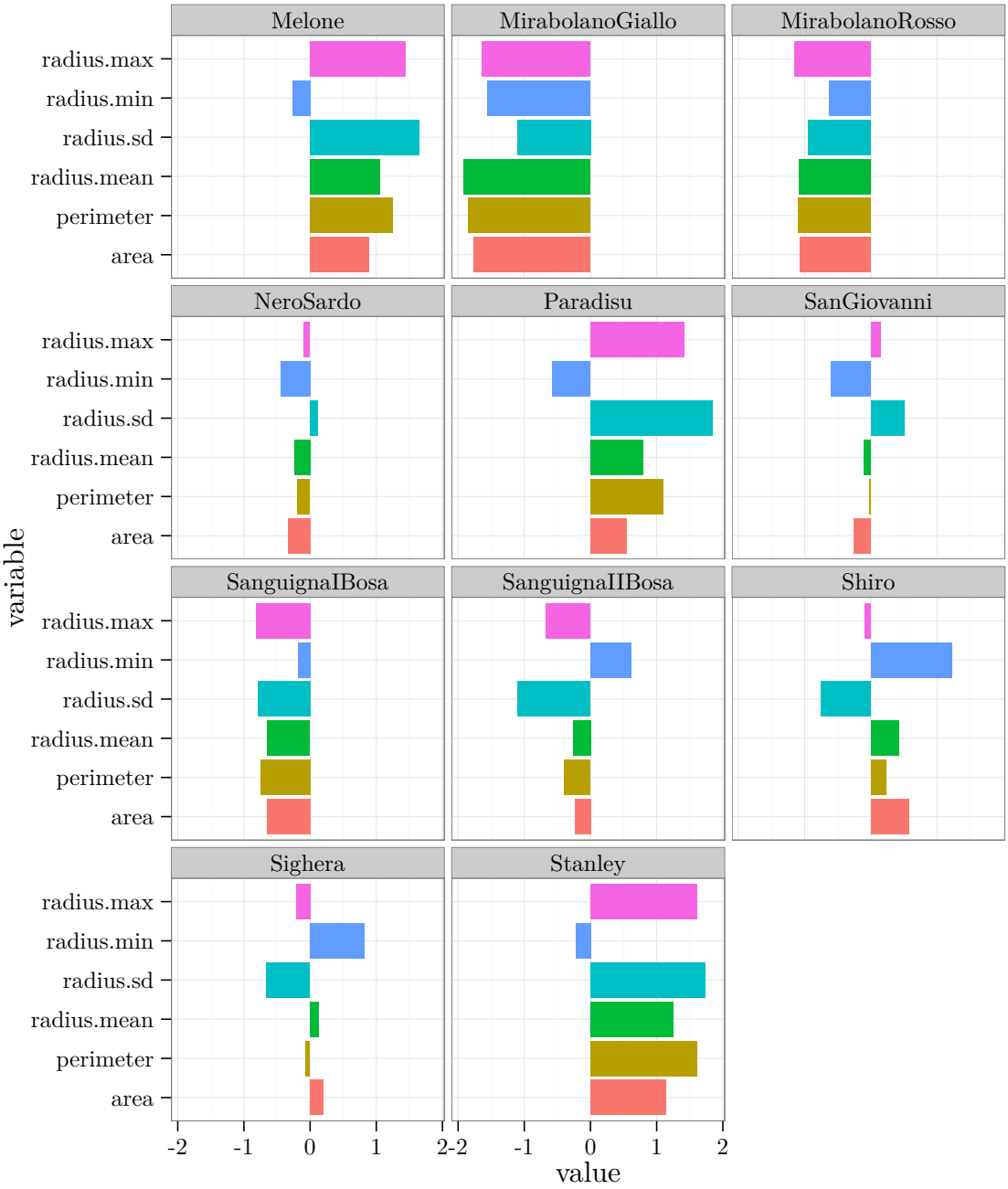


Figure 6.8: *Distribution of the standardized values taken by the six size variables for Melone, MirabolanoGiallo, MirabolanoRosso, NeroSardo, Paradisu, SanGiovanni, SanguinosaIBosa,SanguinosaIIBosa, Shiro, Sighera and Stanley.*

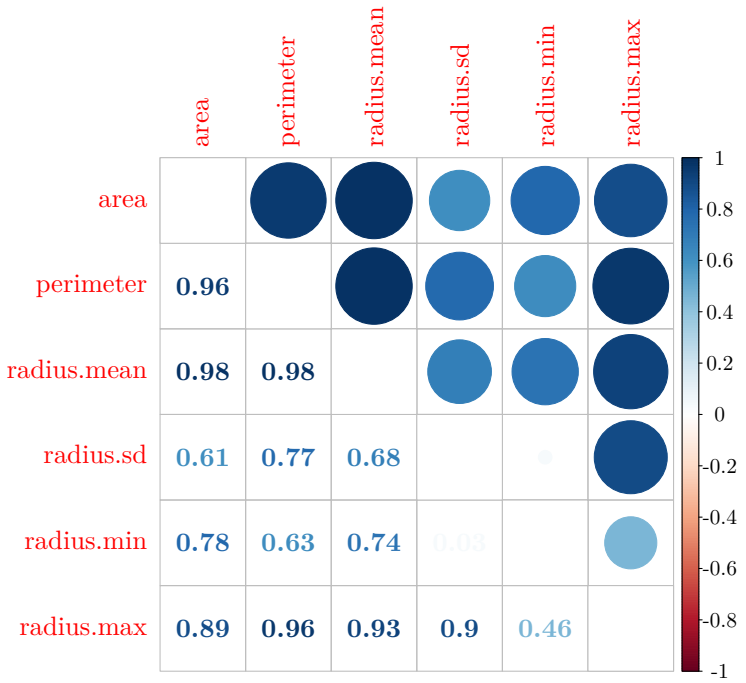


Figure 6.9: *Correlation matrix plot of the six size variables.*

corresponds to values of linear correlations, in other words the darker font the higher value. To identify the two variable labels of a pair is enough to cross horizontally and vertically one cell of the two triangles. Let us analyze now Figure 6.9. As it is shown by the plot, almost all pairs are characterized by a strong linear relationship, in fact only the pair *radius.min-radius.sd* has a value smaller than 0.46. Furthermore all the relationships between variables have a positive direction, in other words the higher value for a variable the higher values for the other ones as well. The intensities of these linear relationships are even quite high, inasmuch 14 coefficients over 15 ($\approx 93\%$) have a value higher than 0.45 and the 40% larger or equal to 0.9. When these latter results are obtained, it means that information conveyed by two variables is practically the same.

Dimensionality reduction

If we wanted to focus on the size aspect we would use as inputs all these variables to take advantage of "noise variation" (i.e. variation not redundant). But since the size is just one of the aspects of the analysis, we want to remove redundant and noise variance. We want to extract only the main information conveyed by them, in order to avoid to use as inputs of classification rules variables characterized by redundancy and noise. To do that we apply to both groups of variables a traditional statistical method, Principal Component Analysis (PCA). It uses an orthogonal transformation to summarize the variation of correlated variables to a set of uncorrelated components (called principal components), each of which is a particular linear combination of the original variables.

Once PCA had been carried out on the six variables, we obtained the results summarized by Table 6.3. As it is possible to note the number of components extracted in a principal component

Principal Component	Eigenvalue	Proportion of Variance	Cumulative Proportion
PC1	4.8683	0.8114	0.8114
PC2	1.0831	0.1805	0.9919
PC3	0.0222	0.0037	0.9956
PC4	0.0187	0.0031	0.9987
PC5	0.0041	0.0007	0.9994
PC6	0.0037	0.0006	1.0000

Table 6.3: *Summary of PCA results obtained from size variables.*

analysis is equal to the number of observed variables being analyzed, in this case six. Speaking about number of extracted components, it exists a trade-off between the amount of explained variance to consider and the need to extract a number factors as less as possible. For excluding principal components some "rules of thumb" exist (Mardia et al., 1980). A first one is to include just enough components that explain a specific threshold of the total variation. A second one is to consider only principal components with eigenvalue (i.e. variance explained) larger than the average, i.e. larger than one because we used correlation matrix. Those principal components explain the variation more than any single input variable and hence can be useful since PCA is to reduce variable space. We decided to apply the first rule, setting the threshold of the total variation to be explained to 99%, consequently we include the first two principal components. The variance explained by the first factor is 4.8683 (81.14% over the total), and so it conveys an amount information roughly as much as five of the original variables. The second factor conveys more information than an original variable

inasmuch it explains a variance equal to 1.0831 (18% over the total). The other four components, instead, convey all together only 0.81% of the all information. In this case if we had decided to adopt the second rule we would have considered the same number of principal components, but not always this happens. Next step consists in trying to interpret the three factors included. In order to do that Table 6.4 contains information very useful, showing the loadings that correspond to coefficients of linear combinations that define principal components. Loadings supply a measure

	PC1	PC2	PC3	PC4	PC5	PC6
area	0.4414	-0.1807	0.7391	0.4660	-0.0327	0.0894
perimeter	0.4502	0.0390	0.1597	-0.7487	-0.3290	0.3186
radius.mean	0.4499	-0.1006	-0.0453	-0.2177	0.2961	-0.8065
radius.sd	0.3429	0.6246	-0.2733	0.3441	-0.5242	-0.1565
radius.min	0.3030	-0.7099	-0.5157	0.2116	-0.2780	0.1274
radius.max	0.4367	0.2483	-0.2925	0.1080	0.6716	0.4465

Table 6.4: *Summary of loadings obtained applying PCA to size variables.*

of contribution of each original variable to principal components. Since all original variables give a roughly the same positive contribution to the first factor, being all their loadings around to 0.40, we can interpret it as a general measure of the size. About the second factor it is possible to note that it is correlated to variables concerning radius, and this relation is negative with *radius.min* and positive with the other two. Consequently it is possible to state that it conveys information over radius length.

In Figure 6.10 it is represented the factorial plane defined by PC1 in x-axis and PC2 in y-axis. In this plane are projected the contour lines based on 2d density kernel divided by class. We did not plot single observation for making clear the plot avoiding the confusion of projecting almost 1,400 points. This plot is very useful to know how the classes are explained by the factors and to study their relationships. As we expected, we find again the same groups identified using summary information of original variables plotted in Figure 6.7 and Figure 6.8. Moreover, it is easy to check if the observations are concentrated or not. For instance we can see that the some classes such as *MirabolanoGiallo* and *Cariadoggia* are very concentrated, whereas others such as *DoreA* and *Stanley* have the observations scattered. It is interesting to note as some classes, for instance *Coru*, *CorueColumbu* and *LaconiA* are overlapping each others, consequently this two factors will not able to distinguish very well among them. Nevertheless, they assume higher values of PC1 than all other classes with the exception of *Paradisu*, *Stanley*, *Melone* and *LaconiD*, but for which they assume lower values of PC2. It means that during classification process PC2 will have an important rule for discriminate them from *Paradisu*, *Stanley*, *Melone* and *LaconiD*, and PC1 for the other classes.

6.4.3 Texture data

Let us consider now the texture indicators. They consist of 26 synthetic indicators (listed in Table 6.2 and described in details in Chapter 1). If the comprehension of the six size indicators has been quite intuitive, inasmuch all the indicators represent physical features whereby everybody had the opportunity to engage with them, the same thing is not true for the texture indicators. In fact the problem "What do the textural features represent?" from a human perception point of view has been considered as subjective even by Haralick, who introduced them for the first time

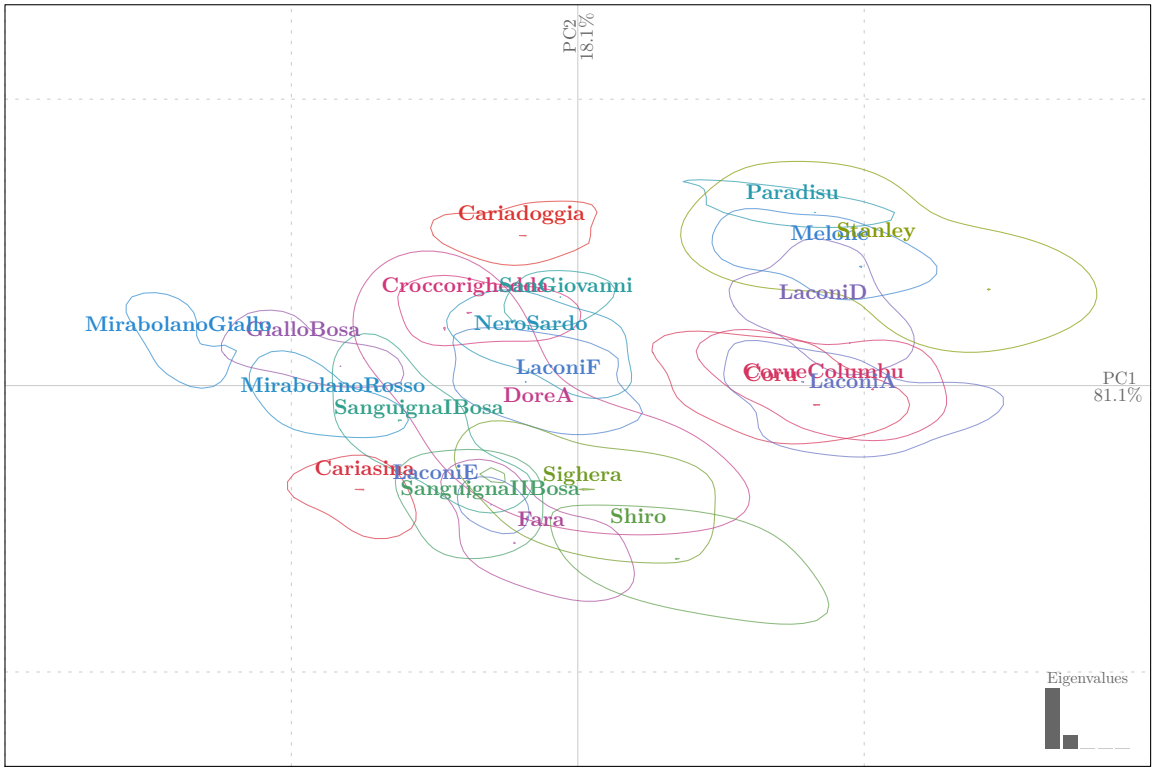


Figure 6.10: *Biplot of the scores (i.e. the new coordinates of the observations in the factorial space) divided by class projected in the plane defined by PC1 (x-axis) and PC2 (y-axis) of size information.*

(Haralick et al., 1973). Definitely some features such as the entropy (i.e. *ent.s1* and *ent.s2*) may be consider as intuitive, in fact one might expect to take higher values for more complex images. But since almost all the others cannot be consider as intuitive, we have decided to skip the univariate exploration of these.

Let us check the correlations between texture features. We did not show the correlation matrix plot because it has been not possible to insert the plot into a page. Nevertheless we report a summary of result about it. We computed the correlation matrix of the 26 texture variables getting 325 coefficients. Essentially all of them (313 over 325, i.e. $\approx 96\%$) have a value statistically significant different from zero, so it is possible to state that a linear relationship, weak or strong, between variables exists. In other words information convey by these variables is very similar, so they are redundant.

Dimensionality reduction

Carrying out PCA on the texture variables, we obtained the results summarized by Table 6.5. In this case for reaching the threshold of 99% we need of the first seven principal components. On the other hand if we took the principal components with eigenvalue larger than the average, we would consider just the first four, loosing 2.86% of overall information. It can seem little, but considering that our final task is to define a rule that classifies among 23 classes, so also that information can be useful. Nevertheless, almost all information is conveyed by the first two components, which summarize an amount of information attributable, respectively, to eleven and nine original variables. Unfortunately in this case the interpretation of the included factors is not as easy as in the size

Principal Component	Eigenvalue	Proportion of Variance	Cumulative Proportion
PC1	11.0953	0.4267	0.4267
PC2	9.5757	0.3683	0.7950
PC3	2.5184	0.0969	0.8919
PC4	1.8973	0.0730	0.9649
PC5	0.3535	0.0136	0.9785
PC6	0.2242	0.0086	0.9871
PC7	0.1666	0.0064	0.9935
PC8	0.0847	0.0033	0.9968
...	...	...	...
PC26	0.0000	0.0000	1.0000

Table 6.5: *Summary of PCA results for texture data.*

contest, since comprehension of texture variables is not so intuitive, and as a result even the same is true for correspondent factors. In any case for providing full disclosure we report also the loadings of the first seven principal components in Table 6.6 and a projection of the observations into two factorial planes defined, respectively, by PC1 in x-axis and PC2 in y-axis in Figure 6.11(a), and by PC3 in x-axis and PC4 in y-axis in Figure 6.11(b). Through the factorial planes it stands out a chaotic situation. In fact several classes have been projected close to the axes origin, and consequently it means those features are not very useful to explain their differences. This argument is not true for same classes such as *Croccorighedda*, *GialloBosa* and *Sighera* in the first plane, and

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
h.asm.s1	-0.1520	0.1250	-0.4578	0.1124	-0.1271	0.3880	-0.0027
h.con.s1	0.2749	0.0961	-0.0679	0.1096	-0.2184	-0.0372	0.2465
h.cor.s1	-0.1063	-0.2692	0.1203	0.1873	-0.3028	0.0523	-0.5017
h.var.s1	0.1121	-0.1874	-0.1770	0.4450	0.4386	-0.0722	-0.0089
h.idm.s1	-0.2756	-0.0970	-0.0922	0.0771	-0.1108	-0.2655	0.0776
h.sav.s1	-0.0954	0.2304	0.3332	0.2306	-0.0386	0.1278	0.1194
h.sva.s1	-0.1038	0.2363	0.2513	0.3093	0.0119	0.1126	0.0356
h.sen.s1	0.1646	-0.2565	0.1193	-0.0686	-0.0044	0.2291	0.2114
h.ent.s1	0.2612	-0.1341	0.1122	-0.0906	0.0297	0.2952	0.1014
h.dva.s1	0.2750	0.0961	-0.0679	0.1096	-0.2182	-0.0372	0.2465
h.den.s1	0.2849	0.0850	0.0416	0.0658	0.0833	0.1022	-0.1200
h.f12.s1	-0.1551	-0.2693	0.0139	0.0827	-0.1137	-0.1002	0.2589
h.f13.s1	-0.0924	-0.2997	0.0727	0.0797	-0.1873	0.0319	-0.0692
h.asm.s2	-0.1407	0.1252	-0.4655	0.0904	-0.0990	0.5330	0.0279
h.con.s2	0.2654	0.0269	-0.1289	0.2342	-0.3796	-0.2041	0.0090
h.cor.s2	-0.1102	-0.2733	0.1214	0.1790	-0.1582	0.1886	-0.3117
h.var.s2	0.1125	-0.1871	-0.1780	0.4438	0.4405	-0.0764	-0.0057
h.idm.s2	-0.2884	-0.0303	-0.0981	0.1037	-0.0679	-0.1702	0.1687
h.sav.s2	-0.0955	0.2299	0.3337	0.2316	-0.0385	0.1293	0.1188
h.sva.s2	-0.1047	0.2361	0.2520	0.3075	0.0139	0.1157	0.0373
h.sen.s2	0.1640	-0.2581	0.1272	-0.0436	0.0012	0.2026	0.1861
h.ent.s2	0.2453	-0.1669	0.1143	-0.0846	-0.0014	0.2307	0.0451
h.dva.s2	0.2655	0.0270	-0.1288	0.2342	-0.3786	-0.2041	0.0090
h.den.s2	0.2919	0.0187	0.0377	0.1139	0.0145	0.0012	-0.2581
h.f12.s2	-0.1590	-0.2613	0.0277	0.0955	-0.0268	0.0124	0.4591
h.f13.s2	-0.1027	-0.2952	0.0800	0.0896	-0.1057	0.1253	0.0880

Table 6.6: *Summary of loadings obtained applying PCA to texture variables. For space reason just loadings of the first seven principle components (those included) are shown.*

such as *DoreA* and *LaconiE* in the second plane, which are clearly separated from the other ones, and so potentially easily discriminable through these texture information.

6.4.4 Outline data

Outline convey a lot of information because they represent seed shapes perfectly. In order to make this information usable for classification analysis, it is necessary we described outlines using the Fourier series decomposition.

Fourier series decomposition

The need of applying a Fourier series decomposition in *outline* field as well as the reasons of choosing "elliptical analysis" as decomposition function have been already explained in Section 6.3. Consequently here we describe how to carry out it. Firstly it is mandatory to estimate the number of necessary harmonics after examining the spectrum of harmonic Fourier power. Precisely the 90%

of harmonic power is provided by the first 4 harmonics, the 95% by the first 5, the 99% by the first 6 and the 99.9% always by the first 6. Since we need a lot of information in order to define a classification rule able to distinguish among seeds of the same Species (i.e. very similar), we decided to take into account the information of the first six harmonic.

In order to compare outlines, it is necessary either to superimpose outlines on their centroid and perform any necessary alignment or to use the normalized Fourier transform (see Chapter 2 for details). We chose the latter option. After normalization, so as to examine variation in the whole sample, we carried out a PCA of the first six harmonics selected on basis of the cumulative total power and on the proportion of explained variation. Principal component analysis and other multivariate approaches can be directly carried out on Fourier decomposition output since all of the harmonic coefficients can be considered as quantitative variables. Since we got four coefficients for each harmonics, in total we have available 24 variables (6 harmonics $\times$ 4 coefficients). Consequentially PCA has been carried out on them, and we obtained the results summarized by Table 6.7. In this case for reaching the threshold of 99% we need of the eleven seven components. It

Principal Component	Variance	Proportion of Variance	Cumulative Proportion
PC1	0.0135	0.8628	0.8628
PC2	0.0006	0.0395	0.9023
PC3	0.0006	0.0354	0.9376
PC4	0.0002	0.0139	0.9515
PC5	0.0002	0.0105	0.9620
PC6	0.0002	0.0098	0.9718
PC7	0.0001	0.0083	0.9801
PC8	0.0001	0.0041	0.9842
PC9	0.0000	0.0031	0.9873
PC10	0.0000	0.0026	0.9899
PC11	0.0000	0.0023	0.9923
PC12	0.0000	0.0013	0.9936
...	...	...	...
PC24	0.0000	0.0000	1.0000

Table 6.7: *Summary of PCA results for Fourier coefficients describing outline data.*

is interesting to note that from the table it stands out that the variance is lower than 1 even for the first principal component. This is due to normalization process applied to data, in fact after it the first harmonic has three constant coefficients, whereas the remaining term is associated with the harmonic eccentricity. Consequently in this situation the rule of extracting eigenvalue higher than 1 does not have any sense. Unfortunately here, as for texture field, component interpretation is not possible because comprehension of harmonic coefficients is quite difficult, especially when they are 24. In Figure 6.12 the contour lines based on 2d density kernel representing class distributions are projected into the factorial plane defined by PC1 in x-axis and PC2 in y-axis. As it was expected almost all variability is in the first axis, in fact all classes are distributed practically just on it, inasmuch it explains more than 86% of information versus less than 4% of the second factor. Furthermore since normalization is not able to define the "right" direction of ellipse major axis, if shapes are prone to bad alignments due to they are either roughly circular or with a strong

bilateral symmetry, as in this situation, some "problem" can occur. In fact it is easy to note from Figure 6.12 that the second factor conveys information over different directions of ellipse major axis. Unfortunately because of individual outlines are defined by a different number of point each others, it is not possible apply methods such as Full Generalized Procrustes alignment, which would be able to solve the problem. Then why do not we remove that factor? Because it does not convey just direction of ellipse major axis but even other information. This is clear if we focus on the top right shape and the bottom right one, in fact if the information explained were just direction of ellipse major axis, then the two shapes would be the same turned upside down. But if we look at it carefully we note some differences exist between them, and the same argument is for the others as well. As a result we decided to keep all the first eleven factors, being aware that they can contain some inevitable noise.

6.4.5 Landmarks data

Landmarks of a seed consist in a matrix of coordinates that summarize its shape. The number of landmarks depends on the presence of specimens on the object and sometimes it is practically obligated since an object can have a specific number of specimens to be taken into account. In other case, instead, the object can have only few relevant specimens and so it is necessary to use other rules for their positioning. Since we want to gather only information about shape, we located them just into outline equally spaced and as starting point we took the point of the maximum diameter, that is farther from the centroid. In these cases, a large number of landmarks is usually set so as to approximate as well as possible the shape. Nevertheless we decided to set just eight landmark, because we wanted to avoid to approximate too much well the outline. In fact one of the reasons why we decided to include *landmarks* in our analysis is for avoiding the overfitting of shape. We wanted to get a different kind of data respect to those concerning outline analysis, and try to compare them in order to know which is the most efficient for a classification task.

Procrustes analysis

Another reason why we wanted to include *landmarks* in addition to *outline*, is because they can be used in Procrustes methods on the contrary of *outline*, since it is necessary that all objects are expressed with the same number of coordinates. As already explained in Section 6.3 we have to perform first Procrustes analysis and then Principal Component Analysis in order to make these data usable for further analysis. Among Procrustes analysis methods we perform Full generalized Procrustes analysis because we needed to match more than two of configurations with a unit size equal to one, inasmuch we are not interested in studying size and shape together, but just shape.

Exploratory analysis

Once Full generalized Procrustes analysis has been performed we have, for each seed, a matrix 8×2 of coordinates superimposed. In order to study the shape variation, we consider landmarks individually, that is, consider how much the coordinates of a single landmark of a seed vary respect to those of the correspondent landmarks of the other seeds. In that way we consider each coordinate as a single variable indicated by a combination of a letter between x or y according to the axis, and of a number from one to eight according to the position of the landmark. For instance the coordinate $x3$ corresponds to the coordinate of the third landmark on the x-axis.

In order to understand the linear relationships between coordinates, we performed correlation matrix shown by Figure 6.13. It points out the presence of strong linear relations between coordinates. Furthermore it is possible to identify two groups of coordinates positively correlated. The first group is made up of $x5$, $x6$, $x7$, $x8$, $y6$, $y7$, $y8$ and $y1$, whereas the second one of the remaining. Immediately it points out that in the first group (consequently in the second one as well) both the "x" and "y" coordinates are all sequential, it means that information of whole shape can be divided into two half shapes positively correlated. Between groups instead the relationships are negative.

Dimensionality reduction

Next step consists in carrying out PCA on the 16 variables, getting the results summarized by Table 6.8. In this case for reaching the threshold of 99% we need of the first eleven principal components. Although their interpretation is not intuitive, in Table 6.9 we reported the loadings of the first

Principal Component	Variance	Proportion of Variance	Cumulative Proportion
PC1	0.0069	0.6771	0.6771
PC2	0.0013	0.1288	0.8059
PC3	0.0004	0.0434	0.8493
PC4	0.0004	0.0351	0.8844
PC5	0.0003	0.0285	0.9130
PC6	0.0002	0.0227	0.9356
PC7	0.0002	0.0168	0.9524
PC8	0.0001	0.0138	0.9663
PC9	0.0001	0.0102	0.9764
PC10	0.0001	0.0085	0.9849
PC11	0.0001	0.0077	0.9926
PC12	0.0001	0.0049	0.9976
PC13	0.0000	0.0024	1.0000
PC14	0.0000	0.0000	1.0000
PC15	0.0000	0.0000	1.0000
PC16	0.0000	0.0000	1.0000

Table 6.8: *Summary of PCA results for coordinates of landmarks data.*

five components. We wanted to highlight how the first principal component, that summarizes the 67.71% of all variance, reproduces the division of the coordinates made above. In fact PC1 is positively correlated with$x5$, $x6$, $x7$, $x8$, $y6$, $y7$, $y8$ and $y1$ (the coordinates of the first group) and negatively with the other ones (the coordinates of the second group). If we project the observations into two factorial planes defined, respectively, by PC1 in x-axis and PC2 in y-axis we get the Figure 6.14. Through the factorial planes it stands out that, as it was expected, the majority of the classes are distributed on the first component. Definitely PC1 does not discriminate perfectly all classes, inasmuch someone is overlapping. Nevertheless classes such as *Paradisu* and *Stanley* are clearly separated from the other ones such as *LaconiE*, so easily discriminable through these landmarks information.

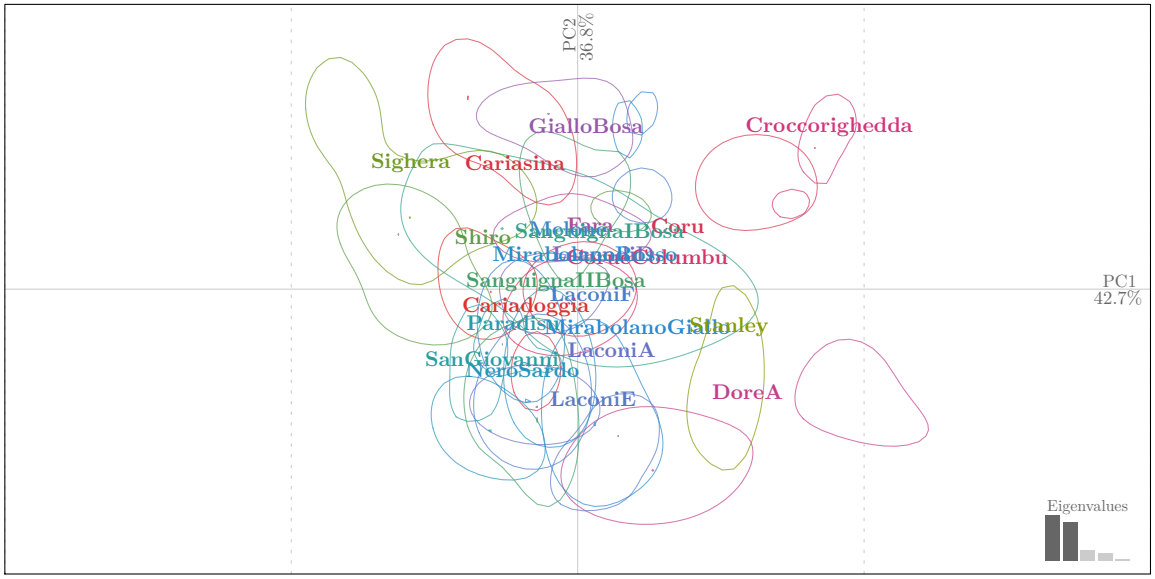
	PC1	PC2	PC3	PC4	PC5
x1	-0.1681	-0.3243	-0.4055	0.3864	-0.0694
x2	-0.2991	-0.0745	-0.1993	-0.4472	0.0979
x3	-0.3774	0.2119	0.2965	-0.2390	-0.0300
x4	-0.1947	0.3876	0.2294	0.3386	0.2669
x5	0.1789	0.1152	-0.2276	0.2726	-0.0825
x6	0.2367	0.0806	-0.3339	-0.1775	-0.3692
x7	0.3654	-0.1282	0.1398	-0.3601	0.1501
x8	0.2582	-0.2682	0.5006	0.2264	0.0362
y1	0.3535	-0.1677	-0.1930	0.1296	0.3968
y2	-0.0607	-0.3176	0.0934	-0.1700	-0.4720
y3	-0.1601	-0.3018	0.1890	-0.0915	0.0810
y4	-0.1461	-0.2002	-0.0012	-0.0318	0.3761
y5	-0.4074	0.0234	-0.0800	0.2664	-0.1686
y6	0.0465	0.3820	-0.2949	-0.1875	0.2445
y7	0.1903	0.3620	0.0896	-0.0744	-0.1164
y8	0.1839	0.2198	0.1971	0.1592	-0.3414

Table 6.9: *Summary of loadings obtained applying PCA to landmarks superimposed coordinates. For space reason just loadings of the first five principle components are shown.*

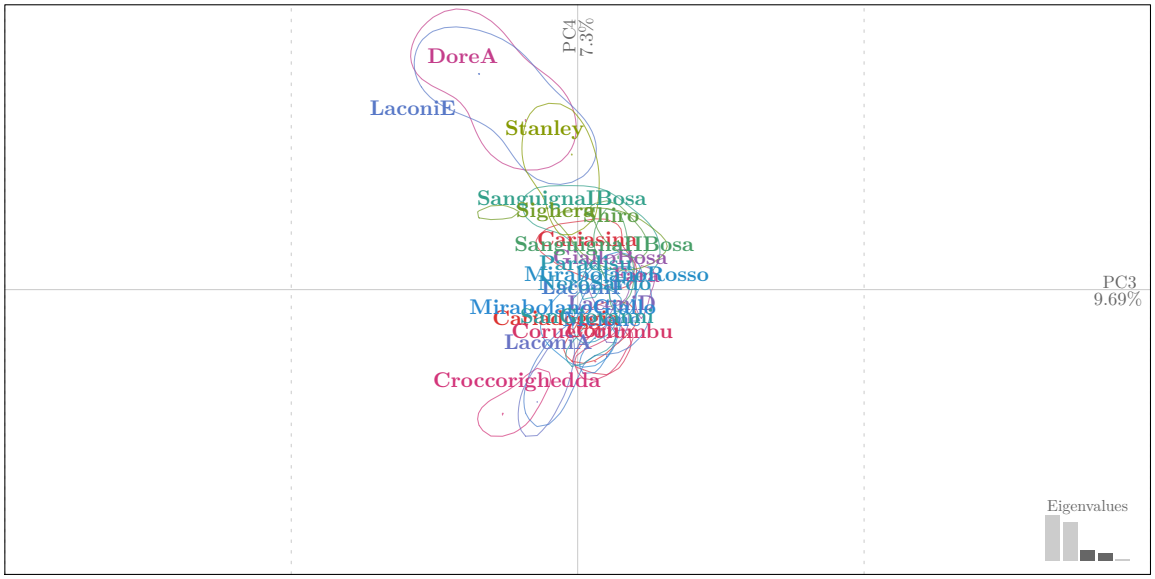
6.4.6 Color data

Many people think that color is an important feature for the classification task. This certainty exists, probably, because color is the first feature anyone notes of an object, so that they assign to it a great power of distinction. At first sight roughly all seeds seem to be characterized by a very similar color, so probably it will not give us a great contribution for classification. Nevertheless we tried to analyze the color differences among our 23 classes. Each seed has been captured twice (once with black background and the other one with white background), and usually it is unlikely the pixels of the first image have an identical intensity value to the correspondent ones of the second image, because some little changes in lighting can occur. Consequently we decided to average them in order to get a single image, and avoiding a subjective choice between them. We have available, for each seed, its empirical distributions of the three color channels Red, Green and Blue. Figure 6.15 shows an illustrative representation of the color empirical distributions of a seed. In order to check the informative power of color for our classification task, performing the Kolmogorov-Smirnov Goodness-of-Fit Test[4] we compared, two at a time, the empirical distributions of the three color channels of the 23 classes, so as to find out if they differ each other. All distributions proved to be different to each other with a p-value < 0.001. The second step consisted of testing if the color distributions of single seeds are recognized as belonging to their own class distribution, but they proved to be different with a p-value < 0.001. It means that seeds of a same class have distributions different each other, and so they do not be used in classification. As a result we decided to do not consider that kind of data in the further stage of our analysis.

[4] Kolmogorov-Smirnov Goodness-of-Fit Test is a non-parametrical test that, making no assumption about the distribution of data, defines if two distributions differ significantly.



(a) Plane defined by PC1 and PC2



(b) Plane defined by PC3 and PC4

Figure 6.11: *Two biplots of the scores (i.e. the new coordinates of the observations in the factorial space) divided by class projected, respectively, in the plane defined by PC1 (x-axis) and PC2 (y-axis) and in the plane defined by PC3 (x-axis) and PC4 (y-axis). All four the principal components are from texture information.*

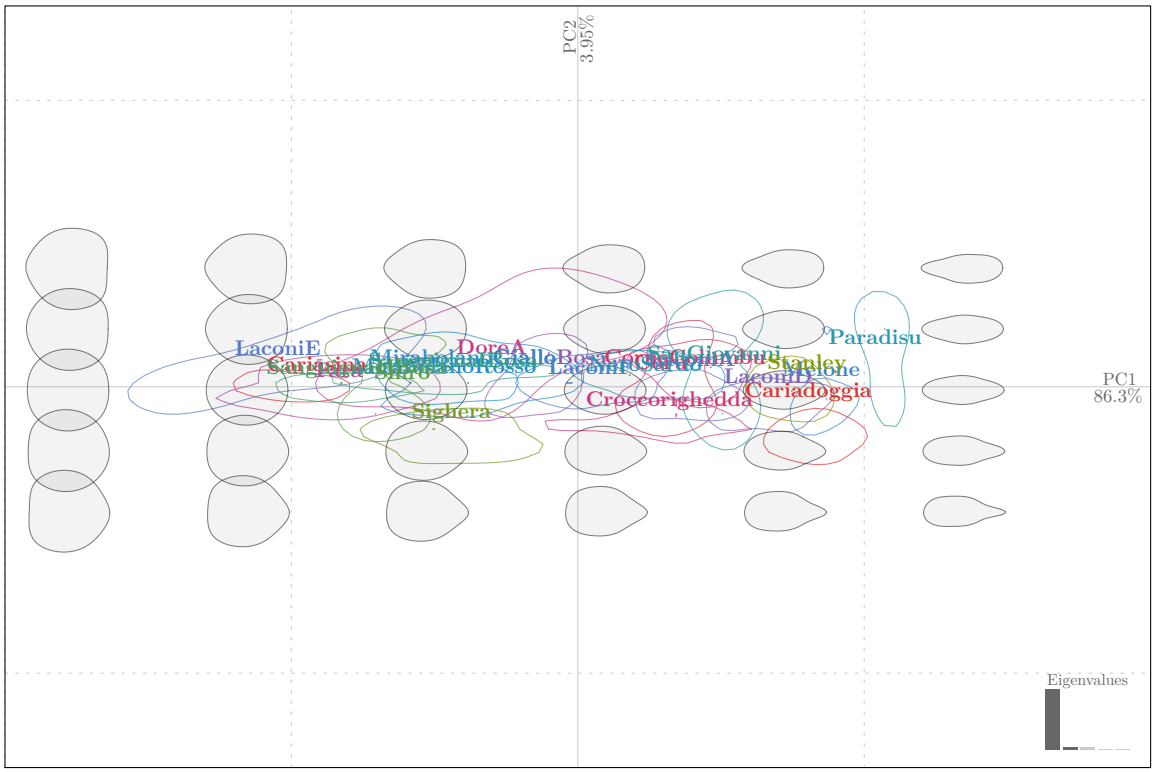


Figure 6.12: *Biplot of the scores (i.e. the new coordinates of the observations in the factorial space) divided by class projected in the plane defined by PC1 (x-axis) and PC2 (y-axis) of outline information. In the background it is possible to see the outline shape that are associated to observations in different parts of the plane. On the other hand they illustrate the kind of information conveys by the components.*

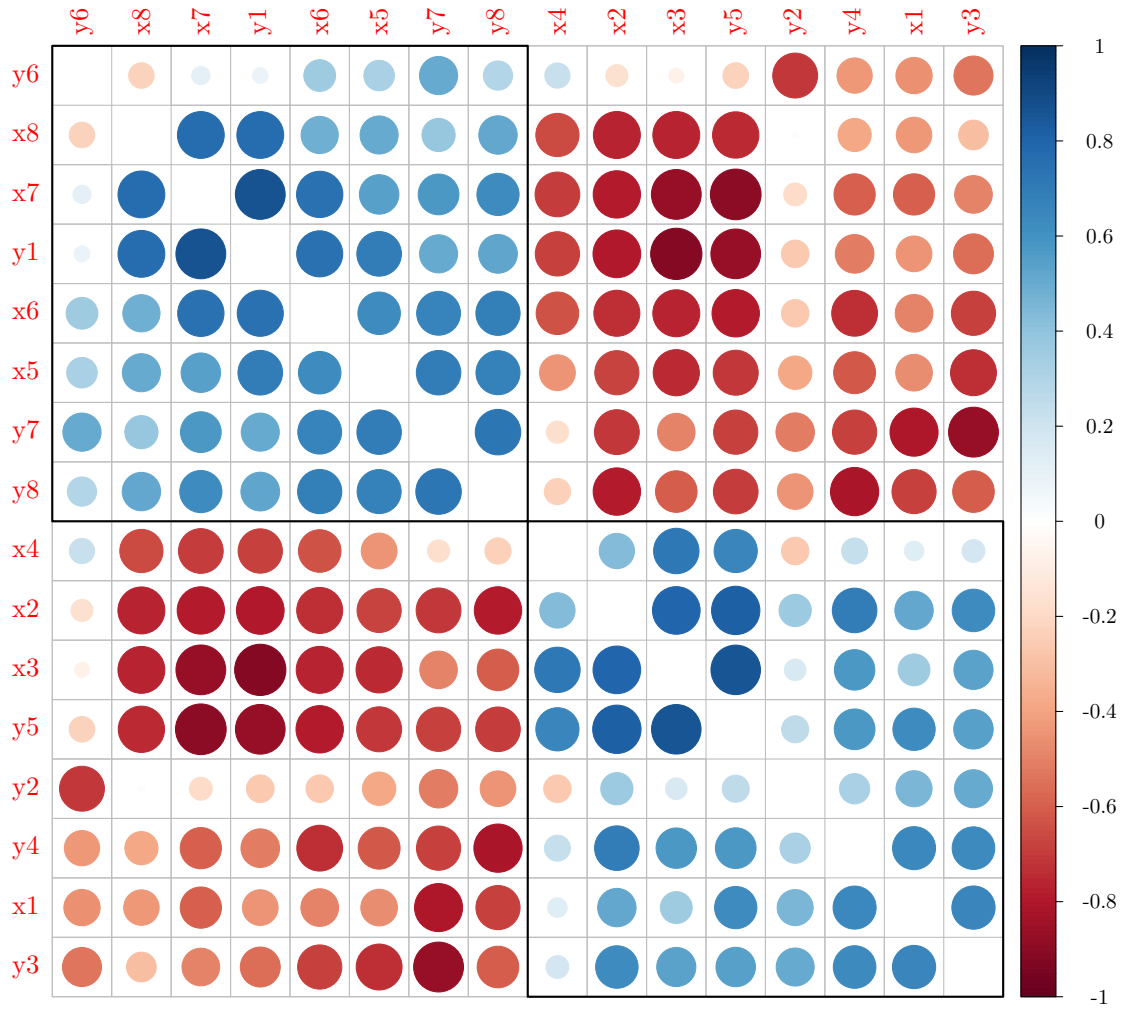


Figure 6.13: *Correlation matrix plot of the 16 coordinates superimposed.*

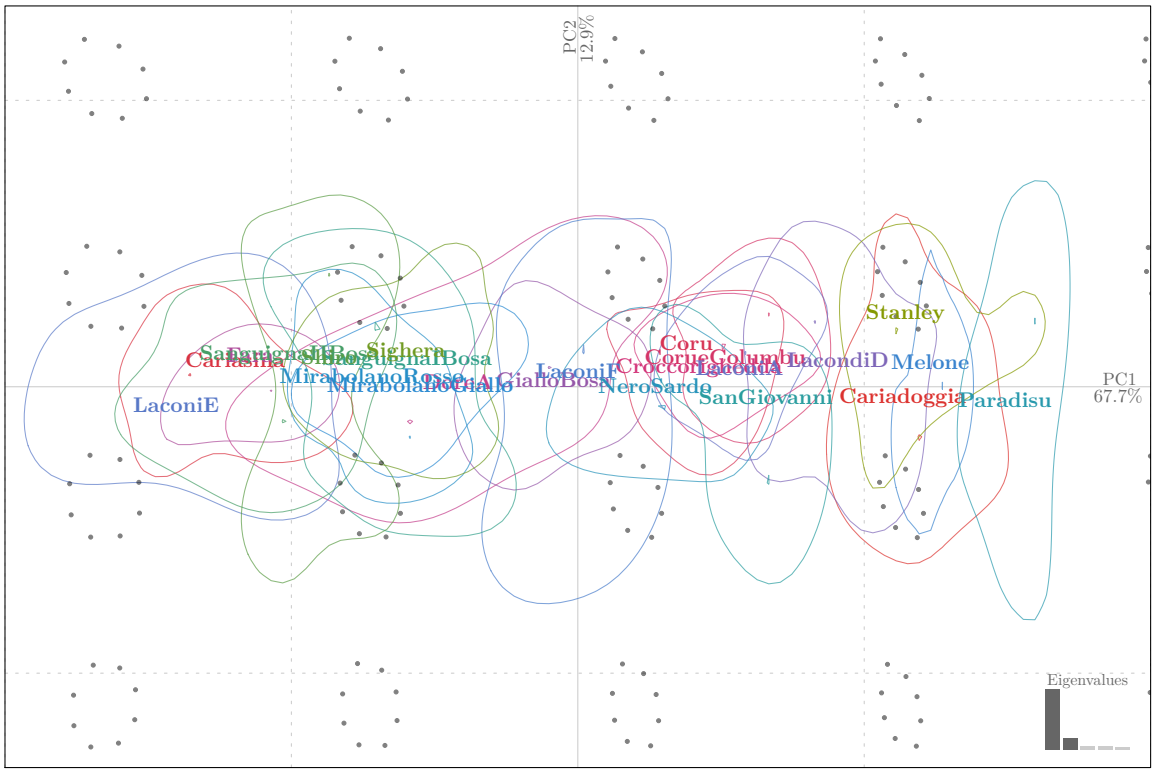


Figure 6.14: *Biplot of the scores (i.e. the new coordinates of the observations in the factorial space) divided by class projected in the plane defined by PC1 (x-axis) and PC2 (y-axis) of landmarks information superimposed. In the background it is possible to see the landmarks distributions that are associated to observations in different parts of the plane. On the other hand they illustrate the kind of information conveys by the components.*

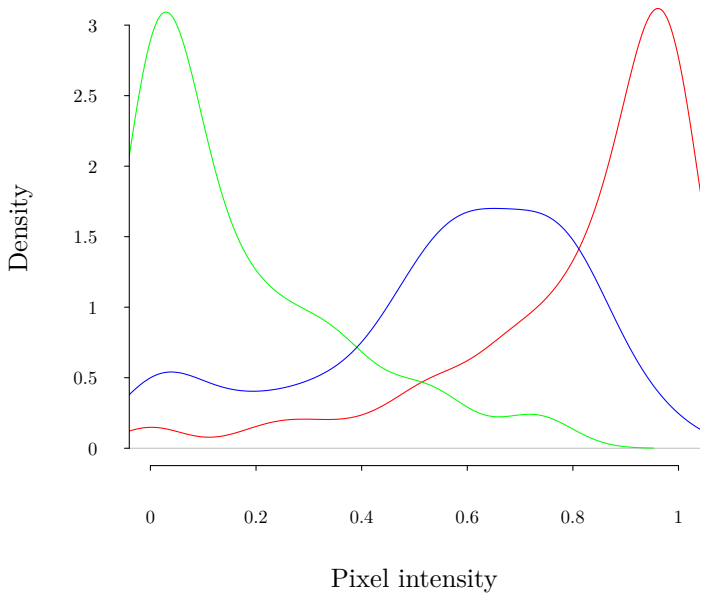


Figure 6.15: *Example of empirical distributions of the Red, Green and Blue color channels of a seed.*

6.5 Classification analysis

The aim of this chapter is to study informative power of the different kind of data and to compare performances of several classification algorithms carried out on them. Nowadays in statistics actually many classification algorithms exist, and obviously it is not possible to consider all of them in this work. Since it is even difficult to try defining a objective criteria so as to select among all of them, we decided to select among the most known subjectively. Definitely we considered Linear Discriminant Analysis (LDA), which is one of the most known in general and in botanical classification field. Then Classification And Regression Trees (CART), Support Vector Machines (SVM) and Naïve Bayes (NB), which are all classification methods very well-known. Furthermore we tried to combine these four classifiers through the tree approach of classifier combination (C_A) introduced in Chapter 3.

On the other hand, so far we dealt with five kind of data: *size*, *texture*, *outline*, *landmarks* and *color*. Nevertheless, unfortunately we had to exclude *color* from our analysis, since we found out it is not useful for our classification task (see Section 6.4.6 for details). Since the four remaining convey different kind of information (even if *outline* and *landmarks* are similar) we tried to merge their corresponding datasets in all possible combinations so as to create new datasets with different informative power. In other words we considered firstly the four datasets individually, then merging them by two, then by three, and finally by four (i.e. merging all four together). In that way we define fifteen different datasets, on which we carried out the algorithms one by one.

Choosing a measure to assess classifier accuracy and comparing them so as to decide which one is the best in a specific sampled population, is very complicated, because several possibilities exist. Labatut and Cherifi (2012) reviewed several measures, focusing on comparing classifiers characterized by having discrete outputs only. The measures analyzed are *Nominal Association Measures*, *Overall Success Rate*, *Marginal Rates*, *F-measure*, *Jaccard Coefficient*, *Classification Success Index*, *Agreement Coefficients* and *Ground Truth Index*. They analyzed the measures in several respects, considering class focus, functional relationship, range, interpretation and correction for chance. From their study it has emerged that some respects are not useful to discriminate the measures. The reasons of this similarity are different. First, all monotonically measures are lead to the same ordering of algorithms, then their range is not important because relative values are considered. Furthermore, about the chance-correction they did not find, in the measures analyzed, relevant estimation for chance. For determining the best measure, the authors discard measures obtained from combination of other different measures because of their complicated or impossible interpretation. Also measures associating weights to class are discarded, because they think it is better using different methods to assign an importance to classes and applying a multiclass measure. At the end, they advise to choose simple measures with straight interpretations. For overall accuracy assessment they point out *Overall Success Rate* (i.e. accuracy), and *True Positive Rate* or *Positive Predictive Value* if the focus is on a specific class. Consequently, since we focus on overall accuracy, we decided to use the *Overall Success Rate*.

Among all possible ways to estimate this measure, it is very common resubstitution. In this case accuracy is estimated using the same data used to construct the classifier, instead of an independent sample. All classification procedures attempt to minimize the misclassification rate, so if we use as test observations those contributed in estimating the classification rule, then it is more likely that they will be well classified, inasmuch information is the same in both cases. As a result resubstitution can give overly optimistic picture of the accuracy of the classifier, that is, an overestimation of classification goodness. Due to this shortcoming we used test sample approach.

It consists in partition sample into independent subsets: a training set and a test set. Training set consists of observations we use to train, or teach, our method how to estimate classification rule, whereas test set of observations used for computing how much the classification rule estimated is good (i.e. the accuracy). As a result we split all 1,396 observations into a training set made up of 1123 units ($\approx$ 80%) and a test set of 273 units ($\approx$ 20%). The splitting has been carried out randomly, the only criteria respected is that the original composition of the 23 classes has been kept.

About the setting of C_A we considered all attributes of each dataset as a unique feature type, whereas λ as equal to 0.5, so as to exclude from the analysis just the classifiers that predict worse than randomly, and to decided to which class is classified we followed "maximum" criteria.

Finally the classification algorithms have been carried out one by one to the fifteen training sets. Then the accuracy rates were computed predicting the observations of the test sets through the classification rules estimated. In Table 6.10 are shown all results obtained. By columns it is possible to check the accuracy rates computed by each algorithm on the fifteen datasets, whereas by rows the accuracy rates computed by all classification algorithms on each dataset.

dataset	CART	NB	LDA	SVM	C_A
s	0.3883	0.4689	0.5018	0.4982	**0.5128**
t	0.3663	0.3846	0.3919	**0.6044**	0.5495
l	0.2857	0.4029	0.3736	**0.4762**	0.4579
o	0.3516	0.5604	0.4396	**0.6007**	**0.6007**
st	0.5971	0.6190	0.6520	0.7766	**0.7802**
sl	0.4615	0.5385	0.5201	0.5934	**0.6300**
so	0.5604	0.6337	0.5751	**0.6996**	0.6813
tl	0.4249	0.6154	0.5788	0.6813	**0.7106**
to	0.5165	0.6300	0.6300	0.6960	**0.7802**
lo	0.3516	**0.5678**	0.4579	0.5604	0.5348
stl	0.5714	0.7326	0.6740	0.7546	**0.8095**
sto	0.6227	0.7436	0.6886	0.7582	**0.8315**
slo	0.5421	0.6337	0.6044	0.6593	**0.6960**
tlo	0.5018	0.6484	0.6227	0.7033	**0.7070**
stlo	0.5934	0.7582	0.7070	0.7436	**0.8242**

Table 6.10: *Summary of accuracy rates obtained carrying out four classification algorithms (CART, NB, LDA and SVM) and their combination (C_A) on fifteen datasets obtained combining in all possible ways the four original datasets. The first column presents the dataset labels, in order to decode them it is enough to know that "s" corresponds to size, "t" to texture "l" to landmarks and "o" to outline, consequently "st" is the dataset made up of size and texture data, "slo" of size, landmarks and outline data, and so on. Finally in bold it is highlighted the best result for each dataset, whereas in blue the best result for each classification rule, and in red the best result overall.*

The most evident result is that C_A got the highest level of accuracy 11 times over 15 (73%), and the highest level overall, classifying correctly the 83.15% of cases using "sto" dataset. Furthermore it is interesting to note that the larger number of datasets merged the larger proportion of the highest level of accuracy got by C_A. In fact, considering datasets individually it gets the highest level of accuracy two times over four (50%), merged by two four times over six (67%), merged

by three four times over four (100%) and by four one time over one (100%). On the other hand, considering only the other four classifiers, SVM and NB can be considered as the best ones. SVM is definitely the most reliable, in fact 12 times over 15 it got the best result among them. It reaches its best (0.7766) when *size* and *texture* are merged. Adding to that shape information the results decrease to 0.7546 using *landmarks*, to 0.7582 using *outline* and to 0.7436 using both *landmarks* and *outline*. Using just *texture* it got 0.6044 that can be consider as a very good result, inasmuch the other three classifiers do not reach 0.40. Instead, NB and LDA got their best results (0.7582 and 0.7070) while the amount of information is maximum, that is, when all four datasets are merged. Finally CART got definitely the worst results inasmuch it got always the lowest accuracy. It got its best result (0.6227) for the same datasets C_A got its best ("sto").

If we analyze the Table 6.10 by rows, it is easy to note that, the overall trend is the higher amount of information, the better performances. Indeed, taking into account the datasets individually, the average of all accuracy values is equal to 0.4608 and it is the lowest one. In fact, the higher value is 0.6044, which is not fantastic, and the lowest is even 0.2857. Among these four datasets the worst one is definitely *landmarks* inasmuch its performances are almost always lower than other ones, whereas it is not possible to state which is the best one since there is not anyone that dominates over others. Considering all those six made up of two original datasets, the average of all accuracy values increases to 0.6018. Moreover the accuracy values of some classifier leapt significantly, reaching values such as 0.7802 and 0.7766. It means that to get significantly information can be enough to merge just two datasets. As we expected, the worst combination is given merging *landmarks* and *outline*, in fact they express very similar information so that the redundancy will be very high. On the contrary the best combination is definitely "st", it can mean that shape information is not as powerful as size and texture information. Adding one more data source, the average of all accuracy values increases again to 0.6753. Furthermore with this combination is got the best overall performance by C_A, and CART reaches its top with 0.6227. Even here the worst combinations are those including at the same time both *landmarks* and *outline*. Finally, in the last dataset composed by all original ones, the average of all accuracy values reaches its top to 0.7253.

In order to study the performances and the utility of C_A, it is necessary to compare its results to those got by the four classifiers in terms of reduction of misclassification error, in other words we want to measure the advantages of using C_A instead of the other ones individually. From the results presented in Table 6.11 emerged that C_A has been less convenient 5 times over 60 and always with SVM. Among those just one deserves notice, when SVM was trained on *texture*, in the other four cases the differences are very small. For analyzing the advantages of C_A respect to the amount of information, in Figure 6.16 we summarized the results of Table 6.11 averaging the values by the number of datasets grouped. As it is possible to see from the plot, the tendency is C_A reduces increasingly misclassification error obtained from the classifiers individually as the amount of information used for training models increases.

Finally, as the final model for our classification task we have chosen the best one: that of C_A trained on "sto" dataset. Figure 6.17 shows the tree plot constructed aggregating classes step by step. It is a useful tool for understanding the similarity among classes in general and, in analysis where the aim is to check if some classes in reality are the same, for identifying classes on which focus on. In Table 6.12, instead, it is reported the confusion matrix with the values normalized by reference classes and expressed in percentage. It is interesting how the information of tree plot and confusion matrix coincide. Following tree plot the most similar classes are those grouped at the first steps, such as *Coru* and *CorueColombu*, *Cariadoggia* and *SanGiovanni*, *Cariasina* and *SanguignaIIBosa*. If we look at these classes in the confusion matrix we can note in correspondence

dataset	CART	NB	LDA	SVM
s	0.2036	0.0827	0.0220	0.0292
t	0.2891	0.2679	0.2591	-0.1388
l	0.2411	0.0921	0.1345	-0.0349
o	0.3841	0.0916	0.2875	-0.0001
st	0.4545	0.4230	0.3684	0.0163
sl	0.3129	0.1983	0.2289	0.0900
so	0.2750	0.1299	0.2500	-0.0610
tl	0.4968	0.2476	0.3130	0.0919
to	0.5454	0.4059	0.4059	0.2770
lo	0.2825	-0.0763	0.1419	-0.0583
stl	0.5555	0.2876	0.4157	0.2238
sto	0.5534	0.3428	0.4588	0.3030
slo	0.3361	0.1701	0.2316	0.1076
tlo	0.4118	0.1668	0.2234	0.0125
stlo	0.5676	0.2728	0.4001	0.3144

Table 6.11: *Reduction of misclassification error if C_A is used instead of a classifier performed individually.*

to their paired class there is the highest misclassification error.

	Cariadoggia	Cariasina	Coru	CorueColumbu	Croccorighedda	DoreA	Fara	GialloBosa	LaconiD	LaconiA	LaconiE	LaconiF	Melone	MirabolanoGiallo	MirabolanoRosso	NeroSardo	Paradisu	SanGiovanni	SanguignaIBosa	SanguignaIIBosa	Shiro	Sighera	Stanley
Cariadoggia	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	57	0	0	0	0	0
Cariasina	0	71	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	6	8	0	0	0
Coru	0	0	64	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CorueColumbu	0	0	18	81	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0
Croccorighedda	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DoreA	0	0	0	0	0	67	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0
Fara	0	0	0	0	0	0	100	0	0	0	0	0	0	0	7	0	0	0	0	17	6	0	0
GialloBosa	0	0	0	0	0	0	0	83	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LaconiD	0	0	9	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LaconiA	0	0	0	6	0	0	0	0	0	94	0	7	0	0	0	0	0	0	0	0	0	0	0
LaconiE	0	0	0	0	0	33	0	0	0	0	100	0	0	6	0	0	0	0	0	0	0	0	0
LaconiF	0	0	9	0	0	0	0	0	0	0	0	57	0	0	0	0	0	0	0	0	0	0	0
Melone	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0
MirabolanoGiallo	0	0	0	0	0	0	0	17	0	0	0	0	0	83	0	0	0	0	0	0	0	0	0
MirabolanoRosso	14	0	0	0	0	0	0	0	0	0	0	7	0	0	80	0	0	0	0	0	6	0	0
NeroSardo	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	89	0	29	0	0	0	0	0
Paradisu	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	67	0	0	0	0	0	0
SanGiovanni	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	14	0	0	0	0	0
SanguignaIBosa	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	94	17	6	0	0
SanguignaIIBosa	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	58	6	0	0
Shiro	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	78	6	0
Sighera	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	88	0
Stanley	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	100

Table 6.12: *Confusion matrix of C_A trained on "sto" dataset. In rows there are the predicted classes and in columns the reference ones. All values are normalized by column and expressed in percentage.*

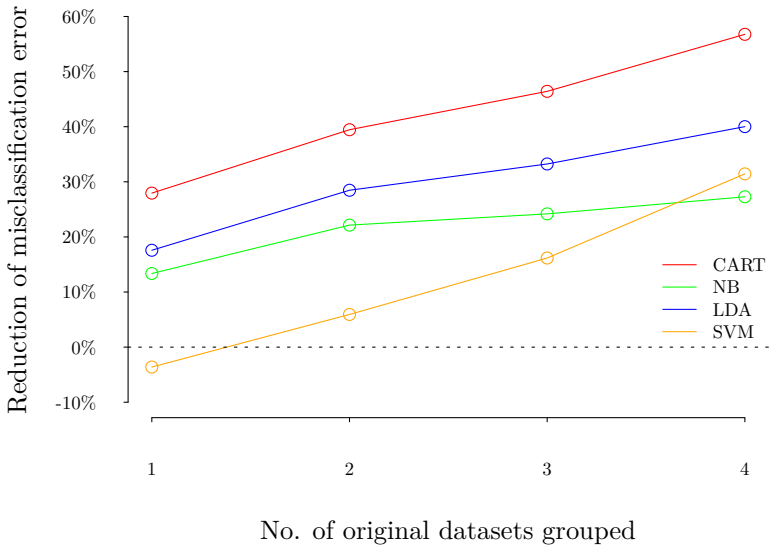


Figure 6.16: *Distributions of reduction of misclassification provides by C_A respect to CART, NB, LDA, and SVM.*

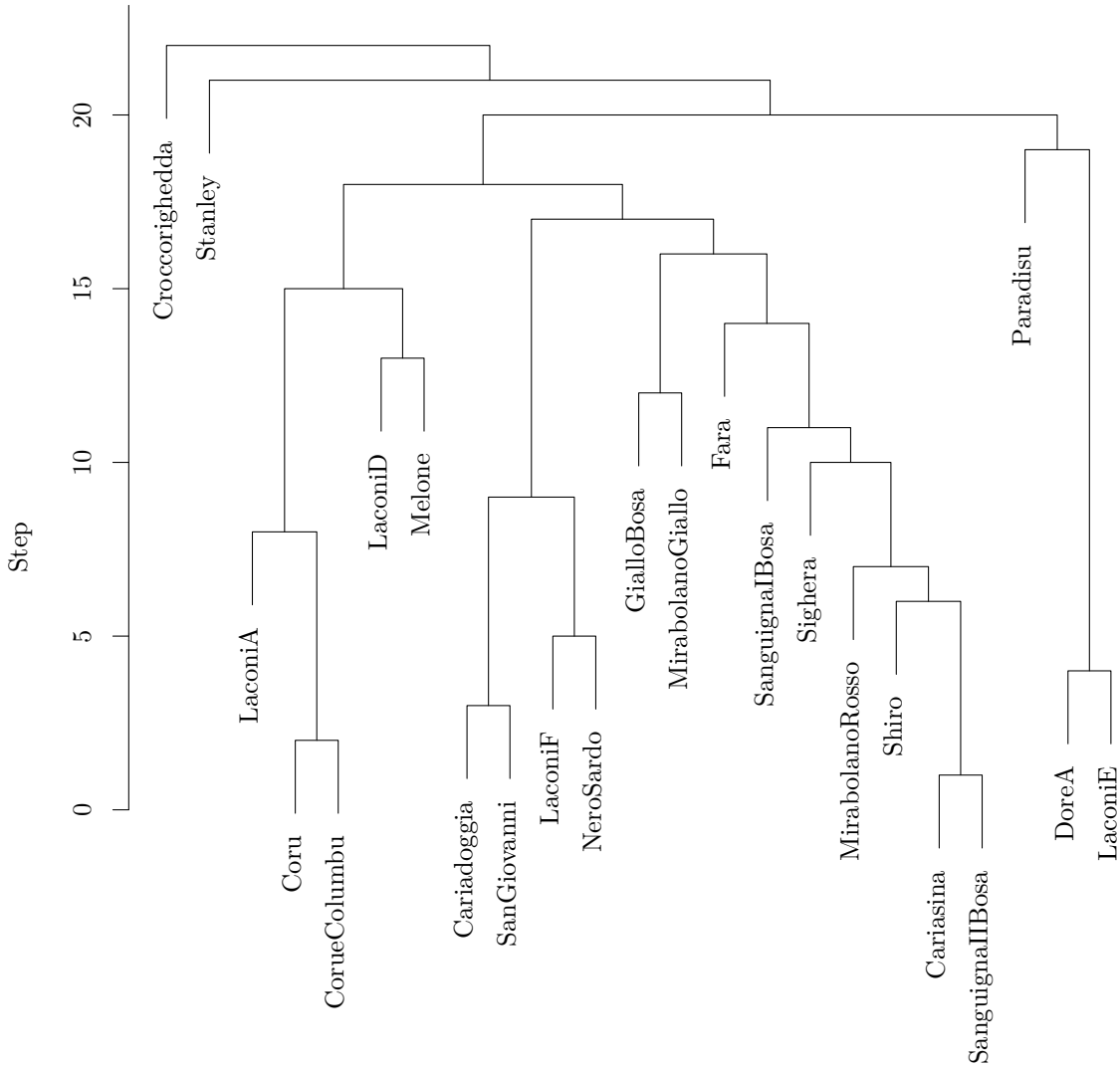


Figure 6.17: *Tree plot of C_A trained on "sto" dataset.*

Conclusions

This dissertation deals with statistical methodologies to apply to morphological classification of seeds through extracting information directly from their digital images. It concentrates more on the classification task, trying to enhance the quality of prediction, and on the automatizing of the classification process. These tasks are very important in botany because they avoid human contradictions in seed classification and to save a lot of time to specialized botanists.

Firstly we focused on describing all stages necessary to move from a picture containing raw information of scanned objects to a data matrix usable as input for further statistical analyses. We illustrated how to convert an image so as to enhance its inner contrast in order to get easier the image segmentation. It has been introduced an approach that adapts a widely used method for detecting moving objects from video, called background subtraction (foreground detection), to image segmentation framework. It has been shown how it assists segmentation process to get good results, and allows to automate the process when foreground color of images is not constant, as well as speeding it up significantly. Then methods for enhancing quality of objects and removing residual noise have been illustrated. At the end of the first chapter, a kind of general features that characterized the objects are explained, pointing out which information they convey.

In the second chapter we focused on tools used by modern morphometrics and the theoretical concepts of shape analysis. Firstly we explained the concept of landmarks and its importance. Then we showed the different strategies that can be followed so as to remove from the Configuration Space the influences of location, rotation and scale moving to the Shape Space, illustrating step by step how to transform Euclidean coordinates of objects into Kendall coordinates and into Bookstein coordinates. Furthermore we illustrated General Procrustes Methods, that are used for analyzing distribution of objects optimally superimposed. Then we dealt with Fourier Analysis, a mathematical method widely used in several fields for decomposing and analyzing periodic signals into a weighted sum of simpler sinusoidal component functions, and in our case for summarizing the geometrical information of the object outlines. Finally we described several approaches for fitting a function in case outlines are opened.

In the third chapter we presented an original tree approach for combining different classifiers. Since in a classification problem with large number of classes the complexity is high, this algorithm splits the complex problem of classifying among C classes into $C-1$ sub problems less complex than the original one, each of them classifying between only two classes. It builds a binary tree of $C-1$ nodes, and places a classification rule in each node, taking advantage of the different prediction capability of the classifiers. Helped by a real dataset, we found that the tree approach proposed

can be a useful tool for enhancing the goodness of prediction, although this is not true for every situation, but according to kind of data and type of classifiers.

In the fourth chapter we presented an original approach aimed at evaluating the reliability of a classification rule. This task is pursued by re-training the classifier on resampled versions of the original data. User-defined misclassification costs are assigned to the obtained confusion matrices and then used as inputs in a Beta regression model which provides a cost-sensitive weighted classification index. The latter is used jointly with another index measuring dissimilarity in distribution between observed classes and predicted ones. Both index are defined in $[0, 1]$ so that their values can be graphically represented in a $[0, 1]^2$ space. The examination of the points in the $[0, 1]^2$ space for each classifier, computing the convex hull, allows us to evaluate its reliability on the basis of the relationship between the values of both indexes obtained on the original data and on resampled versions of it. Even in this case we tested our original approach on real data, in order to check its operation.

In the fifth chapter, and in Appendix A more in details, we presented the R functions we created in order to check the reliability of the original theoretical approaches presented in the previous chapters and to be able to perform the analyses of the next chapter.

In the last chapter we applied all theoretical concepts developed in the previous chapters to real botanical data. The data consists in germplasm data, and the main goal was to study how this kind of data respond to morphometrics approaches of classification, comparing the results obtained using different kind of classification algorithms so as to check if and how much their performances of classification are distance. Furthermore we combined all methods developed into a single automated process, that is resulted consistent and efficient, able to perform morphological classification of seeds extracting information directly from their digital images.

Appendices

Appendix A

R functions

A.1 Generals

A.1.1 Diameters

Description

Compute all diameters of a closed outline passing as close as possible to its centroid.

Usage

```
diameters(coo, int = 0)
```

Arguments

`coo`	a matrix object. Coordinates of closed outlines.
`int`	a numeric or integer one-length object. Number of point to ignore around where diameter starts.

Values

`diameters`	a matrix $n \times 4$ object where the in first two columns there are the coordinates of the first point, and in the last two columns the coordinates of the second point.
`length`	a numeric vector with the lengths of all diameters.
`theta`	a numeric vector. It returns the angle in radians between the x-axis and the diameters.
`centroid`	a numeric vector with the coordinates of centroid.

Examples

```
mat <- matrix(c(1,2,3,4,3,8,0,4), 4, 2)
d <- diameters(mat)

len_out <- dim(d$diameters)[1]

# plot points of closed outline
plot(mat, asp = 1)

# plot centroid
points(d$centroid[1], d$centroid[2], col = "green")

# plot all diameters
for(i in 1:len_out){
  lines(c(d$diameters[i,1], d$diameters[i,3]),
  c(d$diameters[i,2],d$diameters[i,4]), col = "red")
}
```

Function

```
diameters <- function(coo, int = 0){

  if(!is.matrix(coo)) stop("coo must be a matrix")
  if(dim(coo)[2]!=2) stop("coo must have 2 columns")
  if(!(is.numeric(int) | is.integer(int))) stop("int must be numeric or integer")
  if(length(int)!=1) stop("int must be one-length object")
  if(int < 0) stop("int must be >= 0")

  int <- as.integer(int)
  cF <- apply(coo,2,mean)

  len_out <- dim(coo)[1]
  dmts <- matrix(rep(NA_real_,len_out*4), len_out, 4)
  len_dmt <- rep(NA_real_,len_out)
  ang <- rep(NA_real_,len_out)

  for(i in 1:len_out){
    y <- rep(NA_real_,len_out)
    for(j in (1:len_out)[-c(abs((i:(i+int)) - (int/2)))]){
      y[j] <- abs(((coo[j,1] - coo[i,1])/(cF[1] - coo[i,1])*
                   (cF[2] - coo[i,2]) + coo[i,2]) - coo[j,2])
    }
    dmts[i,] <- c(coo[i,], coo[which.min(y),])
    len_dmt[i] <- sqrt((dmts[i,1] - dmts[i,3])^2 + (dmts[i,2] - dmts[i,4])^2)
    ang[i] <- atan((dmts[i,2] - dmts[i,4])/(dmts[i,1] - dmts[i,3]))
  }
```

```
  return(list(diameters = dmts,
              length = len_dmt,
              theta = ang,
              baricenter = cF))
}
```

A.1.2 Gaussian pairing

Description

Compute a matrix in which each element is a neighbor of the correspondent element of `x`.

Usage

```
gPairing(x, sigma = 25)
```

Arguments

`x`	a matrix object.
`sigma`	a numeric or integer length-one object.

Details

Each element of the computed matrix is paired with a element of `x`. In that pairing process a randomized scheme is applied: sampling by Gaussian pairing. It consists in choosing the element of the computed matrix randomly according to a displacement vector from an isotropic bivariate Gaussian. In other words, for each element of `x` the horizontal and vertical size neighborhood for locating the element of the computed matrix are each drawn from a univariate Gaussian distribution with mean equal to 0 and variance to `(2/pi)sigma^2`, getting `sigma` as the expected distance between paired pixels.

Values

It returns a matrix object.

Function

```
gPairing <- function(x, sigma = 25){

  if(!is.matrix(x)) stop("x must be a matrix")
  if(!(is.numeric(sigma) | is.integer(sigma))) stop(
    "sigma must be numeric or integer")
  if(length(sigma)!=1) stop("length(sigma) must be 1")

  r <- 1:dim(x)[1]
  c <- 1:dim(x)[2]
  y <- x[gSampling(x = r, sigma = sigma), gSampling(x = c, sigma = sigma)]
```

```
  return(y)
}
```

A.1.3 Sampling from Gaussian distribution

Description

Compute drawings from a truncated Gaussian distribution.

Usage

```
gSampling(x, sigma = 25)
```

Arguments

`x`	a numeric or integer vector of length greater than one.
`sigma`	a numeric or integer length-one vector.

Details

`gSampling` draws `length(x)` elements from a truncated Gaussian distribution with lower and upper equal to `min(x)` and `max(x)`, mean to `x` and standard deviation to `sqrt((2/pi)*sigma^2)`.

Values

It returns a numeric vector of the same length of `x`.

Examples

```
x <- c(4,5,7,15)

set.seed(123)
gSampling(x, sigma = 3)     # 8  5  7 12
```

Function

```
gSampling <- function(x, sigma = 25){

  if(!(is.numeric(x) | is.integer(x))) stop("x must be numeric or integer")
  if(length(x)<2) stop("length(x) must be > 1")
  if(!(is.numeric(sigma) | is.integer(sigma))) stop(
    "sigma must be numeric or integer")
  if(length(sigma)!=1) stop("length(sigma) must be 1")

  x <- as.numeric(x)
  u <- round(rtruncnorm(length(x), mean = x, a = min(x),
                        b = max(x), sd = sqrt((2/pi)*sigma^2)))
  return(u)
```

```
}
```

A.1.4 Local mean

Description

Compute local means for each point of `n` considering as neighbors those within a distance `w/2`.

Usage

```
lMean(n, w = 30)
```

Arguments

`n`	a matrix object. The input image.
`w`	a numeric or integer length-one object. The dimension of the window to define neighbors.

Values

It returns a matrix object.

Examples

```
set.seed(123)
m <- matrix(sample.int(25, size = 16, replace = TRUE), 4, 4)

m             #      [,1] [,2] [,3] [,4]
              # [1,]    8   24   14   17
              # [2,]   20    2   12   15
              # [3,]   11   14   24    3
              # [4,]   23   23   12   23

lMean(m, w = 2)
              #      [,1]  [,2]  [,3] [,4]
              # [1,] 0.50  3.50  6.75 3.75
              # [2,] 4.00 13.00 13.50 4.50
              # [3,] 9.25 18.25 15.50 6.50
              # [4,] 5.75  8.75  8.75 5.75
```

Function

```
lMean <- function(n, w = 30){

  if(!(is.matrix(n))) stop("n must be a matrix object")
  if(!(is.numeric(w) | is.integer(w))) stop("w must be numeric or integer")
  if(length(w)!=1) stop("w must be one-length object")
  if(w<2) stop("w must be >= 2")
```

```
  w <- as.integer(w)
  ii <- intImg(n)
  x <- 1:dim(ii)[1]
  y <- 1:dim(ii)[2]

  xInf <- ifelse((x - w/2) > 1, floor(x - w/2), 1)
  xSup <- ifelse((x + w/2) > dim(n)[1], dim(n)[1], floor(x + w/2))
  yInf <- ifelse((y - w/2) > 1, floor(y - w/2), 1)
  ySup <- ifelse((y + w/2) > dim(n)[2], dim(n)[2], floor(y + w/2))

  m <- (ii[xSup, ySup] +
        ii[xInf, yInf] -
        ii[xSup, yInf] -
        ii[xInf, ySup])/w^2

  return(m)
}
```

A.1.5 Local standard deviation

Description

Compute local standard deviation for each point of `n` considering as neighbors those within a distance `w/2`.

Usage

```
lSd(n, w = 30)
```

Arguments

`n`	a matrix object. The input image.
`w`	a numeric or integer length-one object. The dimension of the window to define neighbors.

Values

It returns a matrix object.

Examples

```
set.seed(123)
m <- matrix(sample.int(25, size = 16, replace = TRUE), 4, 4)

m             #      [,1] [,2] [,3] [,4]
              # [1,]    8   24   14   17
```

```
              # [2,]   20    2   12   15
              # [3,]   11   14   24    3
              # [4,]   23   23   12   23

lSd(m, w = 2)
              #           [,1]     [,2]     [,3]     [,4]
              # [1,] 0.8660254 4.974937 6.832825 6.495191
              # [2,] 5.8309519 7.810250 7.500000 6.184658
              # [3,] 9.7819988 5.309190 8.616844 9.604686
              # [4,] 9.9592921 9.575359 9.575359 9.959292
```

Function

```
lSd <- function(n, w = 30){

  if(!(is.matrix(n))) stop("n must be a matrix object")
  if(!(is.numeric(w) | is.integer(w))) stop("w must be numeric or integer")
  if(length(w)!=1) stop("w must be one-length object")
  if(w<2) stop("w must be >= 2")

  ii2 <- intImg((n^2))
  x <- 1:dim(ii2)[1]
  y <- 1:dim(ii2)[2]
  xInf <- ifelse((x - w/2) > 1, floor(x - w/2), 1)
  xSup <- ifelse((x + w/2) > dim(n)[1], dim(n)[1], floor(x + w/2))
  yInf <- ifelse((y - w/2) > 1, floor(y - w/2), 1)
  ySup <- ifelse((y + w/2) > dim(n)[2], dim(n)[2], floor(y + w/2))

  m <- ii2[xSup, ySup] +
       ii2[xInf, yInf] -
       ii2[xSup, yInf] -
       ii2[xInf, ySup]

  v <- m/(w^2) - lMean(n = n,w = w)^2

  return(sqrt(abs(v)))
}
```

A.1.6 Median value of 3 elements

Description

Compute the median value among 3 elements.

Usage

```
median3(x)
```

Arguments

`x`	a numeric or integer vector containing the values whose median is to be computed.

Values

It returns a length-one vector of the same type as `x`.

Examples

```
x <- c(4,3,9)

median3(x)          # 4
```

Function

```
median3 <- function(x){

  if(!(is.integer(x) | is.numeric(x))) stop(
    "x must be a numeric or integer vector")
  if(length(x)!=3) stop("length of x must be 3")

  if (x[1] > x[2]) {
    if (x[2] > x[3]) {
      return(x[2])
    } else if (x[1] > x[3]) {
      return(x[3]);
    } else {
      return(x[1])
    }
  } else {
    if (x[1] > x[3]) {
      return(x[1])
    } else if (x[2] > x[3]) {
      return(x[3])
    } else {
      return(x[2])
    }
  }
}
```

A.1.7 Delta Pairing

Description

Compute the difference between `x` and its pairing matrix.

Usage

```
deltaPairing(x, sigma = 25)
```

Arguments

`x`	a matrix object.
`sigma`	a numeric or integer length-one vector.

Values

It returns a matrix object.

Examples

Function

```
deltaPairing <- function(x, sigma = 25){

  if(!is.matrix(x)) stop("x must be a matrix")
  if(!(is.numeric(sigma) | is.integer(sigma))) stop(
    "sigma must be numeric or integer")
  if(length(sigma)!=1) stop("length(sigma) must be 1")

  dp <- x - gPairing(x = x, sigma = sigma)

  return(dp)
}
```

A.1.8 Subset combinations

Description

Compute all possible subsets made up, each one, of all observations that belong to two levels of a specimen variable.

Usage

```
U(data, pos.class)
```

Arguments

`data`	a data.frame object.
`pos.class`	a numeric or integer one-length object. It indicates the position of the specimen variable.

Values

A list with all the subsets.

Examples

```
df <- data.frame(
  specimen = factor(c("A","C","B","A")),
  x1 = 1:4,
  x2 = 3:6)

U(df, 1)

# $A_vs_B
#   specimen x1 x2
# 1        A  1  3
# 3        B  3  5
# 4        A  4  6

# $A_vs_C
#   specimen x1 x2
# 1        A  1  3
# 2        C  2  4
# 4        A  4  6

# $B_vs_C
#   specimen x1 x2
# 2        C  2  4
# 3        B  3  5
```

Function

```
U <- function(data, pos.class){

  if(!is.data.frame(data)) stop("data must be a data.frame")
  if(!(is.numeric(pos.class) | is.integer(pos.class))) stop(
    "pos.class must be numeric or integer")
  if(length(pos.class)!=1) stop("pos.class must be of length 1")
  if(!(pos.class %in% 1:dim(data)[2])) stop(
    paste0("pos.class must be from 1 to ",dim(data)[2]))

  lev <- levels(data[,pos.class])
  com <- combn(lev,2)
  l <- vector("list",dim(com)[2])
  for(i in 1:dim(com)[2]){
    l[[i]] <- subset(data, data[,pos.class] %in% com[,i])
    l[[i]][,pos.class] <- factor(l[[i]][,pos.class])
```

```
    names(l)[[i]] <- paste0(com[1,i],"_vs_",com[2,i])
  }
  return(l)
}
```

A.1.9 Counting of matches

Description

Count the matches to argument `pattern` within each element of a character vector `x` split by `split`.

Usage

```
strcount(x, pattern, split)
```

Arguments

`x`	a character vector.
`pattern`	a character vector to be matched.
`split`	a character vector of splitting.

Values

A one-length integer object.

Examples

```
strcount("alfa_beta","","_") # 2

strcount("alfa_beta_gamma","","_") # 3
```

Function

```
strcount <- function(x, pattern, split){

  if(!(is.character(x))) stop("x must be character")

  unlist(lapply(
    strsplit(x, split),
    function(z) na.omit(length(grep(pattern, z)))
  ))
}
```

A.2 Image analysis

A.2.1 Color RGB intensity values

Description

Extract the intensity values of the three channels RGB from object pixels.

Usage

```
colorsValue(image, objects)
```

Arguments

`image`	an Image object or an array.
`objects`	a matrix. It must has zero values for background pixels, and non-zero values for foreground pixels.

Values

`red`	a vector of intensity values of Red channel.
`green`	a vector of intensity values of Green channel.
`blue`	a vector of intensity values of Blue channel.

Function

```
colorsValue <- function(image, objects){

  if(!(is.Image(image) | is.array(image))) stop(
    "image must be an Image object or an array")
  if(!(is.matrix(objects))) stop("objects must be a matrix")

  red2 <- as.vector(ifelse(objects!=0,image[,,1],NA))
  green2 <- as.vector(ifelse(objects!=0,image[,,3],NA))
  blue2 <- as.vector(ifelse(objects!=0,image[,,2],NA))

  return(list(red = red2[!is.na(red2)],
    green = green2[!is.na(green2)],
                  blue = blue2[!is.na(blue2)]))
}
```

A.2.2 Integral Image

Description

Computes an integral image of the input intensity image.

Usage

```
intImg(x)
```

Arguments

x	a matrix object. The input image.

Values

It returns a matrix object.

Examples

```
set.seed(123)
m <- matrix(sample.int(25, size = 16, replace = TRUE), 4, 4)

m             #      [,1] [,2] [,3] [,4]
              # [1,]    8   24   14   17
              # [2,]   20    2   12   15
              # [3,]   11   14   24    3
              # [4,]   23   23   12   23

intImg(m)     #      [,1] [,2] [,3] [,4]
              # [1,]    8   32   46   63
              # [2,]   28   54   80  112
              # [3,]   39   79  129  164
              # [4,]   62  125  187  245
```

Function

```
intImg <- function(x){

  if(!(is.matrix(x))) stop("x must be a matrix object")

  y <- apply(apply(x, 1, cumsum), 1, cumsum)

  return(y)
}
```

A.2.3 Landmarks

Description

Usage

```
landmarks(coo, nldks, int = 0)
```

Arguments

`coo`	a matrix object. Coordinates of closed outlines.
`nldks`	a numeric or integer one-length object. Number of landmarks to consider.
`int`	a numeric or integer one-length object. Number of point to ignore around where diameter starts.

Values

A matrix or array object.

Function

```
landmarks <- function(coo, nldks, int = 0){

  if(!is.matrix(coo)) stop("coo must be a matrix")
  if(dim(coo)[2]!=2) stop("coo must have 2 columns")
  for(i in c("nldks","int")){
    if(!(is.numeric(get(i)) | is.integer(get(i)))) stop(
      paste0(i, " must be numeric or integer"))
    if(length(get(i))!=1) stop(paste0(i, " must be of length 1"))
  }
  if(int < 0) stop("int must be >= 0")
  if(nldks < 2) stop("nldks must be >= 2")

  nldks <- as.integer(nldks)
  d <- diameters(coo, int)

  ldks <- matrix(rep(NA_real_,nldks*2), nldks, 2)
  n <- d$diameters[which.max(d$length),]
  cF <- d$baricenter
  length_out <- dim(coo)[1]
  points_step <- floor(length_out/nldks)

  if(dist(n[1:2]-cF) > dist(n[3:4]-cF)){
    start_point <- n[1:2]
  } else {start_point <- n[3:4]}

  out1 <- coo[which(apply(t(t(coo)==start_point),1,sum)==2):length_out,]
  out2 <- coo[1:which(apply(t(t(coo)==start_point),1,sum)==2),]
  out_new <- rbind(out1,out2)
  s <- seq(from = 1, by = points_step, length.out=nldks)

  ldks <- out_new[s,]

  return(ldks)
```

```
}
```

A.2.4 Predominant Chromatic Channel

Description

Compute the Predominant Chromatic Channel removing lower or upper quantiles from the Predominant Chromatic Data and scaling it.

Usage

```
pcc(x, sigma = 25, Yaxis = 0.6686, eta = 0.001)
```

Arguments

`x`	an Image object or an array.
`sigma`	a numeric or integer length-one vector.
`Yaxis`	a numeric or integer length-one vector. It corresponds to the maximum length of Luminance axis.
`eta`	a numeric length-one vector. It represents the lower or upper quantiles (assumed to be outliers) to remove.

Details

The lower and upper quantiles of distribution are identified by `eta`.

Values

An Image object or a matrix object.

Function

```
pcc <- function(x, sigma = 25, Yaxis = 0.6686, eta = 0.001){

  if(!(is.Image(x) | is.array(x))) stop("x must be an Image object or an array")
  if(!(is.numeric(sigma) | is.integer(sigma))) stop(
    "sigma must be numeric or integer")
  if(length(sigma)!=1) stop("length(sigma) must be 1")
  if(eta<0 | eta>=0.5) stop("eta must be >= 0 and < 0.5")

  pcd <- pcd(x=x,sigma=sigma,Yaxis=Yaxis)

  pcc<-pcd/(quantile(abs(pcd),prob=(1-eta)))

  return(pcc)
}
```

A.2.5 Predominant Chromatic Data

Description

Compute the Predominant Chromatic Data projecting the chromatic data onto the predominant chromatic axis.

Usage

```
pcd(x, sigma = 25, Yaxis = 0.6686)
```

Arguments

`x`	an Image object or an array.
`sigma`	a numeric or integer length-one vector.
`Yaxis`	a numeric or integer length-one vector. It corresponds to the maximum length of Luminance axis.

Values

An Image object or a matrix object.

Function

```
pcd <- function(x, sigma = 25, Yaxis = 0.6686){

  if(!(is.Image(x) | is.array(x))) stop("x must be an Image object or an array")
  if(!(is.numeric(sigma) | is.integer(sigma))) stop(
    "sigma must be numeric or integer")
  if(length(sigma)!=1) stop("length(sigma) must be 1")

  colSpace <- YPQHS(x)
  step1 <- sign(deltaPairing(x = colSpace[,,1], sigma = sigma))*
    clr(x = x, sigma = sigma, Yaxis = Yaxis)

  p <- sum(step1*deltaPairing(x = colSpace[,,2], sigma = sigma))
  q <- sum(step1*deltaPairing(x = colSpace[,,3], sigma = sigma))
  pcd <- colSpace[,,2]*p + colSpace[,,3]*q

  return(pcd)
}
```

A.2.6 Opponent Color Space

Description

Compute opponent color space of a color image and its Hue and Saturation.

Usage

```
YPQHS(x)
```

Arguments

x	an Image object or an array.

Details

In a first step YPQHS transforms the RGB color space into Opponent Color Space. It is made up of an achromatic luminance channel Y and two chromatic opponent color channels: the yellow-blue P and the red-green Q channels. In a second step Hue (H) and Saturation (S) are computed. The luminance channel Y provides the default color to grayscale image. It conforms to the NTSC-Rec.601 standard luminance axis.

Values

It returns an array with the third dimensions composed of Luminance, chromatic opponent color Yellow-Blue, chromatic opponent color Red-Green, Hue and Saturation.

Function

```
YPQHS <- function(x){

  if(!(is.Image(x) | is.array(x))) stop("x must be an Image object or an array")

  z <- apply(x,3,as.vector)
  space <- matrix(c(.2989,.5,1,.587,.5,-1,.114,-1,0),3,3)
  ypq <- space%*%t(z)
  Y <- ypq[1,]
  P <- ypq[2,]
  Q <- ypq[3,]
  H <- (1/pi)*(atan(Q/P))
  S <- sqrt(P^2 + Q^2)
  a <- array(data=c(Y,P,Q,H,S),c(dim(x)[1],dim(x)[2],5))

  return(a)
}
```

A.2.7 Local Adaptive Thresholding

Description

Compute local thresholds by Sauvola'smethod through an efficient implementation using integral images.

Usage

```
LAT(n, w = 30, k = 0.34)
```

Arguments

`n`	a matrix object. The input image.
`w`	a numeric or integer length-one object. The dimension of the window to define neighbors.
`k`	a numeric length-one object. It controls the value of the threshold in the local window.

Values

It returns a matrix object.

Function

```
LAT <- function(n, w = 30, k = 0.34){

  if(!(is.matrix(n))) stop("n must be a matrix object")
  if(!(is.numeric(w) | is.integer(w))) stop("w must be numeric or integer")
  if(length(w)!=1) stop("w must be one-length object")
  if(w < 2) stop("w must be >= 2")
  if(!(is.numeric(k))) stop("k must be numeric")
  if(length(k)!=1) stop("k must be one-length object")
  if(k < 0.2 | k > 0.5) stop("k must be >= 0.2 and <= 0.5")

  lsd <- lSd(n = n, w = w)
  tr <- lMean(n = n, w = w)*(1 + k*((lsd/max(lsd))-1))

  return(tr)
}
```

A.2.8 Contrast Loss Ratio

Description

Compute the Contrast Loss Ratio.

Usage

```
clr(x, sigma = 25, Yaxis = 0.6686)
```

Arguments

`x`	an Image object or an array.
`sigma`	a numeric or integer length-one vector.

`Yaxis` a numeric or integer length-one vector. It corresponds to the maximum length of Luminance axis.

Values

It returns an Image object or a matrix object.

Function

```
clr <- function(x, sigma = 25, Yaxis = 0.6686){

  if(!(is.Image(x) | is.array(x))) stop("x must be an Image object or an array")
  if(!(is.numeric(sigma) | is.integer(sigma))) stop(
    "sigma must be numeric or integer")
  if(length(sigma)!=1) stop("length(sigma) must be 1")

  d<-sqrt(deltaPairing(x=x[,,1],sigma=sigma)^2 +
            deltaPairing(x=x[,,2],sigma=sigma)^2 +
            deltaPairing(x=x[,,3],sigma=sigma)^2)
  c <- ifelse(d == 0, 0, (1 - ((1/Yaxis)*abs(
    deltaPairing(x = YPQHS(x)[,,1],sigma = sigma)))/d))
  c[which(c<0)]<-0

  return(c)
}
```

A.2.9 Decolorize

Description

Convert an RGB image into a grayscale image carrying out the Decolorize algorithm.

Usage

```
decolorize(x, sigma = 25, Yaxis = 0.6686, lambda = 0.3, eta = 0.001)
```

Arguments

`x` an Image object or an array.

`sigma` a numeric or integer length-one vector. It is the typical size of relevant image features in pixels.

`Yaxis` a numeric or integer length-one vector. It corresponds to the maximum length of Luminance axis.

`lambda` it represents the degree of image enhancement.

`eta` a numeric length-one vector. It represents the lower or upper quantiles (assumed to be outliers) to remove.

Values

It returns an Image object or a matrix object.

Function

```
decolorize <- function(x, sigma = 25, Yaxis = 0.6686, lambda = 0.3, eta = 0.001){

  if(!(is.Image(x) | is.array(x))) stop("x must be an Image object or an array")
  if(!(is.numeric(sigma) | is.integer(sigma))) stop(
    "sigma must be numeric or integer")
  if(length(sigma)!=1) stop("length(sigma) must be 1")
  if(eta<0 | eta>=0.5) stop("eta must be >= 0 and < 0.5")
  if(lambda<0 | lambda>1) stop("lambda must be >=0 and <= 1")

  maxY<-1
  maxS<-1.1180339887498948482
  alter<-lambda*(maxY/maxS)

  colSpace<-YPQHS(x)
  Y <- colSpace[,,1]
  S <- colSpace[,,5]
  U <- Y + lambda*pcc(x, sigma = sigma, Yaxis = Yaxis, eta = eta)

  Umin <- quantile(U, prob = eta, na.rm = TRUE)
  Umax <- quantile(U, prob = (1 - eta), na.rm = TRUE)

  Vmin <- lambda*0 + (1 - lambda)*quantile(Y, prob = eta, na.rm = TRUE)
  Vmax <- lambda*maxY + (1 - lambda)*quantile(Y, prob = (1 - eta), na.rm = TRUE)

  V <- Vmin + ((Vmax - Vmin)/(Umax - Umin))*(U - Umin)
  E <- ifelse(Y - alter*S > 0, Y - alter*S, 0)
  Fi <- ifelse(Y + alter*S < maxY, Y + alter*S, maxY)

  VEFi <- array(data = c(V,E,Fi), c(dim(x)[1], dim(x)[2], 3))

  gsT <- apply(VEFi, c(1,2), median3)

  return(gsT)
}
```

A.2.10 Extraction of information

Description

Compute for each object of a labeled binary image outline coordinates, landmarks, color intensity values, size statistics and Haralick's features.

Usage

```
extinfo(objs, image, nldks, int = 0)
```

Arguments

objs	a matrix of labeled objects.
image	an Image object or an array.
nldks	a numeric or integer vector. Number of landmarks to consider.
int	a numeric or integer one-length object. Number of point to ignore around where diameter starts.

Values

outline	a list of matrices of outline coordinates.
landmarks	a list of arrays of landmarks coordinates.
genStat	a matrix with size and Haralick's features.
colValue	a list with color intensities for each object.

Function

```
extinfo <- function(objs, image, nldks, int = 0){

  if(!is.matrix(objs)) stop("objs must be a matrix")
  if(!(is.Image(image) | is.array(image))) stop(
    "image must be an Image object or an array")
  for(i in c("nldks","int")){
    if(!(is.numeric(get(i)) | is.integer(get(i)))) stop(
      paste0(i, " must be numeric or integer"))
  }
  if(length(int)!=1) stop("int must be of length 1")
  if(int < 0) stop("int must be >= 0")
  if(min(nldks) < 2) stop("each element of nldks must be >= 2")

  nldks <- as.integer(nldks)

  meanHaralick <- computeFeatures.haralick(objs,image)
  genStat <- cbind(computeFeatures.shape(objs), meanHaralick)

  mat <- imageData(objs)
  outl <- vector("list",max(objs))
  cv <- vector("list",max(objs))
  ldks <- list()
  for(j in 1:length(nldks)){ldks[[j]] <- array(
```

```
    dim = c(nldks[j], 2, max(objs)))}
  names(ldks) <- paste0("ldks",nldks)

  for(i in 1:max(objs)){
    row <- which(apply(mat, 1, function(x) max(which(x==i)))>0)
    col <- which(apply(mat, 2, function(x) max(which(x==i)))>0)
    seed <- mat[row,col]
    seed[seed!=i] <- 0
    coo <- ocontour(seed)
    coo <- do.call(rbind,coo)
    outl[[i]] <- coo

    cv[[i]] <- colorsValue(imageData(image[row,col,]),objs[row,col])

    for(k in 1:length(nldks)){
      ldks[[k]][,,i] <- landmarks(coo,nldks[k],int)
    }
  }

  return(list(outline = outl,
              landmarks = ldks,
              genStat = genStat,
              colValue = cv
  ))
}
```

A.2.11 Binary

Description

Compute the binary image, where 0 corresponds to background and 1 to foreground, using the background subtraction approach.

Usage

```
binary(image1, image2, w = 100)
```

Arguments

`image1`	an Image object or an array.
`image2`	an Image object or an array.
`w`	a numeric or integer length-one object. The dimension of the window to define neighbors.

Values

`objs` a matrix where objects are labeled with a different integer number.

`image1` the same `image1` object of input.

`image2` the same `image2` object of input.

Function

```
binary <- function(image1, image2, w = 100){

  for(i in c("image1","image2")){
    if(!(is.Image(get(i)) | is.array(get(i)))) stop(
      paste0(i, " must be an Image object or an array"))
  }
  if(!(is.numeric(w) | is.integer(w))) stop("w must be numeric or integer")
  if(length(w)!=1) stop("w must be one-length object")
  if(w < 2) stop("w must be >= 2")

  subt <- abs(image1 - image2)
  imageG <- decolorize(subt)
  thresholds <- LAT(imageG, w = w)
  binary <- ifelse(imageG < thresholds, 1, 0)

  filled <- fillHull(binary)
  kern <- makeBrush(size = 5, sigma = 0.3, shape = "disc", angle = 45)
  op <- opening(filled, kern)
  nseg <- bwlabel(op)

  return(list(objs = nseg,
              image1 = image1,
              image2 = image2
  ))
}
```

A.3 Statistical classification

A.3.1 Model accuracy

Description

Compute the accuracy of models according to `groups` settings for all dataset in `test.com`.

Usage

```
accuracy(test.com, mod_list, groups, pos.class, rows = NULL)
```

Arguments

`test.com` a list of datasets to test the models.

`mod_list` a list of models defined.

`groups` a list of all data types to model with the correspondent position of the independent variables in dataset and the label of classifier to perform.

`pos.class` a numeric or integer one-length object. It indicates the position of the specimen variable.

`rows` a numeric vector for selecting specific observations to test. if `NULL` all observations are used.

Values

A data.frame with the accuracy rate for each model.

Details

`groups` argument is defined as follow

```
groups = list(list(all = x.position, methods= c(classifier_list)))
```

where `all` is the label of data type, `x.position` the position the independent variables in dataset and `classifier_list` is a character vector of classifier (e.g. `c("knn","lda")`).

Function

```
accuracy <- function(test.com, mod_list, groups, pos.class, rows = NULL){

  if(!(is.numeric(pos.class) | is.integer(pos.class))) stop(
    "pos.class must be numeric or integer")
  if(length(pos.class)!=1) stop("pos.class must be of length 1")
  if(!(pos.class %in% 1:dim(test.com[[1]])[2])) stop(
    paste0("pos.class must be from 1 to ", dim(test.com[[1]])[2]))
  for(i in c("groups","test.com","mod_list")){
    if(!(is.list(get(i)))) stop(paste0(i, " must be a list"))
  }

  accuracy_table <- data.frame(classe1 = NA_character_, classe2 = NA_character_,
                    matrix(nrow = length(test.com),
                    ncol= sum(sapply(groups, FUN=function(X) length(X[[2]])))))
  accuracy_table[,1:2] <- apply(accuracy_table[,1:2],2,as.character)
  names(accuracy_table) <- c("classe1","classe2",
              paste(rep(sapply(groups, FUN = function(X) names(X[1])),
                        sapply(groups, FUN=function(X) length(X[[2]]))),
                        as.vector(sapply(groups, FUN = function(X) X[[2]])),
                    sep = "_"))
```

```
  l.te <- length(test.com)
  l.db <- length(groups)
  for(k in 1:l.te){
    if(is.null(rows)){n.rows <- 1:dim(test.com[[k]])[1]} else {n.rows <- rows}
    for(i in 1:l.db){
      l.cl <- length(groups[[i]][[2]])
      for(j in 1:l.cl){
        dati.test <- test.com[[k]][n.rows,c(pos.class,groups[[i]][[1]])]

        pred <- switch(groups[[i]][[2]][j],
        rf = predict(mod_list[[Ind(mod_list, names(groups[[i]][1]),
             groups[[i]][[2]][j], levels(dati.test[,1]))]][[4]],
             newdata=dati.test, type = "class"),
        rpart = predict(mod_list[[Ind(mod_list, names(groups[[i]][1]),
             groups[[i]][[2]][j],levels(dati.test[,1]))]][[4]],
             newdata=dati.test, type = "class"),
        svm = predict(mod_list[[Ind(mod_list, names(groups[[i]][1]),
             groups[[i]][[2]][j],levels(dati.test[,1]))]][[4]],
             newdata=dati.test),
        lda = predict(mod_list[[Ind(mod_list, names(groups[[i]][1]),
             groups[[i]][[2]][j],levels(dati.test[,1]))]][[4]],
             newdata=dati.test)$class,
        nb = predict(mod_list[[Ind(mod_list, names(groups[[i]][1]),
             groups[[i]][[2]][j],levels(dati.test[,1]))]][[4]],
             newdata=dati.test, type = "class")
        )
        accuracy_table[k,(j+2+(i-1)*l.cl)] <- confusionMatrix(pred,
                                              dati.test[,1])$overall[1]
        accuracy_table[k,1:2] <- levels(dati.test[,1])
      }
    }
  }
  return(accuracy_table)
}
```

A.3.2 Classification

Description

Compute the probability that some of models according to `groups` settings for all dataset in `test.com`.

Usage

```
classification(data, dichotomy, mod_list, groups, pos.class)
```

Arguments

`data` a data.frame with observations to classify.

`dichotomy` a character vector with the two classes on which to classify.

`mod_list` a list of models defined.

`groups` a list of all data types to model with the correspondent position of the independent variables in dataset and the label of classifier to perform.

`pos.class` a numeric or integer one-length object. It indicates the position of the specimen variable.

Values

A data.frame with the probability to belong to one of `dichotomy` class.

Function

```
classification <- function(data, dichotomy, mod_list, groups, pos.class){

  if(!is.data.frame(data)) stop("data must be a data.frame")
  if(!(is.numeric(pos.class) | is.integer(pos.class))) stop(
    "pos.class must be numeric or integer")
  if(length(pos.class)!=1) stop("pos.class must be of length 1")
  if(!(pos.class %in% 1:dim(data)[2])) stop(
    paste0("pos.class must be from 1 to ", dim(data)[2]))
  for(i in c("groups","mod_list")){
    if(!(is.list(get(i)))) stop(paste0(i, " must be a list"))
  }

  accuracy_table <- data.frame(classe = NA_character_,
              matrix(nrow = 2, ncol= sum(sapply(groups,
              FUN=function(X) length(X[[2]])))))
  accuracy_table[,1] <- as.character(accuracy_table[,1])
  names(accuracy_table) <- c("classe", paste(rep(sapply(
    groups, FUN=function(X) names(X[1])),sapply(
      groups, FUN=function(X) length(X[[2]]))), as.vector(sapply(
        groups, FUN=function(X) X[[2]])), sep = "_"))
  accuracy_table[,1] <- dichotomy
  l.db <- length(groups)
  for(i in 1:l.db){
    l.cl <- length(groups[[i]][[2]])
    for(j in 1:l.cl){
      dati.val <- data

      pred <- switch(groups[[i]][[2]][j],
          rf = predict(mod_list[[Ind(mod_list, names(groups[[i]][1]),
```

```
              groups[[i]][[2]][j],dichotomy)[1]]][[4]],
              newdata = dati.val, type = "prob"),
          rpart = predict(mod_list[[Ind(mod_list, names(groups[[i]][1]),
              groups[[i]][[2]][j],dichotomy)[1]]][[4]],
              newdata = dati.val, type = "prob"),
          svm = attr(predict(mod_list[[Ind(mod_list, names(groups[[i]][1]),
              groups[[i]][[2]][j],dichotomy)[1]]][[4]],
              newdata = dati.val,probability = TRUE), "probabilities"),
          lda = predict(mod_list[[Ind(mod_list, names(groups[[i]][1]),
              groups[[i]][[2]][j],dichotomy)[1]]][[4]],
              newdata = dati.val)$posterior,
          nb = predict(mod_list[[Ind(mod_list, names(groups[[i]][1]),
              groups[[i]][[2]][j],dichotomy)[1]]][[4]],
              newdata = dati.val, type = "raw")
      )
      accuracy_table[1,(j+1+(i-1)*l.cl)] <- apply(pred,2,
        function(x) mean(x,na.rm = TRUE))[which(colnames(pred)==dichotomy[1])]
      accuracy_table[2,(j+1+(i-1)*l.cl)] <- apply(pred,2,
        function(x) mean(x,na.rm = TRUE))[which(colnames(pred)==dichotomy[2])]
    }
  }
  return(accuracy_table)
}
```

A.3.3 Model position

Description

It is a particular subsetting function created ad hoc for `accuracy` function. It allows to subset a specific model in a list of models.

Usage

```
Ind(mod_list, data_type, classifier, classes)
```

Arguments

`mod_list`	a list of models.
`data_type`	a character one-length vector. It refers to the type of data sought.
`classifier`	a character one-length vector. It refers to the classifier sought.
`classes`	a character two-length vector. It refers to the two classes sought.

Values

A numeric value indicating the position of the model sought in the list.

Function

```
Ind <- function(mod_list, data_type, classifier, classes){

  Ind_data_type <- seq_along(mod_list)[sapply(
    mod_list, FUN = function(X) data_type %in% X[[1]])]
  Ind_classifier <- seq_along(mod_list)[sapply(
    mod_list, FUN = function(X) classifier %in% X[[2]])]
  Ind_classes1 <- seq_along(mod_list)[sapply(
    mod_list, FUN = function(X) classes[1] %in% X[[3]])]
  Ind_classes2 <- seq_along(mod_list)[sapply(
    mod_list, FUN = function(X) classes[2] %in% X[[3]])]

  ind <- Ind_data_type[which(Ind_data_type %in% Ind_classifier &
    Ind_data_type %in% Ind_classes1 & Ind_data_type %in% Ind_classes2)]

  return(ind)
}
```

A.3.4 Model list

Description

Define all combination of models according to `groups` settings for each dataset in `train.com`.

Usage

```
models(train.com, prev_mod, groups, pos.class)
```

Arguments

`train.com`	a list of datasets.
`prev_mod`	a list of models already defined and to not define again.
`groups`	a list of all data types to model with the correspondent position of the independent variables in dataset and the label of classifier to perform.
`pos.class`	a numeric or integer one-length object. It indicates the position of the specimen variable.

Values

A list of all model defined.

Details

`groups` argument is defined as follow

```
groups = list(list(all = 2:3, methods= c(classifier_list)))
```

where `all` is the label of data type, `2:3` the position the independent variables in dataset and `classifier_list` is a character vector of classifier (e.g. `c("knn","lda")`).

Function

```
models <- function(train.com, prev_mod, groups, pos.class){

  if(!is.list(train.com)) stop("train.com must be a list")
  if(!(is.numeric(pos.class) | is.integer(pos.class))) stop(
    "pos.class must be numeric or integer")
  if(length(pos.class)!=1) stop("pos.class must be of length 1")
  if(!(pos.class %in% 1:dim(train.com[[1]])[2])) stop(paste0(
    "pos.class must be from 1 to ",dim(train.com[[1]])[2]))
  if(!is.list(groups)) stop("groups must be a list")

  old <- unique(paste(sapply(prev_mod, FUN=function(X) X[[3]][1]),
                      sapply(prev_mod, FUN=function(X) X[[3]][2]),
                      sep="_vs_"))

  if(length(which(names(train.com) %in% old))!=0){
    train.com <- train.com[-which(names(train.com) %in% old)]
  }

  l.tr <- length(train.com)
  l.db <- length(groups)
  for(k in 1:l.tr){
    for(i in 1:l.db){
      l.cl <- length(groups[[i]][[2]])
      for(j in 1:l.cl){
        dati.train <- train.com[[k]][,c(pos.class,groups[[i]][[1]])]
        prev_mod <- c(list(list(
          data_type = names(groups[[i]][1]),
          classifier = groups[[i]][[2]][j],
          classes = levels(dati.train[,1]),
          model = switch(groups[[i]][[2]][j],
                         rf = randomForest(as.formula(paste0("dati.train$",
                              names(dati.train)[1],"~.")),data = dati.train),
                         rpart = rpart(as.formula(paste0("dati.train$",
                              names(dati.train)[1],"~.")),data = dati.train),
                         svm = svm(as.formula(paste0(names(dati.train)[1],"~.")),
                               data = dati.train,probability = TRUE),
                         lda = MASS::lda(as.formula(paste0("dati.train$",
                               names(dati.train)[1],"~.")),data = dati.train),
                         nb = naiveBayes(as.formula(paste0("dati.train$",
                              names(dati.train)[1],"~.")),data = dati.train))
        )
```

```
        ),
        prev_mod)
      }
    }
  }
  return(prev_mod)
}
```

A.3.5 Dendrogram

Description

Create a `hclust` class object from the list of aggregating steps obtained from `treeModel`.

Usage

```
toTree(tree, height = NULL, order = NULL)
```

Arguments

`tree`	a character vector with the aggregating steps obtained from
`height`	the dendrogram heights.
`order`	a vector giving the permutation of the original observations suitable for plotting. `treeModel`.

Values

A `hclust` class object.

Function

```
toTree <- function(tree, height = NULL, order = NULL){

  if(is.null(height)){height <- 1:length(tree)}
  if(is.null(order)){order <- 1:(length(tree)+1)}

  m <- matrix(ncol = 2, nrow = length(tree))
  aggr <- unlist(sapply(strsplit(tree,"__"), function(X) paste(
    X,collapse = "_")))

  classes <- unlist(strsplit(aggr[length(tree)],"_"))

  for(i in 1:length(tree)){
    cl1 <- strsplit(tree[i], "__")[[1]][1]
    cl2 <- strsplit(tree[i], "__")[[1]][2]

    x1 <- which(classes == cl1)
```

```
    x2 <- which(classes == cl2)

    if(length(x1)!=0){m[i,1] <- -x1} else {m[i,1] <- which(cl1 == aggr)}
    if(length(x2)!=0){m[i,2] <- -x2} else {m[i,2] <- which(cl2 == aggr)}
  }

  dendrogram <- list()
  dendrogram$merge <- m
  dendrogram$height <- height
  dendrogram$order <- order
  dendrogram$labels <- classes
  class(dendrogram) <- "hclust"

  return(dendrogram)
}
```

A.3.6 Tree Combination Model

Description

Define the model of the tree approach of classifier combination. See Section 3.1 for details.

Usage

```
treeModel(data, groups, pos.class, criteria = c("max", "average"), acc = c("resub",
"test", "val"), val = 0.2, xi = NULL, lambda = 0.5, seed = 123)
```

Arguments

`data`	a data.frame for creating the model.
`groups`	a list of all data types to model with the correspondent position of the independent variables in dataset and the label of classifier to perform.
`pos.class`	a numeric or integer one-length object. It indicates the position of the specimen variable.
`criteria`	the classifier combination criteria.
`acc`	the accuracy computation criteria.
`val`	the proportion of data left for validation process.
`xi`	the proportion of training data used for defining the model (it used only with `acc = "test"`)
`lambda`	the threshold of accuracy that classifiers must reach to be selected for combination process.
`seed`	the argument of `set.seed` for specifying seeds.

Values

`models` a list of all models defined.

`accuracy` a list of accuracy of all models.

`weights` a list with the accuracy that defined the merger between classes for each step.

`tree` a character vector that summarizes the classes opposition for each step.

`groups` the same of input argument.

`parameters` a vector with the numerical input arguments (`lambda`, `pos.class`, `xi`, `val` and `seed`)

`criteria` a character vector with input criteria arguments (`combCriteria` and `acc`)

`dataset` a list with the three datasets came from partition (`training`, `testing` and `validating`)

Details

`groups` argument is defined as follow

```
groups = list(list(all = x.position, methods= c(classifier_list)))
```

where `all` is the label of data type, `x.position` the position the independent variables in dataset and `classifier_list` is a character vector of classifier (e.g. `c("knn","lda")`).

Function

```
treeModel <- function(data, groups, pos.class, criteria = c("max", "average"),
                      acc = c("resub","test","val"), val = 0.2, xi = NULL,
                      lambda = 0.5, seed = 123){

  if(!is.data.frame(data)) stop("data must a data.frame")
  if(!(is.numeric(pos.class) | is.integer(pos.class))) stop(
    "pos.class must be numeric or integer")
  if(length(pos.class)!=1) stop("pos.class must be of length 1")
  if(!(pos.class %in% 1:dim(data)[2])) stop(
    paste0("pos.class must be from 1 to ", dim(data)[2]))
  if(!is.list(groups)) stop("groups must a list")
  if(!is.numeric(val)) stop("val must be numeric")
  if(!(val > 0 & val < 1)) stop("pos.class must be from 0 to 1")

  CRITERIA <- c("max", "average")
  criteria <- tryCatch(match.arg(criteria, CRITERIA), error = function(e) NULL)
  if(is.null(criteria)) stop(paste0("Error: criteria should be one of ",
                                    paste0(CRITERIA,collapse = " or ")))

  ACC <- c("resub","test","val")
  acc <- tryCatch(match.arg(acc, ACC), error = function(e) NULL)
```

```
if(is.null(acc)) stop(paste0("Error: acc should be one of ",
                             paste0(ACC,collapse = " or ")))

pos.class <- as.integer(pos.class)

set.seed(seed)
r.val <- createDataPartition(y = data[, pos.class], p = (1-val), list = FALSE)
validation <- data[-r.val,]
mod <- data[r.val,]

if(acc == "test"){
  if(!is.numeric(xi)) stop("xi must be numeric")
  if(!(xi > 0 & xi < 1)) stop("pos.class must be from 0 to 1")
  set.seed(seed)
  inTrain <- createDataPartition(y = mod[,pos.class], p = xi, list = FALSE)
}

train <- switch(acc,
                resub = mod,
                test = mod[inTrain,],
                val = mod)
test <- switch(acc,
               resub = mod,
               test = mod[-inTrain,],
               val = validation)

leng <- (length(levels(data[,pos.class]))-1)
tree <- rep(NA_character_,leng)
allmodels <- list()
allaccuracy <- list()
height <- rep(NA_real_,leng)
weight <- list()

for(i in 1:leng){
  train.com <- U(train,pos.class)
  test.com <- U(test,pos.class)

  allmodels <- models(train.com, allmodels, groups, pos.class)
  allaccuracy[[i]] <- a <- accuracy(test.com, allmodels, groups, pos.class)
  names(allaccuracy)[[i]] <- paste0("step",i)


  if(criteria == "average"){
    if(!is.numeric(lambda)) stop("lambda must be numeric")
    if(!(lambda > 0 & lambda < 1)) stop("pos.class must be from 0 to 1")
    for(k in seq_along(a)[-1]){set(a, i=which(a[,k]<lambda), j=k, value=NA)}
```

```
    means <- apply(data.frame(a[,-c(1:2)]),1,mean,na.rm=TRUE)
  }

  if(criteria == "max"){
    means <- apply(data.frame(a[,-c(1:2)]),1,max,na.rm=TRUE)
  }

  weight[[i]] <- allaccuracy[[i]][which.min(means),]
  height[i] <- means[which.min(means)]
  classes <- as.character(a[which.min(means),1:2])

  tree[i] <- paste(classes,collapse = "__")
  names(tree)[i] <- paste0("step",i)

  set(train, i = which(train[,pos.class] == classes[1] |
                         train[,pos.class] == classes[2]),
      j = pos.class, value = paste0(classes[1],"_",classes[2]))
  set(test, i = which(test[,pos.class] == classes[1] |
                        test[,pos.class] == classes[2]),
      j = pos.class, value = paste0(classes[1], "_", classes[2]))
  train[,pos.class] <- factor(train[,pos.class])
  test[,pos.class] <- factor(test[,pos.class])
}

mod <- data[r.val,]
validation <- data[-r.val,]

train <- switch(acc,
                resub = mod,
                test = mod[inTrain,],
                val = mod)
test <- switch(acc,
               resub = mod,
               test = mod[-inTrain,],
               val = validation)

return(list(
  models = allmodels,
  accuracy = allaccuracy,
  weights = weight,
  tree = tree,
  groups = groups,
  heights = height,
  parameters = c(lambda = lambda,
                 pos.class = pos.class,
                 xi = xi,
```

```
                  val = val,
                  seed = seed),
    criteria = c(criteria = criteria,
                 acc = acc),
    dataset = list(training = train,
                  testing = test,
                  validating = validation)
  ))
}
```

A.3.7 Tree validation

Description

Define the model of the tree approach of classifier combination. See Section 3.1.2 for details.

Usage

```
validation(output, newdata, criteria = c("max", "average"))
```

Arguments

`output`	the output of `treeModel`.
`newdata`	a data.frame to use as model validation.
`criteria`	the classifier combination criteria.

Values

A character vector of the predicted classes.

Function

```
validation <- function(output, newdata, criteria = c("max", "average")){

  if(!is.list(output)) stop("output must be a list")
  if(!is.data.frame(newdata)) stop("newdata must be a data.frame")
  CRITERIA <- c("max", "average")
  criteria <- tryCatch(match.arg(criteria, CRITERIA), error = function(e) NULL)
  if(is.null(criteria)) stop(paste0("Error: criteria should be one of ",
                                    paste0(CRITERIA,collapse = " or ")))

  tree <- output$tree
  groups <- output$groups
  allmodels <- output$models
  allaccuracy <- output$accuracy
  wth <- output$weights
  lambda <- output$parameters[1]
```

```
pos.class <- output$parameters[2]

tree2 <- lapply(as.list(tree), function(X) gsub("__", "_", X))

predicted <- character()

col <- c()
for(m in 1:length(groups)){
  col <- c(col,apply(expand.grid(names(groups[[m]][1]), unlist(groups[[m]][2])),
  1, paste, collapse="_"))
}

for(k in 1:dim(newdata)[1]){
  #  for(j in levels(newdata[,pos.class])){
  pos.tree <- length(tree)
  val.com <- newdata[k,]
  superclass = 1

  while(superclass != 0){
    dicho <- unlist(strsplit(tree[pos.tree],"__"))

    clsf <- classification(val.com,dicho,allmodels,groups,pos.class)

    if(criteria == "average"){
      wth <- data.frame(allaccuracy[[pos.tree]][which(tree[[pos.tree]]==
             apply(allaccuracy[[pos.tree]][,1:2],1,function(X) paste(
               X,collapse = "__"))),col])
      wth <- wth-lambda
      wth[is.na(wth)] <- 0
      wth <- wth/sum(wth)

      chosen_class <- if(sum(clsf[1,-1]*wth)>sum(clsf[2,-1]*wth)){
        clsf[1,1]} else {clsf[2,1]}
    }

    if(criteria == "max"){

      wth <- rep(0,length(col))
      wth[which.max(data.frame(
        allaccuracy[[pos.tree]][which(tree[[pos.tree]] ==
        apply(allaccuracy[[pos.tree]][,1:2],1,
              function(X) paste(X,collapse = "__"))),col]))] <- 1

      chosen_class <- if(sum(clsf[1,-1]*wth) > sum(clsf[2,-1]*wth)){
        clsf[1,1]} else {clsf[2,1]}
    }
```

```
      pos.tree <- which(tree2 == chosen_class)
      superclass <- strcount(chosen_class, "_", "")
    }
    predicted <- c(predicted,chosen_class)
  }
  return(predicted)
}
```

Bibliography

Abramson, N. (1963). *Information theory and coding*. McGraw Hill.

Bacchetta, G., Fenu, G., Mattana, E., Piotto, B., and Virevaire, M. (2006). *Manuale per la raccolta, studio, conservazione e gestione ex situ del germoplasma*, volume 37. APAT-Agenzia per la protezione dell'ambiente e per i servizi tecnici.

Bacchetta, G., Grillo, O., Mattana, E., and Venora, G. (2008). Morpho-colorimetric characterization by image analysis to identify diaspores of wild plant species. *Flora-Morphology, Distribution, Functional Ecology of Plants*, 203(8):669–682.

Badekas, E. and Papamarkos, N. (2005). Automatic evaluation of document binarization results. In *Progress in pattern recognition, image analysis and applications*, pages 1005–1014. Springer.

Bala, R. and Eschbach, R. (2004). Spatial color-to-grayscale transform preserving chrominance edge information. In *Color and Imaging Conference*, volume 2004, pages 82–86. Society for Imaging Science and Technology.

Bonhomme, V., Picq, S., Gaucherel, C., and Claude, J. (2014). Momocs: outline analysis using r. *Journal of Statistical Software*, 56(13):1–24.

Bookstein, F. L. (1984). A statistical method for biological shape comparisons. *Journal of Theoretical Biology*, 107(3):475–520.

Bookstein, F. L. (1986). Size and shape spaces for landmark data in two dimensions. *Statistical Science*, pages 181–222.

Bookstein, F. L. (1997). *Morphometric tools for landmark data: geometry and biology*. Cambridge University Press.

Burrascano, S., Rosati, L., and Blasi, C. (2009). Plant species diversity in mediterranean old-growth forests: a case study from central italy. *Plant Biosystems*, 143(1):190–200.

Čadík, M. (2008). Perceptual evaluation of color-to-grayscale image conversions. In *Computer Graphics Forum*, volume 27, pages 1745–1754. Wiley Online Library.

Chambers, J. M. (1998). *Programming with data: A guide to the S language*. Springer Science & Business Media.

Chan, T. and Shen, J. (2005). *Image processing and analysis: variational, PDE, wavelet, and stochastic methods*. Siam.

Chen, C., Chiang, Y., and Pomeranz, Y. (1989). Image analysis and characterization of cereal grains with a laser range finder and camera contour extractor. *Cereal Chem*, 66(6):466–470.

Chtioui, Y., Bertrand, D., Dattée, Y., and Devaux, M.-F. (1996). Identification of seeds by colour imaging: Comparison of discriminant analysis and artificial neural network. *Journal of the Science of Food and Agriculture*, 71(4):433–441.

Claude, J. (2008). *Morphometrics with R*. Springer.

Clemen, R. T. and Winkler, R. L. (1999). Combining probability distributions from experts in risk analysis. *Risk analysis*, 19(2):187–203.

Cooke, R. M. (1991). *Experts in uncertainty: opinion and subjective probability in science*. Oxford University Press.

Crampton, J. S. (1995). Elliptic fourier shape analysis of fossil bivalves: some practical considerations. *Lethaia*, 28(2):179–186.

Cribari-Neto, F. and Zeileis, A. (2009). Beta regression in r.

David, H. A. (1963). *The method of paired comparisons*, volume 12. DTIC Document.

Dryden, I. L. and Mardia, K. V. (1998). *Statistical shape analysis*, volume 4. John Wiley & Sons New York.

Fairchild, M. D. (2013). *Color appearance models*. John Wiley & Sons.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37.

Ferrari, S. and Cribari-Neto, F. (2004). Beta regression for modelling rates and proportions. *Journal of Applied Statistics*, 31(7):799–815.

Ferson, S., Rohlf, F. J., and Koehn, R. K. (1985). Measuring shape variation of two-dimensional outlines. *Systematic Biology*, 34(1):59–68.

Foley, J. D., Van Dam, A., et al. (1982). *Fundamentals of interactive computer graphics*, volume 2. Addison-Wesley Reading, MA.

Frawley, W. J., Piatetsky-Shapiro, G., and Matheus, C. J. (1992). Knowledge discovery in databases: An overview. *AI magazine*, 13(3):57.

Freitas, C. O., De Carvalho, J. M., Oliveira Jr, J., Aires, S. B., and Sabourin, R. (2007). Confusion matrix disagreement for multiple classifiers. In *Progress in Pattern Recognition, Image Analysis and Applications*, pages 387–396. Springer.

Fu, K.-S. and Mui, J. (1981). A survey on image segmentation. *Pattern recognition*, 13(1):3–16.

Giardina, C. R. and Kuhl, F. P. (1977). Accuracy of curve approximation by harmonically related vectors with elliptical loci. *Computer Graphics and Image Processing*, 6(3):277–285.

Glasbey, C. A. and Horgan, G. W. (1995). *Image analysis for the biological sciences*, volume 1. Wiley Chichester.

Gooch, A. A., Olsen, S. C., Tumblin, J., and Gooch, B. (2005). Color2gray: salience-preserving color removal. *ACM Transactions on Graphics (TOG)*, 24(3):634–639.

Granitto, P. M., Verdes, P. F., and Ceccatto, H. A. (2005). Large-scale investigation of weed seed identification by machine vision. *Computers and Electronics in Agriculture*, 47(1):15–24.

Grillo, O., Mattana, E., Venora, G., and Bacchetta, G. (2010). Statistical seed classifiers of 10 plant families representative of the mediterranean vascular flora. *Seed Science and Technology*, 38(2):455–476.

Grundland, M. and Dodgson, N. A. (2005). The decolorize algorithm for contrast enhancing, color to grayscale conversion. *Technical Report—University of Cambridge. Computer Laboratory, XX, XX, vol. UCAM-CL-TR-649*, pages 1–15.

Guarino, L., Rao, V. R., Reid, R., et al. (1995). *Collecting plant genetic diversity: technical guidelines.* CAB international.

Hand, D. J. and Till, R. J. (2001). A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45(2):171–186.

Haralick, R. M., Shanmugam, K., and Dinstein, I. H. (1973). Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, (6):610–621.

Harper, J. L., Lovell, P., and Moore, K. (1970). The shapes and sizes of seeds. *Annual review of ecology and systematics*, 1:327–356.

Hunt, R. W. G. (2005). *The reproduction of colour*. John Wiley & Sons.

Iriondo, J. M. and Perez, C. (1999). Propagation from seeds and seed preservation. *A Colour Atlas of Plant Propagation and Conservation. Manson Publishing, London*, pages 46–57.

Keefe, P. and Draper, S. (1986). The measurement of new characters for cultivar identification in wheat using machine vision. *Seed science and technology*, 14(3):715–724.

Kendall, D. G. (1977). The diffusion of shape. *Advances in applied probability*, 9(3):428–430.

Kuhl, F. P. and Giardina, C. R. (1982). Elliptic fourier features of a closed contour. *Computer graphics and image processing*, 18(3):236–258.

Labatut, V. and Cherifi, H. (2012). Accuracy measures for the comparison of classifiers. *arXiv preprint arXiv:1207.3790*.

Laliberté, B. (1997). Botanic garden seed banks/genebanks worldwide, their facilities, collections and network. *Botanic Gardens Conservation News*, 2(9):18–23.

Li, D.-Z. and Pritchard, H. W. (2009). The science and economics of ex situ plant conservation. *Trends in plant science*, 14(11):614–621.

Lichman, M. (2013). Uci machine learning repository. *University of California, Irvine, School of Information and Computer Sciences*.

Mantiuk, R., Myszkowski, K., and Seidel, H.-P. (2006). A perceptual framework for contrast processing of high dynamic range images. *ACM Transactions on Applied Perception (TAP)*, 3(3):286–308.

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1980). Multivariate analysis.

Mattana, E., Fenu, G., and Bacchetta, G. (2005). La banca del germoplasma della sardegna (bg-sar): uno strumento per la conservazione del germoplasma autoctono sardo. *Inform. Bot. Ital*, 37(1):144–145.

Miller, J. S., Funk, V. A., Wagner, W. L., Barrie, F., Hoch, P. C., and Herendeen, P. (2011). Outcomes of the 2011 botanical nomenclature section at the xviii international botanical congress. *PhytoKeys*, (5):1.

Mola, F. and Siciliano, R. (1997). A fast splitting procedure for classification trees. *Statistics and Computing*, 7(3):209–216.

Neumann, L., Čadík, M., and Nemcsics, A. (2007). An efficient perception-based adaptive color to gray transformation. In *Proceedings of the Third Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging*, pages 73–80. Eurographics Association.

Otsu, N. (1975). A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27.

Pau, G., Fuchs, F., Sklyar, O., Boutros, M., and Huber, W. (2010). Ebimage—an r package for image processing with applications to cellular phenotypes. *Bioinformatics*, 26(7):979–981.

Petersen, P. and Krutz, G. (1992). Automatic identification of weed seeds by colour machine vision. In *Proceedings of the International Seed Testing Association*, volume 20.

Piccardi, M. (2004). Background subtraction techniques: a review. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 4, pages 3099–3104. IEEE.

Pimm, S. L., Russell, G. J., Gittleman, J. L., and Brooks, T. M. (1995). The future of biodiversity. *SCIENCE-NEW YORK THEN WASHINGTON-*, pages 347–347.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rachev, S. T. (1985). The monge-kantorovich mass transference problem and its stochastic applications. *Theory of Probability & Its Applications*, 29(4):647–676.

Rasche, K., Geist, R., and Westall, J. (2005). Re-coloring images for gamuts of lower dimension. In *Computer Graphics Forum*, volume 24, pages 423–432. Wiley Online Library.

Rohlf, F. J. and Archie, J. W. (1984). A comparison of fourier methods for the description of wing shape in mosquitoes (diptera: Culicidae). *Systematic Biology*, 33(3):302–317.

Sauvola, J. and Pietikäinen, M. (2000). Adaptive document image binarization. *Pattern recognition*, 33(2):225–236.

Serra, J. (1982). *Image analysis and mathematical morphology*. London.: Academic Press.[Review by Fensen, EB in: J. Microsc. 131 (1983) 258.] Mathematics, Review article General article, Technique Staining Microscopy, Cell size (PMBD, 185707888).

Shafait, F., Keysers, D., and Breuel, T. M. (2008). Efficient implementation of local adaptive thresholding techniques using integral images. In *Electronic Imaging 2008*, pages 681510–681510. International Society for Optics and Photonics.

Shapiro, L. and Stockman, G. (2001). Computer vision. chapter 12. *New Jersey: Prentice-Hall*.

Sindhwani, V., Bhattacharya, P., and Rakshit, S. (2001). Information theoretic feature crediting in multiclass support vector machines. In *SDM*, pages 1–18. SIAM.

Smith, K., Landes, P.-E., Thollot, J., and Myszkowski, K. (2008). Apparent greyscale: A simple and fast conversion to perceptually accurate images and video. In *Computer Graphics Forum*, volume 27, pages 193–200. Wiley Online Library.

Sonka, M., Hlavac, V., and Boyle, R. (2014). *Image processing, analysis, and machine vision*. Cengage Learning.

Strauss, R. E. and Bookstein, F. L. (1982). The truss: body form reconstructions in morphometrics. *Systematic Biology*, 31(2):113–135.

Tulyakov, S., Jaeger, S., Govindaraju, V., and Doermann, D. (2008). Review of classifier combination methods. In *Machine Learning in Document Analysis and Recognition*, pages 361–386. Springer.

Van Son, R. (1995). A method to quantify the error distribution in confusion matrices. In *EUROSPEECH*.

Venora, G., Grillo, O., Shahin, M., and Symons, S. (2007). Identification of sicilian landraces and canadian cultivars of lentil using an image analysis system. *Food Research International*, 40(1):161–166.

Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, 57(2):137–154.

Wei, J.-M., Yuan, X.-J., Hu, Q.-H., and Wang, S.-Q. (2010). A novel measure for evaluating classifiers. *Expert Systems with Applications*, 37(5):3799–3809.

Zahn, C. T. and Roskies, R. Z. (1972). Fourier descriptors for plane closed curves. *Computers, IEEE Transactions on*, 100(3):269–281.

Zapotoczny, P., Zielinska, M., and Nita, Z. (2008). Application of image analysis for the varietal classification of barley:: Morphological features. *Journal of Cereal Science*, 48(1):104–110.