POLITECNICO DI TORINO

Master degree in Mechatronic engineering

Final project

Machine learning framework for classification of mild cognitive impairment and high resolution multispectral-multitemporal satellite images

Advisor
Prof. Marcello Chiaberge
Prof. Li Yang
Phd. Aleem Khaliq

Master
Leonardo Peroni

March 2018

Abstract

Machine learning is the sub field of computer science that "gives computers the ability to learn without being explicitly programmed" (Arthur Samuel, 1959). Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data. Such algorithms overcome following strictly static program instructions by making data driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms is unfeasible. Examples of the most important areas covered are: finance, healthcare, telecommunications, computer vision, automotive... This thesis is divided in three parts.

The first part is a general overview of machine learning environment where is explained its growing popularity in very different fields of knowledge and the main steps to follow in order to accomplish good results and extract important information. The general scheme of machine learning is proposed and exploited and some concepts about technique and methodology used are explained (pre-processing, feature extraction, feature selection and building of a learning algorithm). The other parts are two different case study, in both of them a different machine learning framework for classification are shown.

In case study 1 are implemented the sub-methods of the general framework used in the published paper "Topological Graph Kernel on Multiple Thresholded Functional Connectivity Networks for Mild Cognitive Impairment Classification" [20] written by Biao Jie et al. in 2013 and corrected in March 2014. The realization is made by using the MATLAB environment and some functions from the LIBSVM and STATISTICAL toolboxes. The topic of the paper is explained more in detail and some recent technique used in the field of supervised learning for brain model are discussed (Cluster Coefficient, ttest for feature extraction, RFE-GK, SVM, kernels, multikernels). In the paper these techniques are used for detecting the "Mild Cognitive Impairment", a condition that is frequently observed before the Alzheimer disease, so a little introduction of this disease is presented. Without loss of generality we can think that the methodologies presented can be applied at each kind of brain illness that involves some anomalies in the functional connection between ROIs (functional part of the brain); so a brief explanation on the modern ways of modeling of the human brain and its connection is given. The implemented code is the practical realization of the theoretical topics treated in this chapter and can be seen as a little library of Matlab useful for further works that involves the same techniques. The research was carried out in School of Automation Science and Electrical Engineering department at the BUAA University, in Beijing, china.

In case study 2 a general framework for crop classification of high resolution multispectral-multiteporal satellite images is implemented.The images were taken by satellite Sentinel-2 lunched in 2015 by European Space Agency (ESA) and were downloaded from Copernicus Open Access Hub website. Since two different data set are considered, two ways of ground truth data acquisition were done. In the first data set the ground truth was obtained by physically going in the fields and taking the GPS coordinates by the help of a mobile-GPS device. In the second dataset the ground truth was obtained from Land Use And Land Cover Survey (LUCAS) archive

offered by ESA and visually analyzed with Quantum Geographic information system (QGIS) application; then for both datasets the features involved are extracted by using Sentinel Application Platform (SNAP) tool from European Space Agency (ESA). The implementation is realized using Python with tensorflow library for deep learning. More in detail are explained the problems concerning pixel-oriented multiclass classification, the extraction and processing of features from satellite image, their selection and the building, training and test phase of Multilayer Perceptrons. For each of the datasets a different framework with a different Multilayer Perceptron architecture is used. The two data set differs significantly in dimension, composition and life-cycle of crops involved, this forced to pursuit a multitemporal time wrapping approach. The results obtained will be showed and commented. The research was carried out in Interdepartmental Laboratory of Mechatronics (LIM) and in Polito Interdepartmental Centre For Service Robotics (PIC4SeR) at Politecnico di Torino.

Acknowledgements

One part of the thesis was carried out in School of Automation Science and Electrical Engineering department at the BUAA University, in Beijing. The other part was carried out in Interdepartmental Mechatronics Laboratory (LIM) of Polito, in Turin. I want to thank the two professors Marcello Chiaberge and Li Yang that gave me the opportunity to work on this very interesting topic; and my instructor Yang Hao that helped me and was always available for every doubts. Special thanks go to Aleem Khaliq that was not only my supervisor, that always helped me whenever I got stuck with the work, but also a friend. Finally I have to thank also my parents for all the support that gave me.

Contents

List of Figures

Chapter 1

The general machine learning approach

1.1 Machine learning

Machine learning is the subfield of computer science that "gives computers the ability to learn without being explicitly programmed" (Arthur Samuel, 1959). Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data, such algorithms overcome following strictly static program instructions by making data driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms is unfeasible; example applications include: spam filtering, detection of network intruders or malicious insiders working towards a data breach, optical character recognition (OCR), search engines and computer vision. The process of machine learning is similar to that of data mining. Both systems search through data to look for patterns. However, instead of extracting data for human comprehension, as is the case in data mining applications, machine learning uses that data to detect patterns in data and adjust program actions accordingly. Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning "signal" or "feedback" available to a learning system. These are:

- **Supervised learning:** The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

- **Unsupervised learning:** No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

- **Reinforcement learning:** A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle), without a teacher explicitly telling it whether it has come close to its goal. Another example is learning to play a game by playing against an opponent.

Between supervised and unsupervised learning is semi-supervised learning, where the teacher gives an incomplete training signal: a training set with some (often many) of the target outputs missing. Transduction is a special case of this principle where the entire set of problem instances is known at learning time, except that part of the targets are missing. There are also other different categories that have a minority role compared to others. For example, Developmental learning, elaborated for robot learning, generates its own sequences (also called curriculum) of learning situations to cumulatively acquire repertoires of novel skills through autonomous self-exploration and social interaction with human teachers and using guidance mechanisms such as active learning, maturation, motor synergies, and imitation. Another categorization of machine learning tasks arises when one considers the desired output of a machine-learned system:

- **In classification**, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".

- **In regression**, also a supervised problem, the outputs are continuous rather than discrete.

- **In clustering**, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

- **Density estimation** finds the distribution of inputs in some space.

- **Dimensionality reduction** simplifies inputs by mapping them into a lower-dimensional space. Topic modeling is a related problem, where a program is given a list of human language documents and is tasked to find out which documents cover similar topics.

1.1.1 General framework

In the basic statistical learning setting, the learner has access to the following:

- **Domain set:** an arbitrary set, X. This is the set of objects that we may wish to label. For example, in the illness classification problem, the domain set will be the set of all people, for crop classification will be the set of all pixels. Usually, these domain points will be represented by a vector of features. The domain points are referred as instances and X as instance space.

- **Label set:** the label set con be a two element, usually {0, 1} or {-1, +1} if refer to binary classification. If it refer to multiclass it can be a two,three or more element. Let Y denote the set of possible labels. For illness classification, let Y be {0, 1}, where 1 represents being sick and 0 stands for being healthy, for satellite image let Y be {0,1,2,3,4}, where 0 is city, 1 maze, 2 cabbage, 3 vineyards, 4 water.

- **Training data:** $S = ((x_1, y_1)...(x_m, y_m))$ is a finite sequence of pairs in $X \times Y$: that is, a sequence of labeled domain points. This is the input that the learner has access to. Such labeled examples are often called training examples. Sometimes S is also referred as training set.

- **The learner's output:** The learner is requested to output a prediction rule, $h : X \to Y$. This function is also called a predictor, a hypothesis, or a classifier. The predictor can be used to predict the label of new domain points. In the preceding examples, it is a rule that our learner will employ to predict whether future people he examines are going to be sick or not or a new pixel is classify as city,maze,cabbage,vineyards or water.

- **The generation of data for the model:** there are different ways to generate data for the model, but for the aim of this thesis the data was taken from samples of a population of people and samples of pixel of a big image. So different inferential statistics techniques could be adopted. (i.e. random sampling, stratified random sampling...) Of course the instances X obtained are distributed with a distribution D. the learner knows nothing about this distribution.

- **Measures of success:** We define the error of a classifier to be the probability that it does not predict the correct label on a random data point generated by the aforementioned underlying distribution. That is, the error of h is the probability to draw a random instance x, according to the distribution D, such that $h(x) \neq f(x)$. Formally, given a domain subset, $A \subset X$, the probability distribution, D, assigns a number, D(A), which determines how likely it is to observe a point $x \in A$. In many cases, we refer to A as an event and express it using a function $\pi : X \to \{0, 1\}$, namely, $A = \{x \in X : \pi(x) = 1\}$. In that case, we also use the notation $P_{x\sim D}[\pi(x)]$ to express D(A). We define the error of a prediction rule, $h : X \to Y$, to be $L_{d,f}(h) \stackrel{def}{=} P_{x\sim D}[h(x) \neq f(x)] \stackrel{def}{=} D(\{x : h(x) \neq f(x)\})$. That is, the error of such h is the probability of randomly choosing an example x for which $h(x) \neq f(x)$. The subscript (D, f) indicates that the error is measured with respect to the probability distribution D and the correct labeling function f. $L_{(d,f)}(h)$ has several synonymous names such as the generalization error, the risk, or the true error of h.

A note about the information available to the learner, the learner is blind to the underlying distribution D over the world and to the labeling function f. The only way the learner can interact with the environment is through observing the training set.

1.2 Data acquisition

Every machine learning algorithm, in order to be trained or to extract useful information, need data. The collection of an initial dataset is the basis for all the steps that will follow. Of course the methods used to acquire data and some refinement of them, strongly depends on the field in which the framework is applied. For example in bio medical environment images are taken from a fRMI (functional magnetic resonance) and than processed as signals in order to be compatible as input for the

learning algorithm. In satellite classification, images of an area are captured by the satellite and downloaded and then resampled and subsetted. For speech recognition continuous signal are captured from a microphone and so on. So in order to have a good framework is important to have a good quality of data, that is why the acquisition step should be made by someone that:

- have a good knowledge of the field of application of ML procedures
- have a good knowledge of the ML algorithm used
- already have in mind, at least as an idea, the total structure of the framework that will be adopted

1.3 Feature extraction

Feature extraction is that activity that starts from an initial set of measured data (such as connectivity matrix and bands of satellite image) and builds derived values in form of vector of features (that represent the characteristics of an observation) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature Extraction methods are transformative in the sense that it is applied a transformation to data to project it into a new feature space. There are at least three reasons why feature extraction is an important problem in predictive modeling and modern data analysis:

- **Dimension Reduction:** in problems with a large number of variables, almost all prediction models suffer from the curse of dimensionality, some more severely than others. Feature extraction can act as a powerful dimension reduction agent. We can understand the curse of dimensionality in very intuitive terms: when a person (or, analogously, a machine learning program) is given too many variables to consider, most of which are irrelevant or simply non-informative, it is naturally much harder to make a good decision. It is therefore desirable to select a much smaller number of relevant and important features. High dimensional problems also pose problems for computation. Sometimes two variables might be equally informative, but are highly correlated with each other; this often causes ill behavior in numerical computation. Feature extraction is, therefore, also an important computational technique.

- **Automatic Exploratory Data Analysis:** In many classical applications, informative features are often selected a priori by field experts, i.e., investigators pick out what they believe are the important variables to build a model. More and more often in modern data-mining applications, however, there is a growing demand for fully automated "black-box" type of prediction models that are capable of identifying the important features on their own. The need for such automated systems arises for two reasons. On the one hand, there are the economic needs to process large amounts of data in a short amount of time with little manual supervision. On the other hand, it is often the case that the problem and the data are so novel that there are simply no field experts who understand the data well enough to be able to pick out the important variables

prior to the analysis. Under such circumstances, automatic exploratory data analysis becomes the key. Instead of relying on pre-conceived ideas, there is a need (as well as interest) to let the data speak for itself.

- **Data Visualization:** Another application of feature extraction that shares the flavor of exploratory data analysis is data visualization. The human eye has an amazing ability in recognizing systematic patterns in the data. At the same time, we are usually unable to make good sense of data if it is more than three dimensional. To maximize the use of the highly developed human faculty in visual identification, we often wish to identify two or three of the most informative features in the data so that we can plot the data in a reduced space. To produce such plots, feature extraction is the crucial analytic step.[37]

1.4 Processing raw features

The two main goal of feature extraction are: dimensionality reduction and improving prediction accuracy. When the focus is on improving prediction accuracy a common step is applied: the processing of features. To describe the processing step, some notations are introduced. Let x be a pattern vector of dimension n, $x = [x_1, x_2, ..., x_n]$. The components x_i of this vector are the original features. We call x' a vector of transformed features of dimension n'. The processing transformations may include:

- **Rescale data:** The features may contain attributes with a mixtures of scales. Many machine learning methods like data attributes to have the same scale.
  - Standardization: features can have different scales although they refer to comparable objects. Consider for instance, a pattern $x = [x_1, x_2]$ where x_1 is a width measured in meters and x_2 is a height measured in centimeters. Both can be compared, added or subtracted but it would be unreasonable to do it before appropriate standardization. The following classical centering and scaling of the data is often used: $x' = (x_i - \mu_i)/\sigma_i$, where μ_i and σ_i are the mean and the standard deviation of feature x_i over training examples.
  - Normalization: consider for example the case where x is an image and the x_i's are the number of pixels with color i, it makes sense to normalize x by dividing it by the total number of counts in order to encode the distribution and remove the dependence on the size of the image. This translates into the formula: $x' = x/|x|$.
  - **Scale to 0-1:**it's just an other kind of standardization but the formula used is $x' = \frac{x_i - min(x)}{max(x) - min(x)}$ and it scale all the feature values from 0 to 1.
- **Signal enhancement:** the signal-to-noise ratio may be improved by applying signal or image-processing filters. These operations include baseline or background removal, de-noising, smoothing, or sharpening. The Fourier transform and wavelet transforms are popular methods.
- **Extraction of local features:** for sequential, spatial or other structured data, specific techniques like convolutional methods using hand-crafted kernels or syntactic and structural methods are used. These techniques encode problem specific knowledge into the features.

- **Linear and non-linear space embedding methods:** when the dimensionality of the data is very high, some techniques might be used to project or embed the data into a lower dimensional space while retaining as much information as possible.

- **Non-linear expansions:** although dimensionality reduction is often summoned when speaking about complex data, it is sometimes better to increase the dimensionality. This happens when the problem is very complex and first order interactions are not enough to derive good results. This consists for instance in computing products of the original features x_i to create monomials $x_{k1}, x_{k2}, ..., x_{kp}$.

- **Feature discretization:** some algorithms do no handle well continuous data. It makes sense then to discretize continuous values into a finite discrete set.

This step not only facilitates the use of certain algorithms, it may simplify the data description and improve data understanding. Some methods do not alter the space dimensionality (e.g. signal enhancement, normalization, standardization), while others enlarge it (non-linear expansions, feature discretization), reduce it (space embedding methods) or can act in either direction (extraction of local features).[13]

1.5 Feature selection

Feature selection is a critical step in the feature construction process. Since the transformed feature space FN (obtained from the feature extraction step) is large, we need another step to select a subset FT of FN. The problem of selecting the optimal subset is NP-hard, and the methods usually perform some sort of sub-optimal greedy search. Frequently used criteria for measuring the utility of a feature space F_i include information gain, correlation coefficient, prediction accuracy on some validation set etc. A plethora of different selection methods have been presented in the literature. selection techniques are used for three reasons:

- simplification of models to make them easier to interpret by researchers/users

- shorter training times

- enhanced generalization by reducing overfitting

The central premise when using a feature selection technique is that the data contains many features that are either redundant or irrelevant, and can thus be removed without incurring much loss of information. Redundant or irrelevant features are two distinct notions, since one relevant feature may be redundant in the presence of another relevant feature with which it is strongly correlated. A feature is useful if it is correlated with or predictive of the class; otherwise it is irrelevant.[16] Empirical evidence from the feature selection literature shows that, along with irrelevant features, redundant information should be eliminated as well. A feature is said to be redundant if one or more of the other features are highly correlated with it. The above definitions for relevance and redundancy lead to the idea that best features for a given classification are those that are highly correlated with one of the classes and have an insignificant correlation with the rest of the features in the set. Feature

selection techniques should be distinguished from feature extraction. Feature extraction creates new features from functions of the original features, whereas feature selection returns a subset of the features. Feature selection techniques are often used in domains where there are many features and comparatively few samples (or data points)

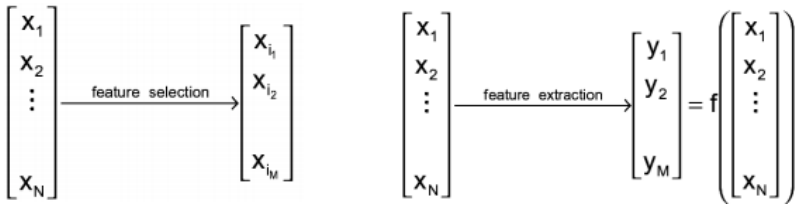


Figure 1.1: comparison feature extraction and feature selection

1.5.1 Feature selection methodologies

The feature selection strategy can be summarized in three aspects:

- feature subset generation (or search strategy);
- evaluation criterion definition (e.g. relevance index or predictive power);
- evaluation criterion estimation (or assessment method).

By the different way of using the aspects above we can subdivide the feature selection methodologies in three categories:

- **Filters:** Filters select the feature subsets independent of the predictor. They essentially operate as a data preprocessing step before a predictor is trained. Variable ranking approaches, which involve ranking individual features using information theoretic or correlation criteria, and then constructing a subset of high scoring features, belong in this category. Filters have an advantage in that they are faster than wrappers. Moreover, they tend to provide a generic (and hence insightful) ordering of features not tuned for a specific learning method. A disadvantage however is that the chosen subset may not be the best suited for the predictor to be used in the next step.

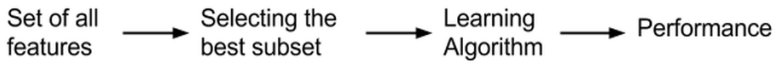


Figure 1.2: work-flow of filter selection approach

- **Wrappers:** Wrappers are feature selection methods that use the learning method to be used for prediction as a black box to select feature subsets. These methods typically divide the training set into a train and validation set (the test set is separate). For any given feature subset, the predictor is trained on the train set and tested on the validation set. The prediction accuracy on the validation set is considered as the score of the feature subset. Thus we

would ultimately want to choose the highest scoring feature subset. Due to repeated train and test cycles for every feature subset, wrappers tend to be much more computationally intensive compared to filters. The goal usually is to traverse the feature space such that the number of subsets to be tested is minimized. An obvious advantage however is that the chosen subset is tuned to the predictor.

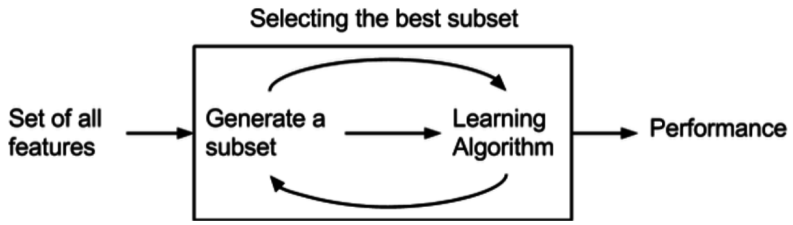


Figure 1.3: work-flow of wrapper selection approach

- **Embedded:** Embedded methods combine the process of feature selection and model learning. These methods are highly specific to the learning machine. For example, we could modify the objective function of an SVM to also minimize the number of features along with the error. Such methods are often fast and lead to accurate predictors. They are however not directly generalizable to any predictor.

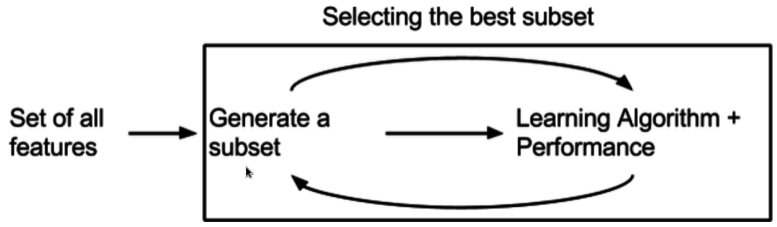


Figure 1.4: work-flow of embedded selection approach

Synthesizing: filters and wrappers differ mostly by the evaluation criterion. It is usually understood that filters use criteria not involving any learning machine, e.g. a relevance index based on correlation coefficients or test statistics, whereas wrappers use the performance of a learning machine trained using a given feature subset. Among wrapper and embedded methods, greedy methods (forward selection or backward elimination) are the most popular. In a forward selection method one starts with an empty set and progressively add features yielding to the improvement of a performance index. In a backward elimination procedure one starts with all the features and progressively eliminate the least useful ones. Both procedures are robust against overfitting.
The main characteristic of forward selection are:

- faster in early steps because fewer features to test
- fast for choosing a small subset of the features

- misses features whose usefulness requires other features (feature synergy)

The main characteristic of backward selection are:

- fast for choosing all but a small subset of the features
- preserves features whose usefulness requires other features

Both procedures provide nested feature subsets. However they may lead to different subsets and, depending on the application and the objectives, one approach may be preferred over the other one.

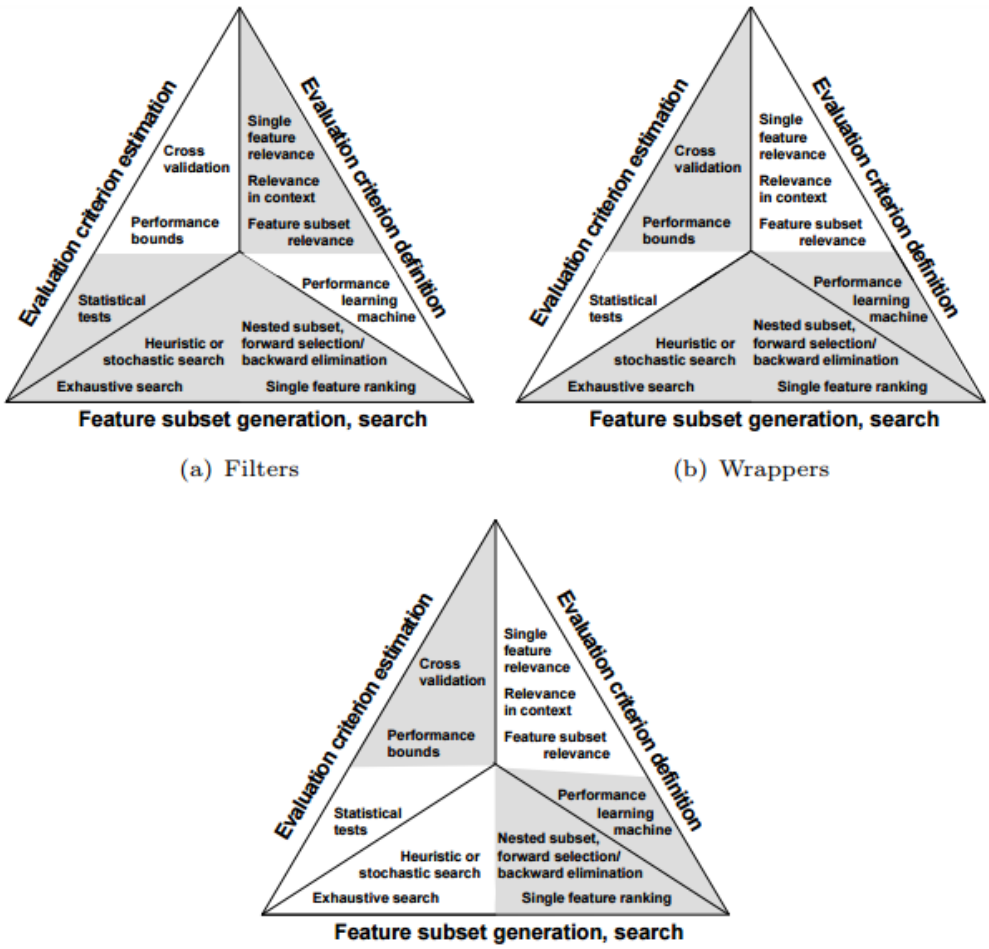


Figure 1.5: description map of the tree feature selection methods. The groups are colored in gray.

1.5.1.1 Assessment method

The difficulty to overcome is that a defined criterion (a relevance index or the performance of a learning machine) must be estimated from a limited amount of training data, so is necessary to use a strategy that permit to give to the chosen criterion the most general validity as possible. In other words: follow a procedure that allow the extension of results obtained from samples to the population. Two strategies are possible: "in-sample" or "out-of-sample".

- The first one (in-sample) is the "classical statistics" approach. It refers to using all the training data to compute an empirical estimate. That estimate is then tested with a statistical test to assess its significance, or a performance bound is used to give a guaranteed estimate.
- The second one (out-of-sample) is the "machine learning" approach. It refers to splitting the training data into a training set used to estimate the parameters of a predictive model (learning machine) and a validation set used to estimate the learning machine predictive performance. Averaging the results of multiple splitting (or "cross-validation") is commonly used to decrease the variance of the estimator.[13]

The focus is on the second strategy and it is also used to evaluate the generality of classifier, so it has been exploited in the next section.

1.5.2 Problems not solved

Although the use of feature selection to improve the predictability of our classifier some problems still remain:

- **Overfitting:** If we consider the task of feature construction as selecting an optimal subset of features in a potentially infinite feature space, the possibility of overfitting (in next section the concept is more investigated) becomes clearly apparent. In a highly expressive feature space with a comparatively small number of examples, there may be many hypotheses that are consistent with the data. Choosing the right one becomes a problem. Most systems use some sort of a heuristic search methodology and tend to prefer a sub-optimal subset as it is less likely to overfit. To the best of knowledge, there is currently no literature that specifically explores this problem to identify the best search strategies.
- **Difficulty in Comparison:** While a large number of feature construction methods have been proposed, there has been little or no comparison between them. Most papers present methods that are either meant for specific problems or are evaluated only in certain settings using specific predictors. There has rarely been any evaluation across classifiers.
- **Incorporating domain knowledge:** Incorporating domain knowledge into the feature construction has been and still remains a major challenge. Most methods do this by choosing a set of operators and putting constraints on the feature generation process. This has two major drawbacks. First it's not always easy to encode domain knowledge in terms of operators. Second, the use of operators frequently results in a huge feature space which then needs to be searched. Recent approaches deal with this problem by allowing users to supply domain knowledge in the form of annotations which are then used for constructing the feature space. More such methods need to be explored.[34]

1.6 Classifier

"A computer program is said to learn from experience E with respect to some class of tasks T and a performance measure P if it improves performance on T (according

to P) with more E." (Tom Mitchell)[26].

This is the core of the machine learning subject: build and algorithm able to predict the output of some input. It will be able to do that because we already have trained it. The training procedure is operatively exploited by using the feature vectors obtained from the previous step as input vectors (the experience E) for the algorithm. Close to the concept of training there is also the concept of evaluating. Evaluating an algorithm means measure how much is able to predict values. From this evaluation step is possible to decide if we need to train more our algorithm or if it is not suitable for our aim.
Every algorithm has three components:

- **Hypothesis space-possible outputs** (decision trees, instances, neural networks, SVM...)
- **Search strategy-strategy for exploring space** (greedy, exhaustive, optimize objective function, branch and bound...)
- **Evaluation** (accuracy,precision,recall,likelihood...)

A Classifier is a particular example of learning algorithm which goal is identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.
In the terminology of machine learning, classification is considered an instance of supervised learning. The corresponding unsupervised procedure is known as clustering, and involves grouping data into categories based on some measure of inherent similarity or distance.

1.6.1 Train

A classifier is essentially a mathematical structure able to associate a new observation to a class. Being a mathematical structure it is composed by some parameters, for example the orientation W and the displacement b in svm, or the biases and connection weights in neural networks. Train a classifier means teach to it which values of the parameters are the best in order to minimize a defined cost function, the minimization of the cost function lead to a better ability to classify the new samples.
Practically in supervised learning the training procedure is done by giving to the learning algorithm a set of observation of which the classes are known, so the algorithm can learn (with different approach, i.e back-propagation for Neural Network or support vectors for svm) the parameters.
From now on it is assumed that the dataset is splitted in two part: the train set and test set, the former used to train the classifier and the latter to evaluate its quality. In section "Assessment method" the details of the splitting of dataset are exploited.

Figure 1.6: work-flow of training phase

1.6.2 Evaluation

Evaluate a classifier essentially means give a measure of quality to it.
The good quality con be evaluated by accounting two different aspects:

- Actual evaluation: use a metrics able to encapsulate how much the trained classifier is good by using as input the test set
- Assessment of classifier: measure how much the evaluation metrics are general or in other words how much the evaluation metrics of classifier is dependent from the dataset used.

Both aspects are strictly correlated; in this section the former aspect is exploited by analyzing the most used metrics in fields of binary and multiclass classification, the latter is exploited in the next section "Assessment method".

1.6.2.1 Confusion matrix

The confusion matrix as suggested by the name is a table able to give information about the quality of a classifier.
Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa).The name stems from the fact that it makes it easy to see if the system is confusing two or more classes. The number of rows and column are the same and all correct predictions are located in the diagonal of the table, so it is easy to visually inspect the table for prediction errors, as they will be represented by values outside the diagonal. It is used both in binary and multiclass classification, but in binary ones the rows and columns are 2 and the instances in the table are called: "true positive","false positive","true negative","false negative".

		Reference Data			
		Water	Forest	Urban	Total
Classified Data	Water	21	6	0	**27**
	Forest	5	31	1	**37**
	Urban	7	2	22	**31**
	Total	**33**	**39**	**23**	***95***

(a) multiclass confusion matrix

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

(b) binary confusion matrix

Figure 1.7: Example of two confusion matrix, a) for multiclass classification and b) for binary classification

From both confusion matrices different metrics can be extracted and in general they are grouped in function of the kind of classification.

Binary classification

For defining the next metrics, especially for bio medical field, is necessary to define:

- True positive (TP): Sick people correctly identified as sick
- False positive (FP): Healthy people incorrectly identified as sick
- True negative (TN): Healthy people correctly identified as healthy
- False negative (FN): Sick people incorrectly identified as healthy

The most used parameters for evaluating the performance of a classifier are:

- $ACC = \frac{TP+TN}{TN+FN+TN+FP}$
- $BAC = \frac{1}{2} \times (\frac{TP}{TP+FN} + \frac{TN}{TN+FP})$

Where the accuracy (ACC) represents the ability of correctly identify a test subject. The balanced accuracy has the same meaning but take care also for the imbalance problem: a two-class data set is said to be imbalanced when one of the classes (the minority one) is heavily under-represented as regards the other class (the majority one). In this case the only accuracy is not significant for evaluating the performance of the classifier.

Sensitivity and specificity

Sensitivity refers to the test's ability to correctly detect patients who do have the condition. In the example of a medical test used to identify a disease, the sensitivity of the test is the proportion of people who test positive for the disease among those who have the disease. Mathematically, this can be expressed as:

$$\text{Sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives + number of false negative}} = \frac{\text{nr TP}}{\text{nr sick people}}$$

$$= \text{probability of a positive test given that the patient has the disease}$$

A negative result in a test with high sensitivity is useful for ruling out disease. A high sensitivity test is reliable when its result is negative, since it rarely misdiagnoses those who have the disease. A test with 100% sensitivity will recognize all patients with the disease by testing positive. A negative test result would definitively rule out presence of the disease in a patient. A positive result in a test with high sensitivity is not useful for ruling in disease. Suppose a 'bogus' test kit is designed to show only one reading, positive. When used on diseased patients, all patients test positive, giving the test 100% sensitivity. However, sensitivity by definition does not take into account false positives. The bogus test also returns positive on all healthy

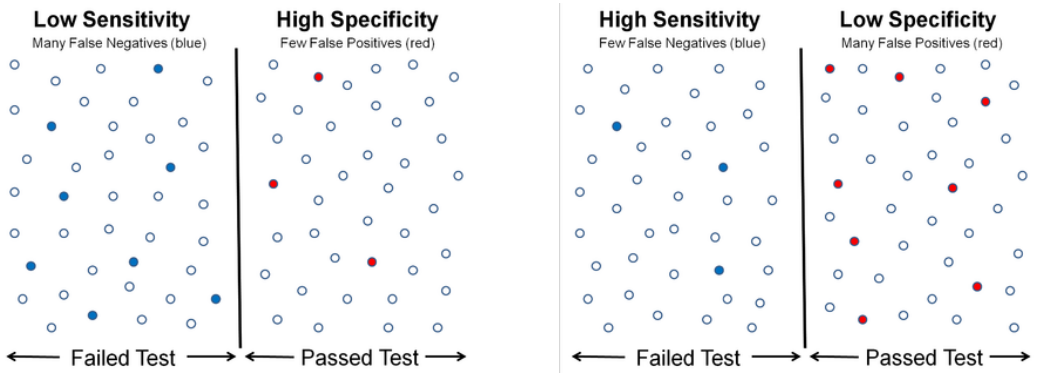
patients, giving it a false positive rate of 100%, rendering it useless for detecting or "ruling in" the disease.

Specificity relates to the test's ability to correctly detect patients without a condition. Consider the example of a medical test for diagnosing a disease. Specificity of a test is the proportion of healthy patients known not to have the disease, who will test negative for it. Mathematically, this can also be written as:

$$\text{Specificity}=\frac{\text{number of true negatives}}{\text{number of true negative + number of false positive}}=\frac{\text{nr TN}}{\text{nr healthy people}}$$

$$=\text{probability of a negative test given that the patient is healthy}$$

A positive result in a test with high specificity is useful for ruling in disease. The test rarely gives positive results in healthy patients. A test with 100% specificity will read negative, and accurately exclude disease from all healthy patients. A positive result signifies a high probability of the presence of disease. A negative result in a test with high specificity is not useful for ruling out disease. Assume a 'bogus' test is designed to read only negative. This is administered to healthy patients, and reads negative on all of them. This will give the test a specificity of 100%. Specificity by definition does not take into account false negatives. The same test will also read negative on diseased patients, therefore it has a false negative rate of 100%, and will be useless for ruling out disease.



(a) low sensitivity, high specificity

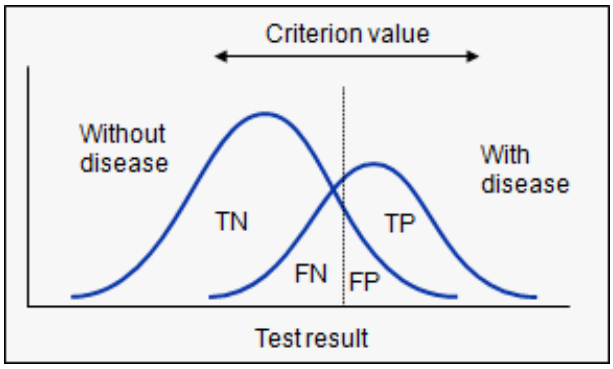
(b) high sensitivity, low specificity

Figure 1.8: representation of two opposite case: a)low sensitivity and high specificity b)high sensitivity and low specificity

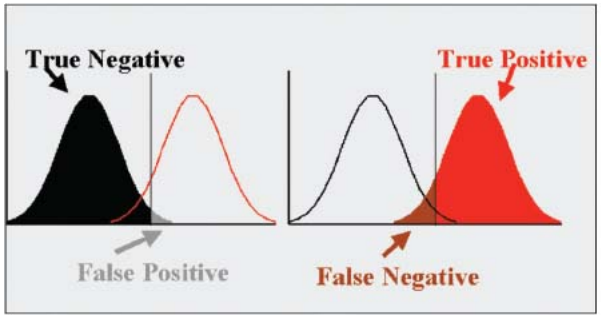
trade off sensitivity-specificity

When the results of a particular test in two populations, one population with a disease, the other population without the disease, are considered; it's rarely observed a perfect separation between the two groups. Indeed, the distribution of the test results will overlap, as shown in figure 1.9.

For every possible cut-off point or criterion value you select to discriminate between the two populations (figure 1.10), there will be some cases with the disease



(a)



(b)

Figure 1.9: plots of the overlapping between true positive and true negative samples

correctly classified as positive (TP = True Positive fraction), but some cases with the disease will be classified negative (FN = False Negative fraction). On the other hand, some cases without the disease will be correctly classified as negative (TN = True Negative fraction), but some cases without the disease will be classified as positive (FP = False Positive fraction). Percentage of sensitivity is the area under the true positive curve and cut-off (red in figure 1.9 (b)). Percentage of specificity is the area under the true negative curve and cut-off (black in figure 1.9 (b)). So by moving the cut-off we are deciding if increasing the sensibility and decreasing the specificity or the opposite. Move the cut-off to have very high value of specificity or sensibility will give result that are worthless: imagine to have sensitivity of 10% and specificity of 90%. Let's say you have a positive result; you might think that the disease is more likely, since the specificity is 90%. In fact, the specificity tells you that 10% of healthy people will have a positive result - but by looking at the sensitivity, we see that only 10% of diseased patients will have a positive result. In other words, the rate of positive results is the same between diseased and healthy people, which means that this positive result means you are no more or less likely to be diseased, i.e. that the test is worthless.

ROC

To draw a ROC curve, only the true positive rate (TPR) and false positive rate (FPR) are needed (as functions of some classifier parameter). The TPR defines how many correct positive results occur among all positive samples available during the test. FPR, on the other hand, defines how many incorrect positive results occur among all negative samples available during the test. A ROC space is defined

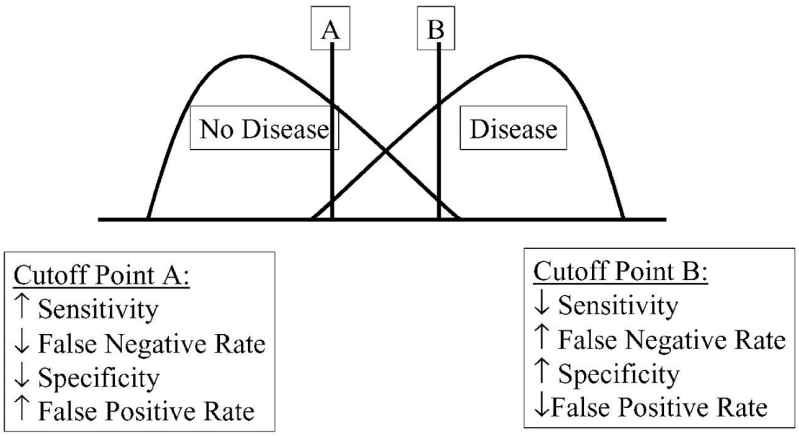


Figure 1.10: comparison between two different cut off thresholds

by FPR and TPR as x and y axes respectively, which depicts relative trade-offs between true positive (benefits) and false positive (costs). Since TPR is equivalent to sensitivity and FPR is equal to 1 - specificity, the ROC graph is sometimes called the sensitivity vs (1 - specificity) plot. Each prediction result or instance of a confusion matrix represents one point in the ROC space. The best possible prediction method would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). The (0,1) point is also called a perfect classification. A completely random guess would give a point along a diagonal line (the so-called line of no-discrimination) from the left bottom to the top right corners (regardless of the positive and negative base rates). In rough words ROC curves represent the relation between sensitivity and (1-specificity) while varying the threshold line. From the

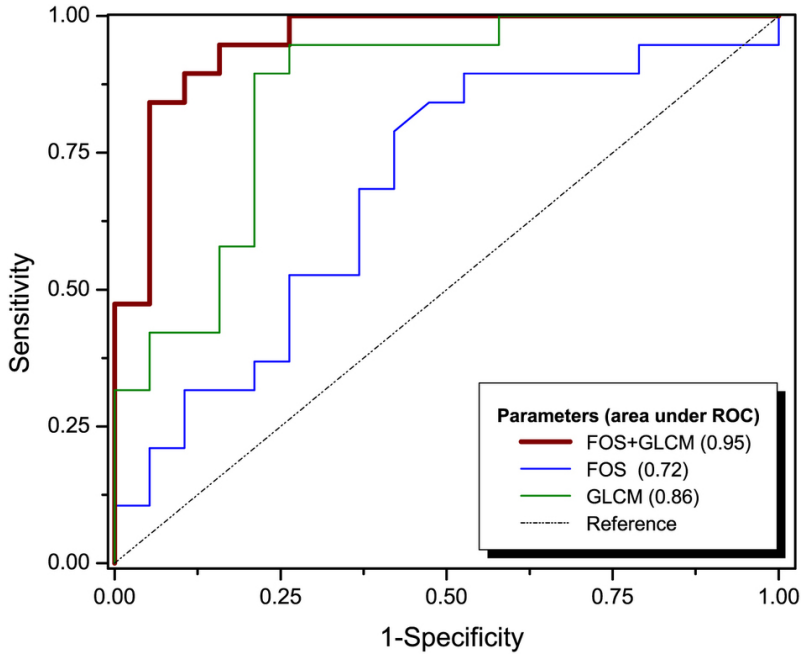


Figure 1.11: example of ROC curve

analysis of the Area Under the Curve (AUC) we are able to detect if a classifier is good or not because the AUC is equal to the probability that a classifier will rank

a randomly chosen positive instance higher than a randomly chosen negative one (assuming 'positive' ranks higher than 'negative'). More the curve look like a line worst will be the classification capabilities of the classifier. In detail the values for quantify the quality of a test:

- AUC 0.9-1: Excellent
- AUC 0.8-0.9: Good
- AUC 0.7-0.8: Fair
- AUC 0.6-0.7: Poor
- AUC 0.5-0.6: Fail

ROC analysis is part of a field called "Signal Detection Theory" developed during World War II for the analysis of radar images. Radar operators had to decide whether a blip on the screen represented an enemy target, a friendly ship, or just noise. Signal detection theory measures the ability of radar receiver operators to make these important distinctions. Their ability to do so was called the Receiver Operating Characteristics. It was not until the 1970's that signal detection theory was recognized as useful for interpreting medical test results.

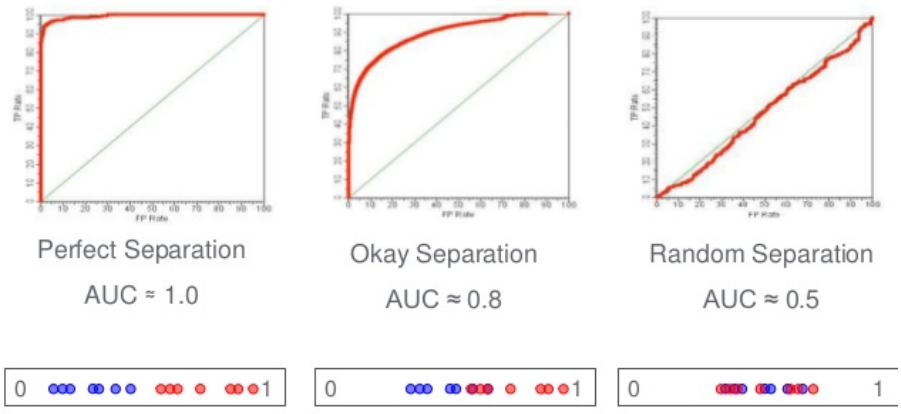


Figure 1.12: work-flow of training-prediction phase

Multiclass classification

Differently form the binary classification the confusion matrix is composed by more than two rows and columns where on the diagonal there are the right classified observations and the misclassificatons on the other cells. The most used metric is the Overall Accuracy, defined as:

$$\text{overall accuracy} = \frac{\text{total correct classified samples}}{\text{total samples}}$$

it is represented in percentage and express the proportion of correctly classified observations.

kappa score

The Kappa statistic (or value) is a metric that compares an Observed Accuracy with an Expected Accuracy (random chance). Taking into account random chance (agreement with a random classifier), generally means it is less misleading than simply using accuracy as a metric (an Observed Accuracy of 80% is a lot less impressive with an Expected Accuracy of 75% versus an Expected Accuracy of 50%). In few words it compares how much better is the predictive ability of the developed classify instead of a one that predict randomly. The formula is:

$$k = \frac{N \sum_{i=1}^{n} m_{i,i} - \sum_{i=1}^{n} (G_i C_i)}{N^2 - \sum_{i=1}^{n} (G_i C_i)}$$

where:

- i is the class number
- N is the total number of classified values compared to truth values
- mi,i is the number of values belonging to the truth class i that have also been classified as class i (i.e., values found along the diagonal of the confusion matrix)
- C_i is the total number of predicted values belonging to class i
- G_i is the total number of truth values belonging to class i[17]

User accuracy

The User's Accuracy (UA) is the accuracy from the point of view of a map user, not the map maker. the User's accuracy essentially tells how often the class on the map will actually be present on the ground. This is referred to as reliability. The User's Accuracy is complement of the Commission Error, User's Accuracy = 100%-Commission Error. The User's Accuracy is calculating by taking the total number of correct classifications for a particular class and dividing it by the row total.
Errors of commission refer to sites that are classified as reference sites that were left out (or omitted) from the correct class in the classified map. Commission errors are calculated by reviewing the classified sites for incorrect classifications.[36] For example in the multiclass confusion matrix of fig 1.7:

- Water:
  - Correctly classified reference sites = 21
  - Total nr of reference sites = 27
  - Producer's Accuracy = 21/33 = 78%
- Forest:
  - Correctly classified reference sites = 31

  - Total nr of reference sites = 37
  - Producer's Accuracy = 31/39 = 84%

- Urban:

  - Correctly classified reference sites = 22
  - Total nr of reference sites = 31
  - Producer's Accuracy = 22/23 =70%

Producer accuracy

Producer's Accuracy (PA) is the map accuracy from the point of view of the map maker (the producer). This is how often are real features on the ground correctly shown on the classified map or the probability that a certain land cover of an area on the ground is classified as such. The Producer's Accuracy is complement of the Omission Error, Producer's Accuracy = 100%-Omission Error. It is also the number of reference sites classified accurately divided by the total number of reference sites for that class.
Errors of omission refer to reference sites that were left out (or omitted) from the correct class in the classified map. The real land cover type was left out or omitted from the classified map. Error of omission is sometime also referred to as a Type I error. An error of omission in one category will be counted as an error in commission in another category. Omission errors are calculated by reviewing the reference sites for incorrect classifications. [36] For example in the multiclass confusion matrix of fig 1.7:

- Water:

  - Correctly classified reference sites = 21
  - Total nr of reference sites = 33
  - Producer's Accuracy = 21/33 = 64%

- Forest:

  - Correctly classified reference sites = 31
  - Total nr of reference sites = 39
  - Producer's Accuracy = 31/39 = 80%

- Urban:

  - Correctly classified reference sites = 22
  - Total nr of reference sites = 23
  - Producer's Accuracy = 22/23 =96%

1.6.3 Overfitting and underfitting

The main goal of machine learning is to build an algorithm that learned with some data is able to predict the behavior of new data. The two main problems that can be encountered after the training and testing phase are:

- overfitting: the error obtained from the model on the trained data is very small and the error from the new unseen data is very big. This actually means that the model is too complex (has to much parameters) and fit too well the trained data, so instead of learning how to generalize a trend, the model memorize the pattern. The three main cause of overfitting are:
  - learning performed too long (especially for Neural Network)
  - too few training samples
  - use of a model with too much parameters (too complex)
- underfitting: the error obtained from the model on the trained data and on the new unseen data is big. This means that the model is too simple (has not enough parameters) to generalize the trend of the data set (for example when fitting a linear model to non-linear data).

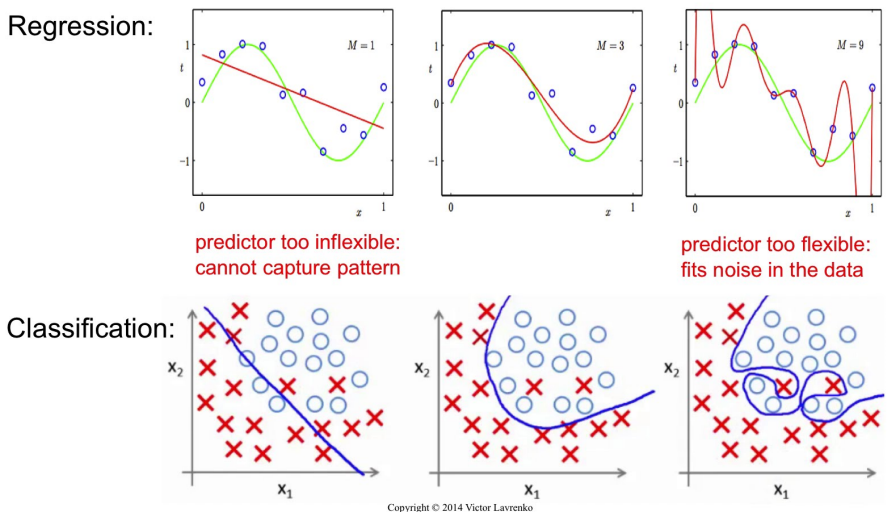


Figure 1.13: Overfitting and underfitting in regression and classification

Strongly related with overfitting and underfitting are the concepts of bias and variance of a learning algorithm. This two concepts are exploited in next section.

1.6.4 Assessment method

Assessing a machine learning result means evaluate how general its behaviour is. For training is taken as data set a sample that came from a population. So the evaluation of a learning algorithm may be sensitive to sampling error. As a result, measurements of prediction error on the current data may not provide much information about predictive ability on new data. As consequence we would like to know how is the algorithm's behaviour for the entire population. In other words the aim of assessing the machine learning framework is to minimize the expected

generalization error (also known as the out of sample error) of the estimator (in this case the classifier function).

Given a training set consisting of a set of points $x_1 ... x_n$ and real values y_i associated with each point x_i. We assume that there is a function with noise $y = f(x) + \epsilon$ where the noise, ϵ has zero mean and variance σ^2, the classifier can be seen as a function $\hat{f}(x)$ that approximates the true function $f(x)$ as well as possible. Defining the expected mean square error between y and $\hat{f}(x)$ as the expected generalization error and by use some maths calculation we can decompose the error into three terms[35]:

$$E[(y - \hat{f}(x))^2] = Bias[\hat{f}(x)]^2 + Var[\hat{f}(x)] + \sigma^2$$

where:

$$Bias[\hat{f}(x)] = E[\hat{f}(x) - f(x)]$$

and

$$Var[\hat{f}(x)] = E[\hat{f}(x)^2] - E[\hat{f}(x)]^2$$

This decomposition show us that the generalization error is composed of three terms:

- the square of bias of model: it is the error due to the incapability of model to understand the pattern to learn, in fewer word: the model is too simple and can't be able to predict well the new input given.
- the variance of model: it's the sensitivity of the model, how much its predictive ability change by changing the data set.
- an irreducible error due to noise.

It's possible to tune just two of the three terms: bias and variance. Among them exists an inversion relation, increase one means decrease the other and vice versa. So the best model is the one that minimize the general expected error that means the best compromise between variance and bias (this problem is called the bias-variance trade off). The relation between bias-variance and overfitted-underfitted

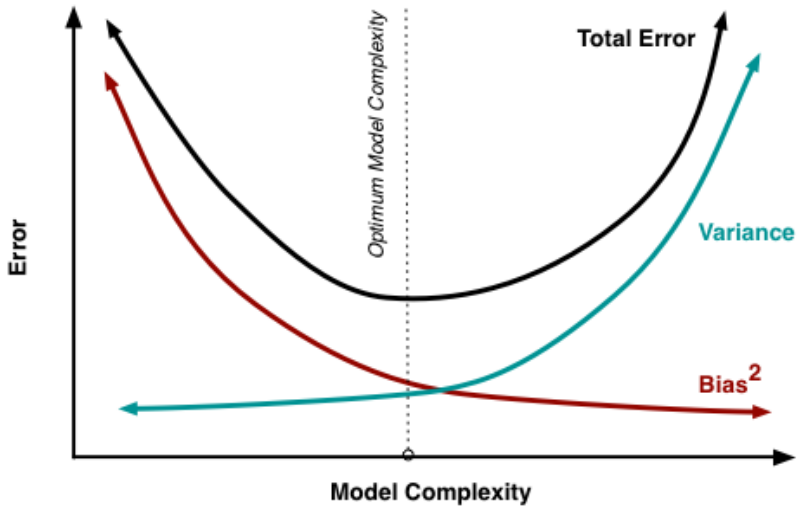


Figure 1.14: Bias-variance trade off

model is strong: a model with low bias and high variance lead to overfitting because

the model learn very well the training pattern but is not predictive for new data; a model with high bias and low variance is underfitted because it has bad predictive ability but almost the same for each dataset. The two most used ways of assess a model in supervised machine learning are: the holdout and the cross validation, both of them are of the class of "out of sample" that means that for evaluate the model an unseen test set different from the training one is used.

1.6.4.1 Holdout

In the old out method the ground truth dataset is divided in three parts:

- **training set:** is a subset of the dataset used to build predictive model.
- **validating set:** is a subset of the dataset used to assess the performance of model built in the training phase. It provides a test platform for fine tuning model's parameters and selecting the best-performing model. Not all modeling algorithms need a validation set.
- **test set:** or unseen examples, is a subset of the dataset to assess the likely future performance of a model. If a model fit to the training set much better than it fits the test set, overfitting is probably the cause.

This method is valid only if there is a large representative sample to train the model and another independent large representative sample to test the model, otherwise if the data set is small or not representative it will lead to a bad generalization result. And it doesn't give a numerical quantification of the variance error. But in return it is fast and doesn't need big computational resources.

1.6.4.2 K-fold cross-validation

In k fold cross validation similarly to holdout method we divide the dataset in train set and test set. But differently form the above method is how we divide. Given the dataset the general approach is:

1. Split the dataset into k sub dataset called folds
2. Build the square matrix where at each round (rows) we use as test set a different fold and as train set all the residual folds (columns)
3. follow the algorithm:

Data: square matrix of folds
Result: mean accuracy estimation,variance
for *each round i* **do**
  validate the model with round i;
  extract the accuracy of round i;
end
extract the mean value of accuracy;
extract the variance;

Algorithm 1: K-fold cross validation

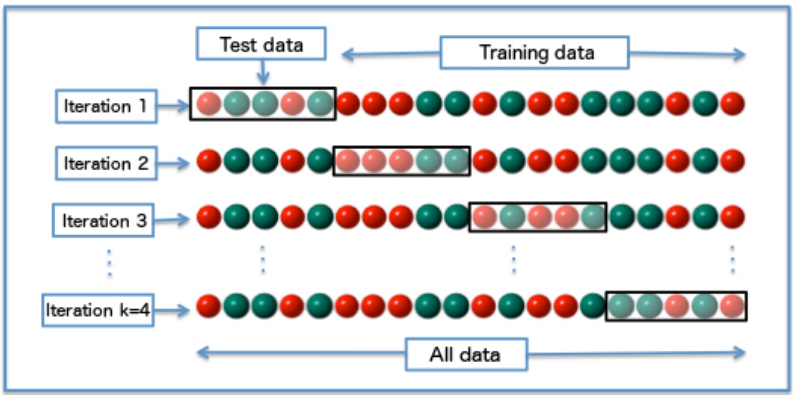


Figure 1.15: example of matrix of 4-fold. Here rounds are called iteration and the chosen test data is shifted at each row

The obtained results with this method are function of k (the number of folds). For large value of k there will be low Bias (The training set is almost the total) but big Variance (highly dependent from the dataset). For low value of k, we will have high Bias and low variance. From experience the recommended values of k are 5 or 10 and there is a big literature that explain how to choose the best k. One extreme case is given by taking k equal to the number of samples. This is used in binary classification with few samples, it is called Leave-one-out cross validation (LOOCV). In general, the use of Cross validation (CV) allows to reduce the risk of incur in overfitting and at the same time is able to give a pretty good evaluation of the performance of a model also with a reduced number of dataset (this happen very frequently in practical problem), and from it, is possible to extract a value for the variance of the model. In the opposite it need a lot of calculus time.
Synthesizing when the computational time for evaluate the framework is not a problem a cross validation method is always preferred. When the dataset is small the cross validation is compulsory. If the dataset is big and representative of all the population and the computational time for CV is unreasonable the hold out method is preferred.

Nested cross-validation

Nested CV estimates the generalization error of the underlying model and its (hyper)parameter search. Choosing the parameters that maximize non-nested CV biases the model to the dataset, yielding an overly-optimistic score. Model selection without nested CV uses the same data to tune model parameters and evaluate model performance. Information may thus "leak" into the model and overfit the data. The magnitude of this effect is primarily dependent on the size of the dataset and the stability of the model[11]. To avoid this problem, nested CV effectively uses a series of train/validation/test set splits: once the dataset are divided in folds (train and test) each training part is divided again in k folds where one fold is for validation and the rest for training. The outer loop is used to evaluate the generalizability of classifier model, and an inner loop is used to choose the best parameter to select the best fitting model. So for each trial of the outer loop a best fitting model is evaluate and at the end of the outer loop an average evaluation of all the best fitting model is presented. And that is the evaluation of the general model. For lead such

evaluation technique a big power of calculus is required.

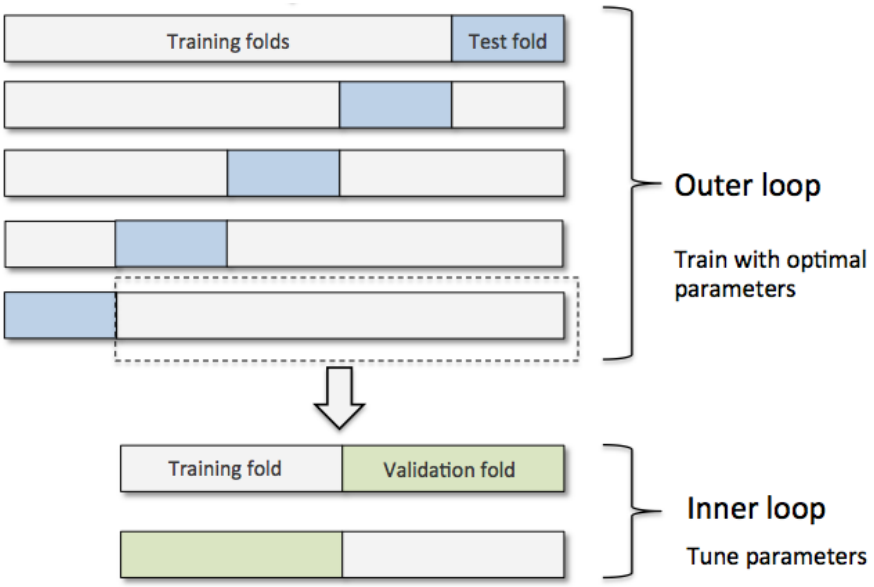


Figure 1.16: Example of nested cross-validation

1.6.5 Prediction

Once the trained and evaluated model is obtained by following all the preceding steps, it is ready to predict the class of total new unlabelled input with the probability, of correctly classify it, given by the accuracy.

Figure 1.17: work-flow of prediction phase

1.7 General structure

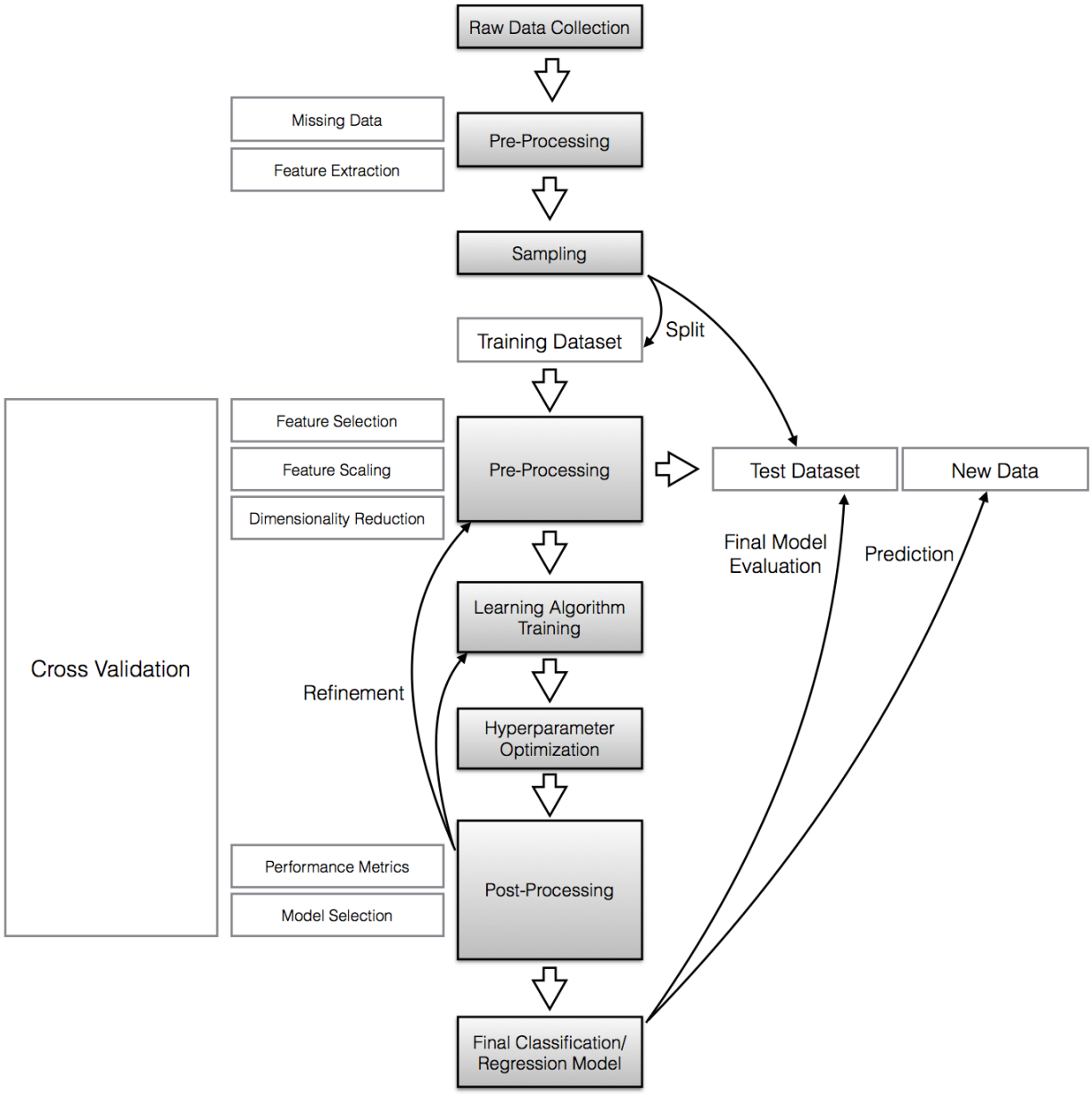


Figure 1.18: General detailed scheme of a supervised machine learning framework

Chapter 2

Case study 1: methodologies involved in the paper "topological graph kernel on multiple thresholded functional connectivity networks for mild cognitive impairment classification"

2.1 Problem definition

Mild cognitive impairment (MCI) is a condition in which someone has minor problems with cognition and their mental abilities such as memory or thinking. In MCI these difficulties are worse than would normally be expected for a healthy person of their age. However, the symptoms are not severe enough to interfere significantly with daily life, and so are not defined as dementia. It is estimated that between 5 and 20 per cent of people aged over 65 have MCI. It is not a type of dementia, but a person with MCI is more likely to go on to develop dementia.

2.1.1 Symptoms

The term MCI describes a set of symptoms, rather than a specific disease. A person with MCI has mild problems with one or more of the following:

- memory - for example, forgetting recent events or repeating the same question
- reasoning, planning or problem-solving - for example, struggling with thinking things through
- attention - for example, being very easily distracted
- language - for example, taking much longer than usual to find the right word for something

- visual depth perception - for example, struggling to interpret an object in three dimensions, judge distances or navigate stairs.

These symptoms will have been noticed by the individual, or by those who know them. For a person with MCI, these changes may cause them to experience minor problems or need a little help with more demanding daily tasks (paying bills, managing medication, driving). If there is a significant impact on everyday activities, this may suggest dementia. Most healthy people experience a gradual decline in mental abilities as part of aging. In someone with MCI, however, the decline in mental abilities is greater than in normal aging.
If the person with MCI has seen a doctor and taken tests of mental abilities, their problems will also be shown by a low test score or by falling test scores over time. This decline in mental abilities is often caused by an underlying illness[1].

2.1.2 MCI Causes and evolution

MCI can have a number of different possible causes. Some of these are treatable and some are not. In some people, MCI is a 'pre-dementia' condition. This means that the brain diseases that cause dementia are already established. These diseases are not generally reversible and so, in time, these people's symptoms will worsen and their condition will progress from MCI to dementia. For example, some people with MCI have mild memory loss that started gradually. These people are likely to develop Alzheimer's disease as their memory worsens. Some people with MCI will turn out to have a different, often treatable, cause following assessment by a doctor. This could include depression, anxiety or stress. The same symptoms could also be caused by a physical illness (constipation, infection), poor eyesight or hearing, vitamin or thyroid deficiencies, or the side effects of medication. Where this is the case, the person will be diagnosed with this condition - a thyroid deficiency or depression, for example - rather than MCI. A doctor will not always be able to say what is causing MCI, even after a thorough assessment. It may be necessary to wait a few months or more, to see how the person's symptoms develop.

2.2 Past related works

So far, many methods have been developed to identify predictive biomarkers of AD (Alzheimer disease) or MCI from different neuroimaging modalities.
In the past decade, modern magnetic resonance imaging (MRI) (e.g., functional MRI (FMRI) and diffusion MRI), and neurophysiological [e.g., electroencephalograph (EEG) and magnetoencephalography (MEG)] techniques have provided efficient and non-invasive ways to map the patterns of structural and functional connectivity of the human brain:

- Structural brain connectivity is referred to as the anatomical connection pattern between different neuronal elements.

- functional brain connectivity is referred to as the functional association pattern among brain regions, which can be obtained by measuring the temporal correlations between spatially remote neurophysiological events from fMRI and EEG/MEG data.

Recent applications of brain connectivity networks include exploring the anatomical and functional connectivity relationship between brain regions and also the connectivity abnormality in neurodegenerative diseases (e.g., MCI and AD) for identifying biomarkers for diagnosis.
It is reported that structural and functional abnormalities can be observed in the brains of AD and MCI. Recent studies have suggested that, in addition to the regional disturbance of brain structure and function, neurodegenerative diseases are also associated with the abnormalities in connections between different brain regions. Network analysis provides a new way for exploring the association between brain functional deficits and the underlying structural disruption related to brain disorders. Due to the increasing reliability of network characterization through neurobiological meaningful and computationally efficient measures, learning connectivity characteristics of network from neuroimaging data shows great promise for identifying image-based biomarkers. Recently, connectivity networks have been used for analysis of AD and MCI. Applications of network-based analysis tools in neuroimaging can be divided into two categories:

- Studies focusing on specific hypothesis-driven tests, for example, on the small-world network, default mode, and hippocampus network.
- Studies focusing on machine learning based methods for individual-based classification.

In the first category, studies mainly focus on network dysfunction perspective of neurodegenerative diseases using graph theoretical analysis, to demonstrate the topological differences of the brain networks between patients and NC. While these studies in general support the hypothesis of disconnection syndrome in AD and MCI, they cannot be automatically used to discriminate MCI and AD from NC at individual level. However, in the second category, machine learning methods are used to train classification models to identify diseased subjects from NC[20][7][15][18].

2.3 Proposed method

The method proposed in the article uses as dataset the functional connectivity matrices associated to each subject. In order to take in account the different topology structures of connectivity networks a threshold approach is applied. Specifically 5 different level of thresholds are applied. Subsequently, following the general machine learning procedures for connectivity network analysis the feature extraction step followed by 2 different future selection methodologies are used. Specifically the weighted clustering coefficient[23] is the algorithm used to extract the important features: it permitted a dimensionality reduction of the original data and at the same time it encapsulates the information of a single node as well as the weighted information of adjacent edges. The two feature selection methods belong to two different macro classes. The first one is the Ttest, that is a filter approach and the second one the RFE-GK is from the family of wrapper approach. The last step is the training of the classifier that in this case is a Multi-kernel SVM (Binary classification) since it showed in past literature a better classification accuracy.[8][9]. In order to improve the framework's generalizability and tuning the hyperparameters for ttest and multikernels a LOOCV nested cross validation methods is used.

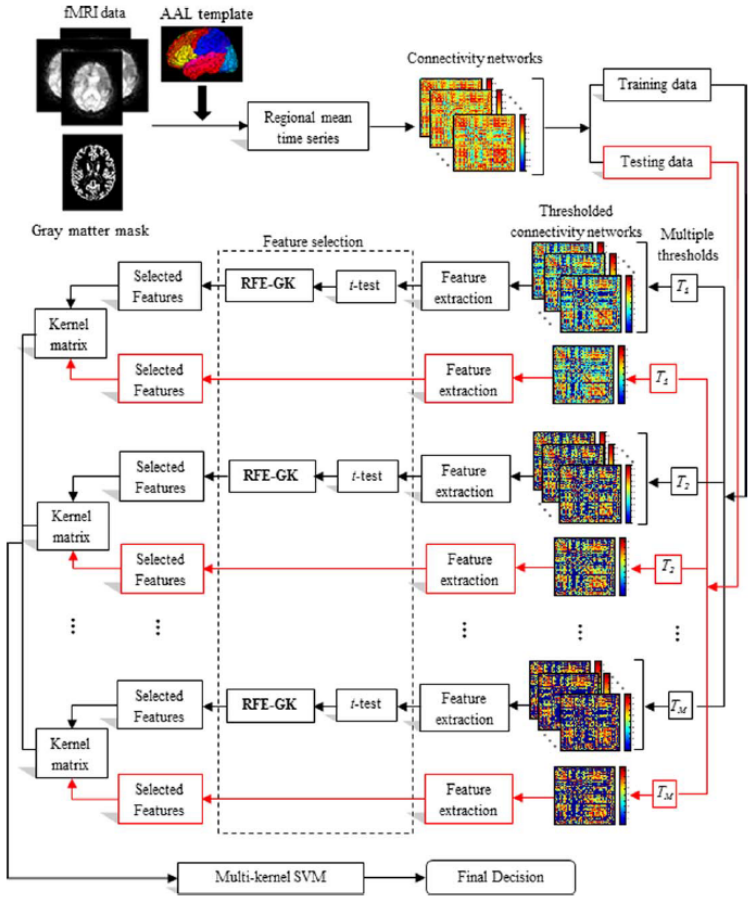


Figure 2.1: General framework of the proposed method for detect the MCI

2.3.1 Data acquisition

In the study, 12 amnestic MCI patients and 25 NC were recruited. Demographic information of the participants is shown in fig 2.2. Informed consent was obtained from

TABLE I. Characteristics of the participants used in this study

Group	MCI	Normal
No. of subjects (male/female)	6/6	9/16
Age (mean ± SD)	75.0 ± 8.0	72.9 ± 7.9
Years of education (mean ± SD)	18.0 ± 4.1	15.8 ± 2.4
MMSE (mean ± SD)	28.5 ± 1.5	29.3 ± 1.1

Figure 2.2: Characteristics of the subjects involved in the paper

all participants, and the experimental protocols were approved by the institutional ethics board. All the recruited subjects were diagnosed by expert consensus panels. Data acquisition was performed using a 3.0-Tesla GE Signa EXCITE scanner. FMRI images of each participant were acquired with the following parameters: flip angle=77°, TR/TE=2000/32 ms, imaging matrix=64X64, FOV=256 X 256 mm^2, 34 slices, 150 volumes, and voxel thickness=4 mm. During scanning, all subjects were instructed to keep their eyes open and stare at a fixation cross in the middle of the screen, which lasted for 5 min.

The preprocessing steps of the fMRI images, which include slice timing correction

and head-motion correction, were performed using Statistical Parametric Mapping software package (SPM8, http://www.fil.ion.ucl.ac.uk.spm). The first 10 acquired fMRI images of each subject were discarded to ensure magnetization equilibrium. The remaining 140 images were first corrected for the acquisition time delay among different slices before they were realigned to the first volume of the remaining images for head motion correction. Since the regions of ventricles and WM (white matter) contain a relatively high proportion of noise caused by the cardiac and respiratory cycles, only BOLD signals were used, extracted from gray matter (GM) tissue to construct functional connectivity network. Accordingly, the T1-weighted image of each subject is firstly segmented into GM, WM, and cerebrospinal fluid (CSF). GM tissue of each subject was then used to mask their corresponding fMRI images to eliminate the possible effect from WM and CSF in the fMRI time series. The first scan of fMRI time series was coregistered to the T1-weighted image of same subject. The estimated transformation was then applied to other fMRI scans of the same subject. The brain space of fMRI scans for each subject was then parcellated into 90 ROIs by warping the automated anatomical labeling template to the subject space using the deformation fields estimated via a deformable registration method called HAMMER. For each subject, the mean time series of each individual ROI was then computed by averaging the GM-masked fMRI time series over all voxels in the particular ROI. In this study, the GM-masked mean time series of each region was band-pass filtered within frequency interval $0.025 =< f =< 0.100Hz$ since the fMRI dynamics of neuronal activities are most salient within this frequency interval[20].

2.3.2 Processing raw features

From the mean time series of each individual ROI, the functional connectivity between each couple of ROIs is obtained by adopting the Pearson correlation coefficient. So for each subject a functional connectivity matrix is built (more details about connectivity matrix in appendix A) and it is the mathematical representation of a graph where vertices corresponding to the ROIs and the weight of edges corresponding to the correlation coefficients.

2.3.2.1 Pearson correlation coefficient

The Pearson correlation coefficients are adopted to compute the functional connectivity between the ROI pairs. A correlation coefficient is a number that quantifies some type of correlation and dependence, meaning statistical relationships between two or more random variables or observed data values. Pearson's correlation coefficient is known as the best method of measuring the association between variables of interest because it is based on the method of covariance. It gives information about the magnitude of the association, or correlation, as well as the direction of the relationship. The Pearson's correlation coefficient for two samples X, Y is referred with the letter r (rho is for two or more populations). it is the covariance of the two variables divided by the product of their standard deviations. The form of the definition involves a "product moment", that is, the mean (the first moment about the origin) of the product of the mean-adjusted random variables; hence the modifier product-moment in the name. Given one dataset $X = x1, ..., xn$ containing n values and another dataset $Y = y1, ..., yn$ containing n values then that formula

for r is:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where:

- n, x_i, y_i are defined as above
- $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$(the sample mean); and analogously for $\bar{y} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

The properties of the correlation coefficient are:

- Limit: Coefficient values can range from +1 to -1
- Pure number: It is independent of the unit of measurement. For example, if one variable's unit of measurement is in inches and the second variable is in quintals, even then, Pearson's correlation coefficient value does not change.
- Symmetric: Correlation of the coefficient between two variables is symmetric. This means between X and Y or Y and X, the coefficient value of will remain the same.

A value of 1 implies that a linear equation describes the relationship between X and Y perfectly, with all data points lying on a line for which Y increases as X increases. A value of -1 implies that all data points lie on a line for which Y decreases as X increases. A value of 0 implies that there is no linear correlation between the variables.

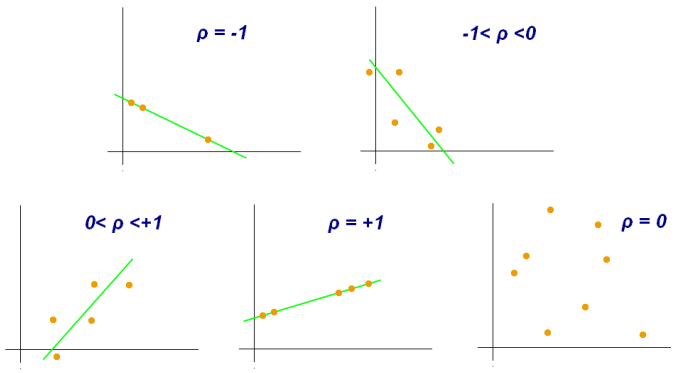


Figure 2.3: Graphical representation of Pearson coefficient

2.3.2.2 Fisher r to z transformation

Fisher's r-to-z transformation was applied on the elements of the functional connectivity network (matrix) to improve the normality of the correlation coefficients. As the name suggest the Fisher R to Z transformation is a function f that transform some data point x_i to another point $y_i = f(x_i)$. In statistics this transformations are very useful to improve the interpretability or appearance of graphs and to obtain a different set of data that are more easy to use. In particular, the fisher transformation is highly connected with the Pearson correlation coefficient because the Pearson correlation coefficient r is not Normally distributed but its density function is more

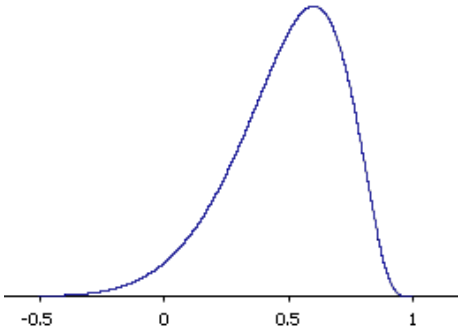


Figure 2.4: Distribution's shape of Pearson correlation coefficient

likely a compressed bell. To improve the interpretability of r we would like to have a Gaussian distribution. We reach this aim by using the Fisher r to z transformation defined as:

$$z = \frac{1}{2} \ln \left(\frac{1+r}{1-r} \right) = \tanh^{-1} r$$

Where r is the Pearson correlation coefficient. So now our variable z is normally distributed with a well-known standard deviation of:

$$\sigma_z = \frac{1}{\sqrt{n-1}}$$

This transformation belongs to the class of Variance-stabilizing transformation as possible to seen from the preceding formula. Its variance depends only on the number of samples and not on some parameter (like the mean) correlated to the population from the sample come.

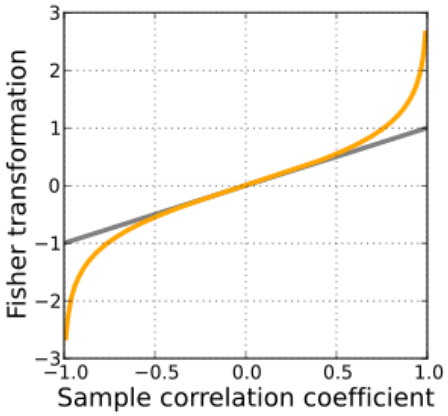


Figure 2.5: A graph of the transformation (in orange). The untransformed sample correlation coefficient is plotted on the horizontal axis, and the transformed coefficient is plotted on the vertical axis. The identity function (gray) is also shown for comparison.

Moreover, to extract the meaningful network measures, all negative correlations have been removed from the obtained functional connectivity networks.

2.3.3 Feature extraction

2.3.3.1 threshold

The first step after the pre-processing is threshold the connectivity network using five different thresholds to get different representation of different characteristics of the brain. In functional connectivity networks, the connectivity describes frequency-dependent correlation between spatially distinct brain regions. Some weak and potentially insignificant connections for identifying patients from controls could obscure the network topology, when considered together with strong and important connections. Thus, it may be important to discard these connections by using a thresholding approach. Moreover, the thresholded networks are often simpler to characterize and thus have more easily defined models for statistical comparison. However, the thresholds are often arbitrarily determined, and the optimal threshold can only be determined after exploring the network properties over a broad range of plausible thresholds. On the other hand, the network with different thresholds may represent different level of topological properties, and these properties may be complementary to each other in improving the classification performance[19]. Therefore, in the current study, a multiple-threshold method is adopted to reflect the multiple levels of network properties. Specifically, given a threshold $T_m(m = 1, \ldots, M)$, the connectivity network $G = [\tau_{ij}]nxn$ is thresholded as:

$$\tau_{ij}^{m} = \left\{ \begin{array}{ll} 0, & \text{if } \tau_{ij} \leq T_m \\ \tau_{ij}, & \text{otherwise} \end{array} \right\}$$

where τ_{ij} denotes the connection weight between the ith and jth network-nodes/ROIs.

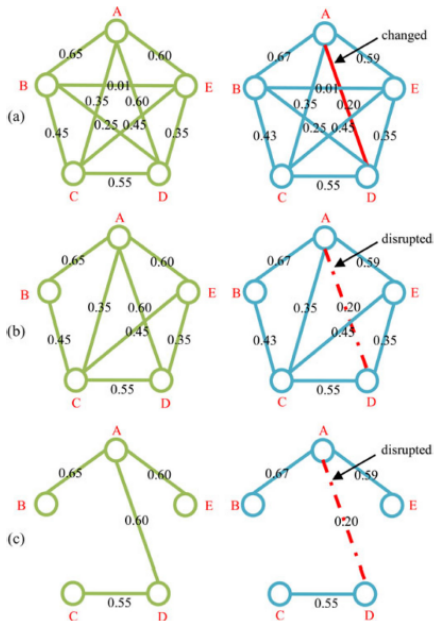


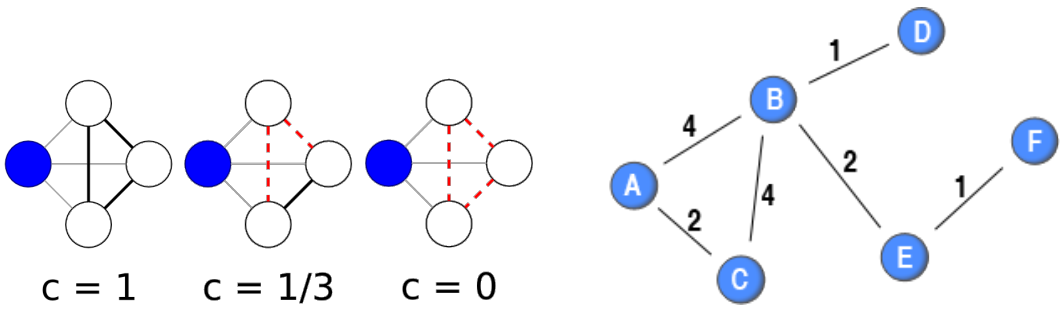
Figure 2.6: Examples of multiple-thresholded connectivity networks for NC (green) and MCI (blue) patients. (a) Original functional connectivity networks, (b) thresholded connectivity networks with T=0.3, and (c) thresholded connectivity networks with T=0.5.

2.3.3.2 clustering coefficient

After thresholding the matrix, the next step is collect information into feature vectors, a lot of different information can be extrapolated from the connectivity matrix, each one representative of a different property of the brain. The analyzed paper used the "local clustering coefficient". As explained the brain of patience involved in the paper is modelled by using a connectivity matrix that represent characteristics between the anatomical and functional connection of different brain area. If the aim is to extrapolate a set of features from such connectivity networks is necessary to focus the important characteristics that we want to extrapolate. An important measure of brain capability is the functional segregation. Functional segregation in the brain is the ability for specialized processing to occur within densely interconnected groups of brain regions. Measures of segregation primarily quantify the presence of such groups, known as clusters or modules, within the network. Measures of segregation have straightforward interpretations: in anatomical networks suggests the potential of functional segregation, while the presence of clusters in functional networks suggests an organization of statistical dependencies indicative of segregated neural processing. Traditionally, the two versions of the clustering coefficient developed for testing the tendency of nodes to cluster together into tightly knit groups are the global clustering coefficient and the local clustering coefficient. The global version was designed to give an overall indication of the clustering in the network, whereas the local gives an indication of the embeddedness of single nodes. The focus in this thesis is on the second kind. The clustering coefficient of a node ν_i quantifies its ability to form a complete graph among its neighbors where all nodes are connected to one another.
The common cluster coefficient doesn't take in account the weights of a graph, so the literature presents various attempts at developing or implementing a clustering coefficient measure designed for weighted networks[21][3][23]. In the paper the used Clustering coefficient is the one proposed by Rubinov and Sporns (2010). The formula is:

$$f_i = \frac{2}{d_i(d_i-1)} \sum_{j,k} (w_m(i,j)w_m(j,k)w_m(k,i))^{1/3}$$

where d_i is the number of neighboring node around node i w_m is the weight of edge and i, j, k represent the considered triplet of neighboring nodes.



(a) Clustering coefficient for unweighted graphs (b) Weighted graphs

Figure 2.7: unweighted (a) and weighted (b) graphs

considered the figure 2.7(b) we will have:

- A: Clustering coefficient=$2\sqrt[3]{32}$

- C: Clustering coefficient=$2\sqrt[3]{32}$
- B: Clustering coefficient=$\sqrt[3]{32}/3$
- E: Clustering coefficient=0
- F: Clustering coefficient=0
- D: Clustering coefficient=0

2.3.3.3 Standardization

After obtained the feature and collected into a feature vector for each subject, a common feature normalization is adopted. For each extracted feature f_i this standardization is applied:

$$f_i = \frac{f_i - \bar{f}_i}{\sigma_i}$$

Where f_i and σ_i are respectively the mean and standard deviation of the i-th feature across all subjects.

2.3.4 Feature selection

For what concern the feature selection step, in the paper are applied two techniques:

- T-test for feature selection first (and 5 values of p are used as parameters for cross validation)
- RFE-GK

The first one belong to the class of filter methods, so it is fast and independent from the learning algorithm used, the second belong to wrapper method so it is computationally expensive and dependent from the classification algorithm employed.

2.3.4.1 T-test

In inferential statistics to prove a correlation or a dependency between two or more groups two hypotheses are set up. H0: the null hypothesis: no difference between groups H1: the alternative hypothesis: there is difference between groups It's impossible to demonstrate mathematically the validity of one of the two hypothesis. By default, the null hypothesis is considerate valid at the beginning. So the research is focused on demonstrating under a certain probability if the null hypothesis is rejected or not. If it is rejected the alternative hypothesis will be valid. Each time you reject a hypothesis there is a chance you made a mistake. There are two different errors: Type1: you incorrectly rejecting the null hypothesis, alpha is the probability of such error Type2: you fail to reject the null when you should have rejected the null hypothesis, beta is the probability of such error Another important parameter is the p-Value: the probability of obtaining result at least as extreme as the current one, assuming null is true. In other words, it is a measurement to tell us how much the observed data disagrees with the null hypothesis. p-value small: there is more disagreement of our data with the null hypothesis p-value big: there

is less disagreement of our data with the null hypothesis The real T-test is a statistical method also used to decide if a feature of two groups is discriminatory for the groups (by confronting their mean value). The T-test can be applied on samples of a population assumed to have a normal distribution. The two samples will have a T-student distribution (which tend to be a Gaussian distribution if the number of samples are infinite). By brutally confronting the two mean value of the sample we'd not obtain a significant difference between the samples, so a kind of signal to ratio parameter. Defined as:

$$t = \frac{\text{Signal}}{\text{Noise}} = \frac{\text{difference between group means}}{\text{variability of groups}} = \frac{|x_1 - x_2|}{\sqrt{(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2})}}$$

where

- x_1 and x_2 =average mean of samples of each group
- n_1 and n_2 =the number of samples for each group
- s_1^2 and s_2^2 =variance of samples of each group

The signal is the difference between groups, so the mean difference and is a measure of signal meanwhile the variance of them represent a measure of noise. Once we

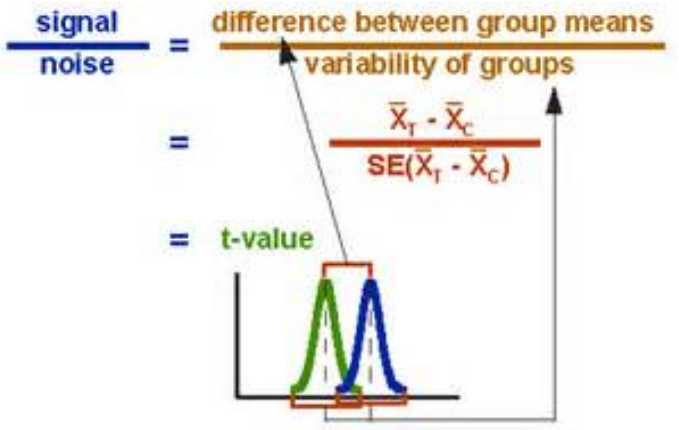


Figure 2.8: graphical representation of the t value

have the t value we should be able to determine if the two groups are different or not (respect the samples considered) under a certain probability. This probability is the p-value that represent the probability of non rejecting the null hypothesis if two other samples of same population are considered. There could be statistically connection with the two group by confronting the t-value obtained with t-critical value obtained from the t-table. The t-table is a table where are explicated the critical values of the t-test, function of p-value and degree of freedom.
The algorithm is: Chosen the p-value (it depends on how much we want that the results be statistically relevant) and calculated the dof=(n_1+n_2)-2 we obtain the t-critical value (we have also to choose double or single tailed). If t-value>t-critical value then there is a statistically significant difference between the samples. From figure 2.9 is possible to image the t-value as a t-student distribution, the p-value as the area under the tails of that distribution and t-critical as the limiting points of

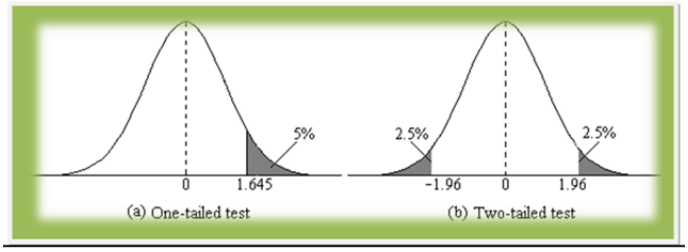


Figure 2.9: t-value distribution with a p-value=0.5% one-tailed test

the p-value area. The white area represent the probability of non rejecting the null hypothesis and the grey area the probability of rejecting. If we are sure of the sign of our t-value, we can choose the one tailed test instead of a two tailed one. The limitation of this approach are:

- The results of inferential statistics can only be applied to population that resemble the sample that was tested
- The sample and population should be roughly normal in distribution. This means most scores will be around the mean with fewer scores further out, resembling a bell curve
- Each group should have about the same number of data points. Comparing large and small groups together may give inaccurate results.
- All data should be independent. This means the scores should not be influenced by each other

2.3.4.2 Graph kernel

Machine learning in domains such as bioinformatics, drug discovery, web data mining and social networks involves the study of relationships between structured objects. Graphs are natural data structures to model such structures, with nodes representing objects and edges the relations between them. Also in this study graph is the perfect structure to represent ROIs and how are the links between them. Kernels represent a function of similarity between two object (major details in section "Kernels"), so comparing nodes in a graph involves constructing a kernel between nodes, while comparing graphs involves constructing a kernel between graphs. In both cases, the challenge is to define a kernel that captures the semantics inherent in the graph structure and is reasonably efficient to evaluate. There exist many graph similarity measures based on graph isomorphism or related concepts such as subgraph isomorphism or the largest common subgraph. Because the most natural measure of similarity of graphs is to check whether the graphs are topologically identical, i.e., isomorphic. The exact definition of isomorphism between two graph is: *Find a mapping f of the vertices of G1 to the vertices of G2 such that G1 and G2 are identical; i.e. (x,y) is an edge of G1 if (f(x),f(y)) is an edge of G2. Then f is an isomorphism, and G1 and G2 are called isomorphic.*
The problem is that no polynomial-time algorithm is known for graph isomorphism, neither is it known to be NP-complete. The concept of Subgraph isomorphism asks if there is a subset of edges and vertices of G1 that is isomorphic to a smaller

graph G2 but also in this case the Subgraph isomorphism is NP-complete. It is also proven that computing a Graph Kernel is at least as hard to compute as deciding if graphs are isomorphic. Therefore, one usually restricts graph kernels to compare only specific types of subgraphs that are computable in polynomial runtime.

Some definitions

A graph G is defined as a triplet (V, E, l), where V is the set of vertices, E is the set of undirected edges, and $l : V \rightarrow \Sigma$ is a function that assigns labels from an alphabet Σ to nodes in the graph . The neighbourhood N (v) of a node v is the set of nodes to which v is connected by an edge, that is $N(v) = v'|(v, v') \in E$. For simplicity, we assume that every graph has n nodes, m edges, and a maximum degree of d. The size of G is defined as the cardinality of V. A walk is a sequence of nodes in a graph, in which consecutive nodes are connected by an edge. A path is a walk that consists of distinct nodes only. A (rooted) subtree is a subgraph of a graph, which has no cycles, but a designated root node. A subtree of G can thus be seen as a connected subset of distinct nodes of G with an underlying tree structure. The height of a subtree is the maximum distance between the root and any other node in the subtree. Just as the notion of walk is extending the notion of path by allowing nodes to be equal, the notion of subtrees can be extended to subtree patterns which can have nodes that are equal. These repetitions of the same node are then treated as distinct nodes, such that the pattern is still a cycle-free tree. All subtree kernels compare subtree patterns in two graphs, not (strict) subtrees.

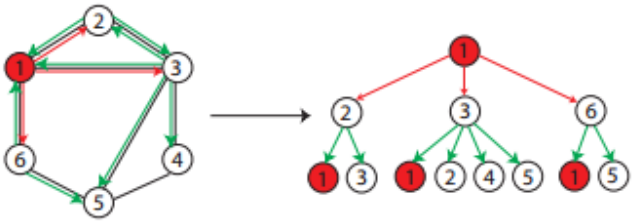


Figure 2.10: A subtree pattern of height 2 rooted at the node 1. Note the repetitions of nodes

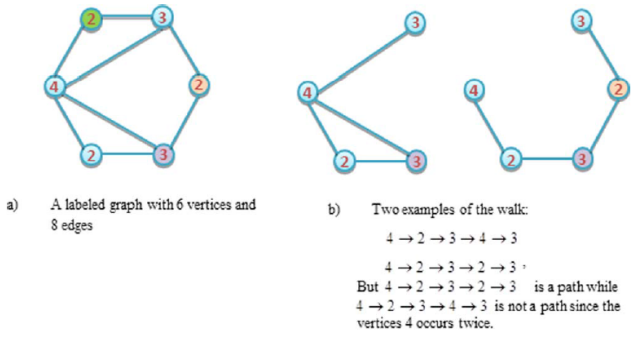


Figure 2.11: Example of walks

Several different graph kernels have been defined in machine learning which can be categorized into three classes:

- graph kernels based on walks and paths: compute the number of matchings of pairs of random walks in two graphs. The standard formulation of the random walk kernel, based on the direct product graph of two graphs, is computable in $O(n^6)$ for a pair of graphs.

- graph kernels based on limited-size subgraphs: also called graphlets, represent graphs as counts of all types of subgraphs of size $k \in 3, 4, 5$.

- graph kernels based on subtree patterns: iteratively compares all matchings between neighbors of two nodes v from G and v' from G'.

The Weisfeiler-Lehman test of isomorphism

The graph kernel used in the analyzed paper is based on the concept of Weisfeiler-Lehman test of isomorphism[28] in its 1-dimensional variant, also known as "naive vertex refinement", and the technique belongs to "graph kernels based on subtree patterns". The key idea of the algorithm is to augment the node labels by the sorted set of node labels of neighboring nodes, and compress these augmented labels into new, short labels. These steps are then repeated until the node label sets of G and G' differ, or the number of iterations reaches n. The algorithm works by execution of 4 steps at each iteration i. The Weisfeiler-Lehman algorithm terminates after step 4 of iteration i if $l_i(\nu)|\nu \in V \neq l_i(\nu')|\nu' \in V'$, that is, if the sets of newly created labels are not identical in G and G'. The graphs are then not isomorphic. If the sets are identical after n iterations, it means that either G and G' are isomorphic, or the algorithm has not been able to determine that they are not isomorphic. As a side note, is mentioned that the 1-dimensional Weisfeiler-Lehman algorithm has been shown to be a valid isomorphism test for almost all graphs.

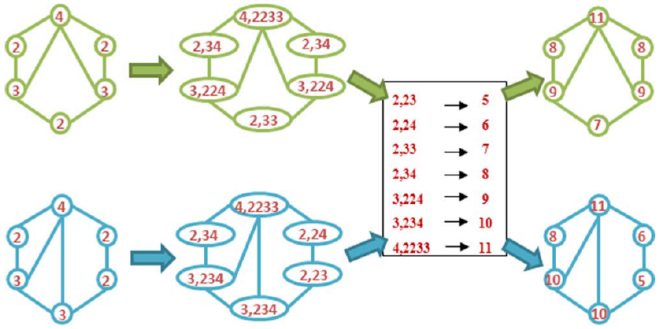


Figure 2.12: Weisfeiler-Lehman algorithm for isomorphism with the maximum number of iteration equals to one

Algorithm explanation:

- Step 1: if it's the first iteration for each node in the graph set a label with the number of its neighbors, if it's not the first iteration the number of the label is already determined at step 4 of the iteration i-1

- Step 2: for each label associate a string that represent a decimal number where the integer part is the value of the label and decimal part is the ascending order of label of neighbors

- Step 3: Ordinate each part of the string representative of decimal number and map each string to a new value of the label. Such a map associate to each string in ascendant order a number each time bigger of 1. The first value of the number that you have to associate is equal to the biggest value +1 of the label at step 1.

- Step 4: Relabel each node with the new values found in Step 3

Note that the compressed labels $l_i(v)$ correspond to subtree patterns of height i rooted at v. The runtime complexity of the 1-dimensional Weisfeiler-Lehman algorithm with h iterations is O(hm).

The real graph kernel

Define the Weisfeiler-Lehman graph at height i of the graph $G = (V, E, l) = (V, E, l_0)$ as the graph $G_i = (V, E, l_i)$. We call the sequence of Weisfeiler-Lehman graphs $G0, G1, ..., Gh = (V, E, l_0), (V, E, l_1), ..., (V, E, l_h)$, where $G_0 = G$ and $l_0 = l$, the Weisfeiler-Lehman sequence up to height h of G. G_0 is the original graph, $G_1 = r(G_0)$ is the graph resulting from the first relabeling, and so on. Note that neither V, nor E ever change in this sequence, but it is defined as a sequence of graphs rather than a sequence of labeling functions for the sake of clarity of definitions that follow. Let k be any kernel for graphs, that will be called the base kernel. Then the Weisfeiler-Lehman kernel with h iterations with the base kernel k is defined as $k_{WL}^{(h)}(G, G') = k(G_0, G'_0) + k(G_1, G'_1) + ... + k(G_h, G'_h)$, where h is the number of Weisfeiler-Lehman iterations and $G_0, ..., G_h$ and $G'_0, ..., G'_h$ are the Weisfeiler-Lehman sequences of G and G' respectively. Let the base kernel k be any positive semidefinite kernel on graphs. Then the corresponding Weisfeiler-Lehman kernel $k_{WL}^{(h)}$ is positive semidefinite. Let G and G' be graphs. Define $\Sigma_i \subseteq \Sigma$ as the set of letters that occur as node labels at least once in G or G' at the end of the i-th iteration of the Weisfeiler-Lehman algorithm. Let Σ_0 be the set of original node labels of G and G'. Assume all Σ_i are pairwise disjoint. Without loss of generality, assume that every $\Sigma_i = \sigma_{i1}, ..., \sigma_{\Sigma_i}$ is ordered. Define a map $c_i : \{G, G'\} \times \Sigma_i \to N$ such that $c_i(G, \sigma_{ij})$ is the number of occurrences of the letter σ_{ij} in the graph G. The Weisfeiler-Lehman subtree kernel on two graphs G and G' with h iterations is defined as:

$$k_{WLsubtree}^{(h)}(G, G') =< \varphi_{WLsubtree}^{(h)}(G), \varphi_{WLsubtree}^{(h)}(G') >$$

$$\varphi_{WLsubtree}^{(h)}(G) = (c_0(G, \sigma_{01}), ..., c_0(G, \sigma_{0|\Sigma 0|}), ..., c_h(G, \sigma_{h1}), ..., c_h(G, \sigma_{h|\Sigma h|}))$$

$$\varphi_{WLsubtree}^{(h)}(G') = (c_0(G', \sigma_{01}), ..., c_0(G', \sigma_{0|\Sigma 0|}), ..., c_h(G', \sigma_{h1}), ..., c_h(G', \sigma_{h|\Sigma h|}))$$

That is, the Weisfeiler-Lehman subtree kernel counts common original and compressed labels in two graphs. The Weisfeiler-Lehman subtree kernel on a pair of graphs G and G0 can be computed in time O(hm). For N graphs, the Weisfeiler-Lehman subtree kernel with h iterations on all pairs of these graphs can be computed in O(Nhm + N2hn). Figure 2.13 represents the Weisfeiler-Lehman test for isomorphism plus the calculus of the Weisfeiler-Lehman subtree kernel, that is the used kernel in the paper. In rough words when finished the test for isomorphism we define the transformation phi of each graph as a vector with the numbers of occurrence of

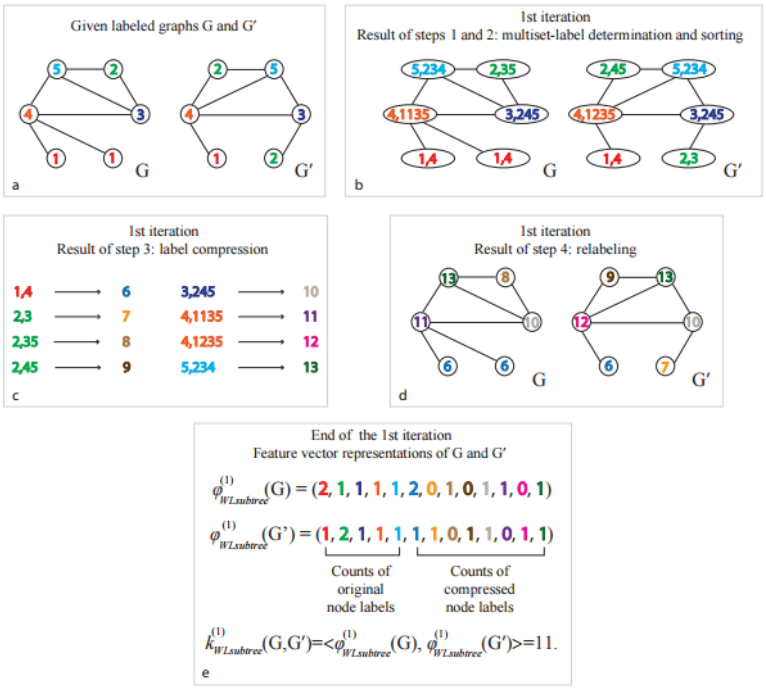


Figure 2.13: Illustration of the computation of the Weisfeiler-Lehman subtree kernel with h=1

the same label for each iteration i. The order of occurrence is follows the ascending order of labels.

$$L = (1, 2, 3, 4, 5|6, 7, 8, 9, 10, 11, 12, 13|...)$$
$$\varphi(G) = (2, 1, 1, 1, 1|2, 0, 1, 0, 1, 1, 0, 1|...)$$
$$\varphi(G') = (1, 2, 1, 1, 1|1, 1, 0, 1, 1, 0, 1, 1|...)$$
$$K =< \varphi(G), \varphi(G') >$$

Where each | represent and iteration. Notice that the example in figure 2.13 doesn't need any iteration because the set of label for each graph is already different at the beginning of algorithm.

2.3.4.3 Rfe-gk

The recursive features elimination is a technique used for features selection. RFE belongs to the "Wrapper methods" consider the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. A predictive model is used to evaluate a combination of features and assign a score based on model accuracy. When each feature has its score then we are able to choose the best ones. The right number of feature could be detected by using other method for classification (i.e. cross validation and so on). The search process may be methodical, stochastic or it may use heuristics. In figure 2.14 the general scheme of the RFE:

In the analyzed paper the model used is the SVM with graph-kernel and the evaluation of the importance is the average of the accuracy.

Essentially the algorithm in figure 2.17 takes as input the sub-connectivity matrix (generated from the survived ROIs of the preceding ttest-selection), then removes

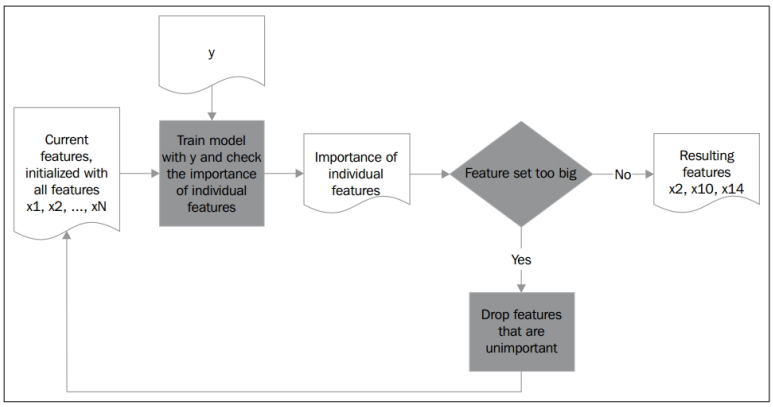


Figure 2.14: general scheme of RFE

one by one the remained features (represented by ROIs in matrix), build up the train and test graph kernel matrix for each of connectivity networks without the removed feature and then evaluate the classification accuracy using SVM with the graph kernel matrix generated before. After that it confronts the classification accuracy of each connectivity networks, finds the one with best accuracy and save the removed feature of that connectivity matrix into a new array. This procedure is repeated until the ROIs are all removed from the initial set. The selected best features are the ones saved in the new array with best average accuracy.
Important details of the procedure:

- The criteria for choosing the right number of features is the average value of the accuracy of features present in the Ranked feature list F until the current step.

Ranked feature list F	**Average accuracy list U**
Feature 5 (best of step 1)	Accuracy step 1
Feature 2	Accuracy step 2
Feature 1	Accuracy step 3
Feature 4	Accuracy step 4
Feature 6	Accuracy step 5
Feature 3 (best of step 6)	Accuracy step 6

Best accuracy

Figure 2.15: Choosing best features in RFE-GK

- An LOOCV for evaluate the accuracy of each built SVM

- The used predictive model is a SVM classifier with a Graph Kernel method for training and testing

Kernel graph matrix built with training subjects			
K(tr1,tr1)	K(tr1,tr2)	K(tr1,tr3)	K(tr1,tr4)
K(tr2,tr1)	K(tr2,tr2)	K(tr2,tr3)	K(tr2,tr4)
K(tr3,tr1)	K(tr3,tr2)	K(tr3,tr3)	K(tr3,tr4)
K(tr4,tr1)	K(tr4,tr2)	K(tr4,tr3)	K(tr4,tr4)

Kernel graph vector for Testing
K(te1,tr1)
K(te1,tr2)
K(te1,tr3)
K(te1,tr4)

Figure 2.16: Example of training and test kernel matrix for 4 training subjects and 1 test subject

- The use of connectivity trhesholded matrix and survived features at each step for building the subnetworks for graph kernel

The two matrix are needed as input for the SVMtrain and SVMpredict function of the SVM library to train and test a SVM.

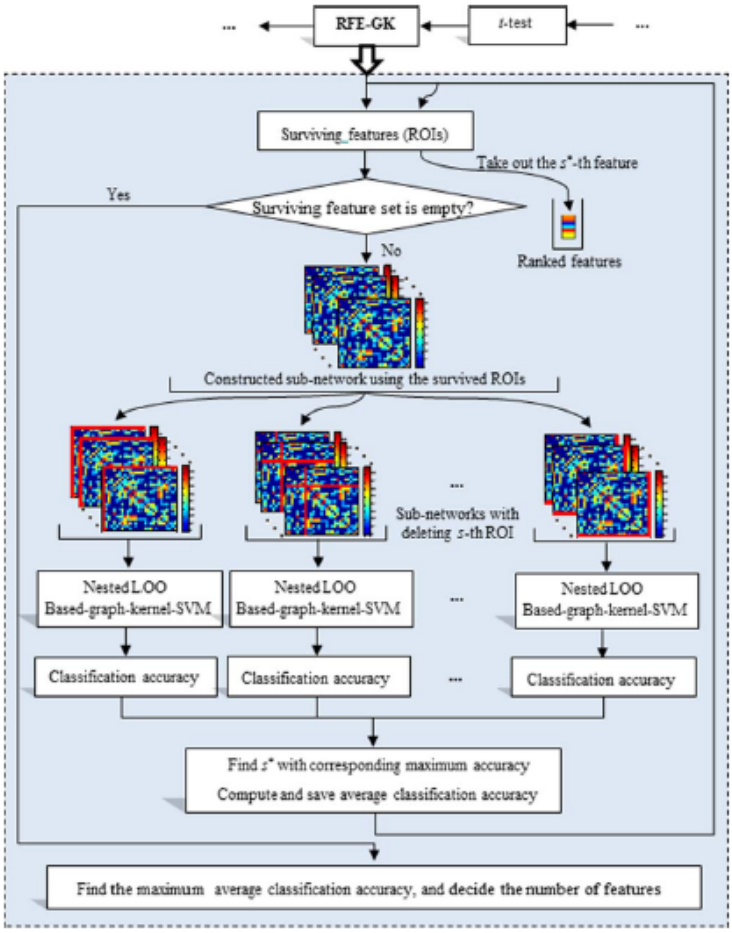


Figure 2.17: scheme of the RFE-GK used in the paper

2.3.5 Classifier

2.3.5.1 SVM

Hard margin

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary classifier. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a new data point will be in. In the case of support vector machines, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a p-1-dimensional hyperplane. This is called a linear classifier. If we have different p-vector (feature vector) and each p-vector can belong only at one of two classes, we can have the situation depicted in picture 2.19 (with p=2): The

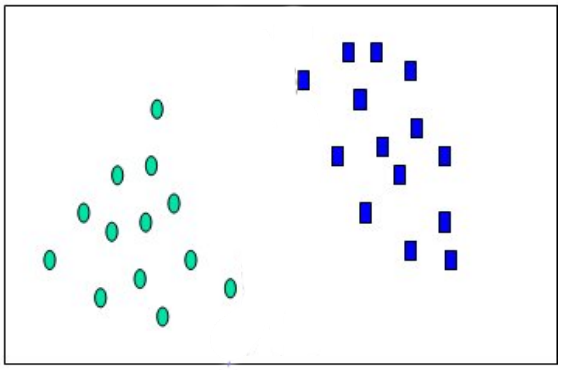


Figure 2.18: two different classes (p=2)

main idea to separate this two class is to build a plane (hyperplane if p¿2) able to discriminate if a vector belongs to one class or to another. Such hyperplane can be written as the set of points x satisfying:

$$\bar{w} \cdot \bar{x} + b = 0$$

Where $\bar{x}$ is a point in the space and $\bar{w}$ is a vector normal to the hyperplane. Geometrically the $\bar{w}$ represent the orientation of the hyperplane and b the displacement in the space (respect to the origin) If the training data are linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible. The region bounded by these two hyperplanes is called the "margin", and the maximum-margin hyperplane is the hyperplane that lies halfway between them. These hyperplanes can be described by the equations:

$$\bar{w} \cdot \bar{x} - b = 1$$

$$and$$

$$\bar{w} \cdot \bar{x} - b = -1$$

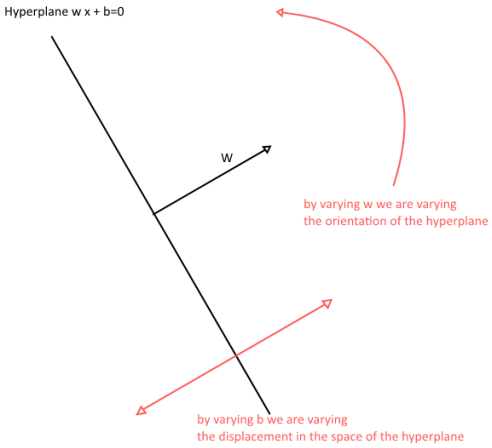


Figure 2.19: hyperplane and its parameters, the orthogonal vector w and the scalar coefficient b

Geometrically, the distance between these two hyperplanes is $\frac{2}{||\bar{w}||}$ so to maximize the distance between the planes we want to minimize $||w||$. As we also have to prevent data points from falling into the margin, we add the following constraint: for each i either

$$\bar{w} \cdot \bar{x} - b \geq 1 \text{ if } y_i = 1$$

or

$$\bar{w} \cdot \bar{x} - b \leq 1 \text{ if } y_i = -1$$

These constraints state that each data point must lie on the correct side of the margin. This can be rewritten as:

$$y_i(\bar{w} \cdot \bar{x} - b) \geq 1 \text{ for all } 1 \leq i \leq n$$

Putting this together the optimization problem can be stated: "Minimize $||\bar{w}||$ subject to $y_i(\bar{w} \cdot \bar{x} - b) \geq 1$ for i=1,...,n"
The $\bar{w}$ and b that solve this problem determine our classifier: $\bar{x} \rightarrow sgn(\bar{w} \cdot \bar{x} - b)$
Minimizing $||\bar{w}||$ is the same as minimizing $\frac{1}{2}||\bar{w}||^2$. In this form The above is an optimization problem with a convex quadratic objective and only linear constraints (this means that is solvable). Its solution gives us the optimal margin classifier. This optimization problem can be solved using commercial quadratic programming (QP) code. By using the Lagrange function we can state the equation:

$$\bar{w} = \sum_{1}^{n} \alpha_i y_i x_i$$

where the $\bar{\alpha}$ vector is the lagrangian multipliers.
The lagrangian problem is solvable and is possible to find the vector $\bar{\alpha}$ and so also the vector $\bar{w}$. So the classifier is built up. As result is obtained that the only α that are bigger than 0 are the α connected to the support vector: $\alpha_i \geq 0 \rightarrow \bar{x_i}$ support vector. Support vectors are such points that lie exactly in the boundary

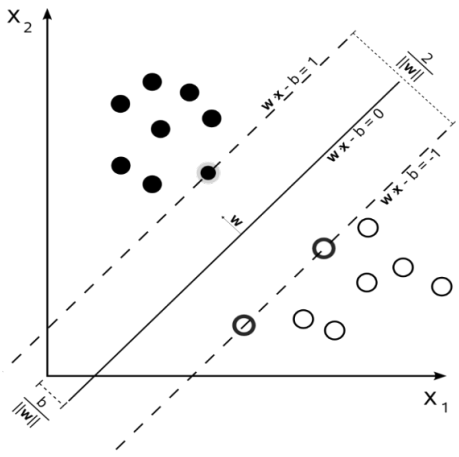


Figure 2.20: : hyperplane and its margins. the maximum distance between the two classes is equal to $2/||w||$

of the margin. Analyzing the results, we obtain that for determining the SVM is sufficient to know only the value of alpha connected with support vectors. Other points are useless.

Soft margin

In the previous analysis were analyzed only data points that are linearly separable but it's frequent to be in a situation where the data point overlaps the hypothetical perfect margin. In this case a soft margin is enforced and we "accept" to have few point that are that cross-over, or are closer to the boundary than the margin (outliers). To reach this aim slacks variable are introduced and the optimization problem can be rewritten as:

$$\begin{aligned} \underset{\bar{w}}{\text{minimize}} \quad & \frac{1}{2}||\bar{w}||^2 \\ \text{subject to} \quad & y_i - \bar{w} \cdot \bar{x} - b \leq \epsilon \\ & -y_i - \bar{w} \cdot \bar{x} + b \leq \epsilon \end{aligned}$$

and the situation is the one in figure 2.21. Only with this formulation we allow each

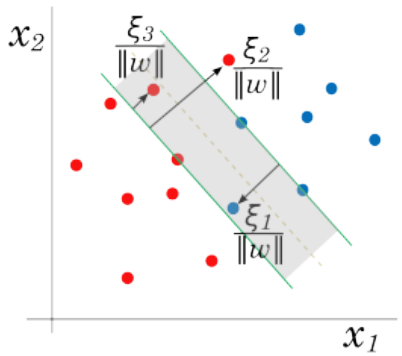


Figure 2.21: Graphical representation of hyperplane and the role of slack variables

point, also the ones with very big ϵ, to violate the margin so we introduce a penalty

for the error ϵ_i. Such a penalty is described by the scalar C, and the optimization problem can be rewritten as:

$$\begin{aligned} \min_{\bar{w},b} \quad & \frac{1}{2}||\bar{w}||^2 + C\sum_{i=1}^{n}\epsilon_i \\ \text{subject to } \forall i, \quad & y_i \cdot (\bar{w}\cdot\bar{x}+b) \leq 1-\epsilon \end{aligned}$$

The role of slack penalty C is:

- C=inf only want to w,b that separate the data
- C=0 can set ϵ_i to anything then w=0 (basically ignores the data)

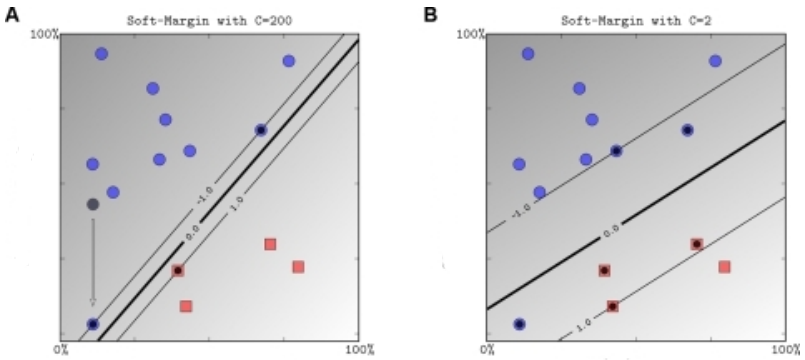


Figure 2.22: Comparison between two SVM with different C parameter

So more C is big more are penalized the ϵ_i more C is little more ϵ_i are tolerated.

Kernels

Imagine to have a space in which the data are highly not linearly separable. The main idea to solve the classification problem is to transform the non-linearly separable set of point into a linearly separable one (by using a hyperplane). This can be done in general by transforming the dataset $D = (x_i, \ldots, x_n)$ belongs to R^d to a bigger space R^D with $D > d$. Actually the necessary dimension of D to make the dataset linearly separable is unknown and it could be very very big. This idea is represented in the next picture where the dataset belong to R^2 space and are mapped into R^3 space by using a transformation φ. The most of the time the transformation φ is very difficult to find and so very hard to apply for our purpose. The solution to this inconvenient comes from dot product and kernel. In the construction of SVM the only operation involved is a dot product of vectors (points):

$$\text{Decision rule: } \sum_{1}^{n}\alpha_i y_i x_i \cdot x + b \geq 0$$

$\sum_1^n \alpha_i y_i x_i$ is the $\bar{w}$ vector,given x if the preceiding disequation is true is a class otherwise is the other
So to project this vectors into a higher space we apply the φ transformation:

$$\text{Decision rule: } \sum_{1}^{n}\alpha_i y_i \varphi(x_i) \cdot \varphi(x) + b \geq 0$$

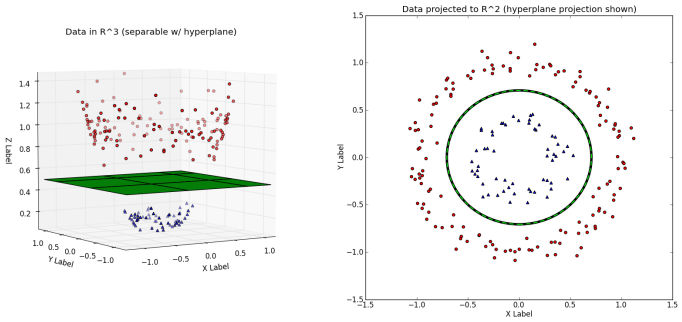


Figure 2.23: On the right side a non linear separable classes, on the left the transformation into a bigger space where that points are linearly separable

And we can notice that we are not really interested in the transformation function but in the dot product result in the new transformation space. But if we define a function K that:

$$K(x_1, x_2) = \varphi(x_1)\varphi(x_2)$$

We only need the original points and the function K, called Kernel function. At the end the equation for building the linear SVM can be rewritten as:

$$\text{Decision rule: } \sum_{1}^{n} \alpha_i y_i K(x_i, x) + b \geq 0$$

Where the final K is a matrix called Gram Matrix (o Kernel Matrix) and it is so composed: Of course not every function is a kernel function. Two mainly approaches

$$K = \begin{pmatrix} \mathcal{K}(x_1, x_1) & \mathcal{K}(x_1, x_2) & \dots & \mathcal{K}(x_1, x_T) \\ \mathcal{K}(x_2, x_1) & \mathcal{K}(x_2, x_2) & \dots & \mathcal{K}(x_2, x_T) \\ \vdots & \vdots & \ddots & \dots \\ \mathcal{K}(x_T, x_1) & \mathcal{K}(x_T, x_2) & \dots & \mathcal{K}(x_T, x_T) \end{pmatrix}$$

Figure 2.24: Kernel matrix structure

are followed to build a kernel:

- By construction: you work in simple space so you are able to check if the find function is a dot product of vector into another space
- By math properties:
  - It has to be symmetric: $K(x_1, x_2) = K(x_2, x_1)$
  - The Kernel Matrix has to be semi-positive definite (Mercer's condition)

To check if a function is a real kernel function by using math properties is not trivial, but luckily there are a lot of good working kernel that are already tested and that respect the math condition. The most famous and used are:

- linear: $K(x_i, x_j) = x_i^T x_j$
- polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$ with $\gamma > 0$

- radial basis function (RBF): $K(x_i, x_j) = e^{\left(\frac{||x_i - x_j||^2}{2\sigma^2}\right)}$ with $\sigma > 0$
- sigmoid: $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)$

Another way of consider the kernel function is a similarity function. If we consider that it represents the dot product of two vector into another space we can imagine that bigger is the result of the kernel function, bigger is the similarity between the two vector in the new space after transformation. This is possible if we remember that the dot product between two vector X Y is equal to: $||X||||Y|| \cos\theta$ Where theta is the angle between the two vectors. So $||X|| \cos\theta$ is the projection of X vector over Y vector and then multiplied by the Y norm. The dot vector result is zero if they are orthogonal or intuitively different it is maximum when the vectors are overlapped or intuitively similar.

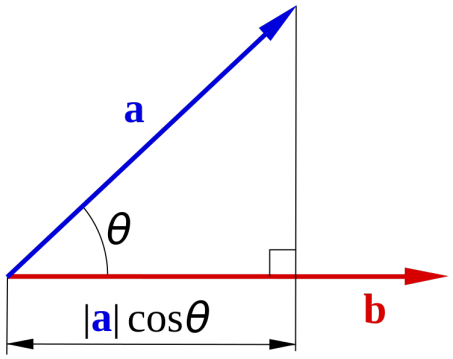


Figure 2.25: dot product in the space

Multikernel

In recent years, multiple kernel learning (MKL) methods have been proposed, where we use multiple kernels instead of selecting one specific kernel function and its corresponding parameters:

$$k_\eta(x_i, x_j) = f_\eta(k_m(x_i^m, x_j^m)_{m=1}^P)$$

where the combination function, $f_\eta : R^P \rightarrow R$, can be a linear or a nonlinear function. Kernel functions, $k_m : R^{D_m} \times R^{D_m} \times \rightarrow R_{m=1}^P$, take P feature representations (not necessarily different) of data instances: $x_i = \{x_i^m\}_{m=1}^P$ where $x_i^m \in R^{D_m}$, and D_m is the dimensionality of the corresponding feature representation. η parameterizes the combination function and the more common implementation is:

$$k_\eta(x_i, x_j) = f_\eta(k_m(x_i^m, x_j^m)_{m=1}^P|\eta)$$

where the parameters are used to combine a set of predefined kernels (i.e., we know the kernel functions and corresponding kernel parameters before training). It is also possible to view this as:

$$k_\eta(x_i, x_j) = f_\eta(k_m(x_i^m, x_j^m|\eta)_{m=1}^P)$$

where the parameters integrated into the kernel functions are optimized during training. Most of the existing MKL algorithms fall into the first category and try to combine predefined kernels in an optimal way. The reasoning is similar to combining different classifiers: Instead of choosing a single kernel function it is better to have a set and let an algorithm do the picking or combination. There can be two uses of MKL:

- Different kernels correspond to different notions of similarity and instead of trying to find which works best, a learning method does the picking for us, or may use a combination of them. Using a specific kernel may be a source of bias, and in allowing a learner to choose among a set of kernels, a better solution can be found.

- Different kernels may be using inputs coming from different representations possibly from different sources or modalities. Since these are different representations, they have different measures of similarity corresponding to different kernels.

There are different ways in which the combination can be done and each has its own combination parameter characteristics. The functional forms of the existing MKL algorithms into three basic categories:

- Linear combination methods are the most popular and have two basic categories: unweighted sum (i.e., using sum or mean of the kernels as the combined kernel) and weighted sum. In the weighted sum case, we can linearly parameterize the combination function:

$$k_\eta(x_i, x_j) = f_\eta(\{k_m(x_i^m, x_j^m)\}_{m=1}^P|\eta) = \sum_{m=1}^{P} \eta_m k_m(x_i^m, x_j^m)$$

  where η denotes the kernel weights.

- Nonlinear combination methods use nonlinear functions of kernels, namely, multiplication, power, and exponentiation.

- Data-dependent combination methods assign specific kernel weights for each data instance. By doing this, they can identify local distributions in the data and learn proper kernel combination rules for each region[25].

2.4 Summary of article's results

In the paper the model evaluation is compared with other model that uses different technique, in particular:

- Use only t-test for feature extraction

- Use t-test plus RFE-LK

- Use t-test plus RFE-RBF

And also two different way of feature selection by merging the feature vector of the five threshold matrices are used. It is shown that the proposed method has the best performance compared to others:

- Accuracy 91.9%
- BAC 94%
- AUC 0.94
- Sensitivity 100%

Sensitivity 100% means that it can successfully classified all the MCI patients. Different tests are conducted by using only one threshold instead of all five together and the result is that using all five tresholded connectivity matrix is better. In figure 2.26 the ROC curves of the investigated method with different treshold. By selecting

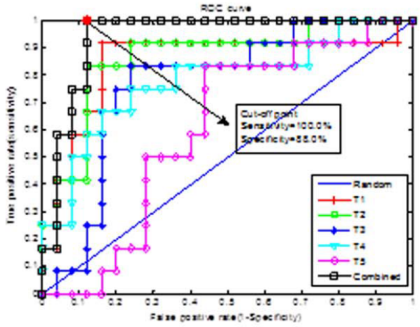


Figure 2.26: ROC curves of the investigated method (RFE-GK) by using different threshold

the best discriminative features (the ones with the highest occurrence frequency in all LOO cross-validation) 19 ROIs are obtained. Two average connectivity network based on all 19 ROIs is built. The vertexes are the ROIs and connections are the average of weights of corresponding edge in connectivity networks for subjects in the same group. So confronting the average connectivity matrix of MCI and a healthy person a significant difference can be seen in connections (figure 2.27). Other sec-

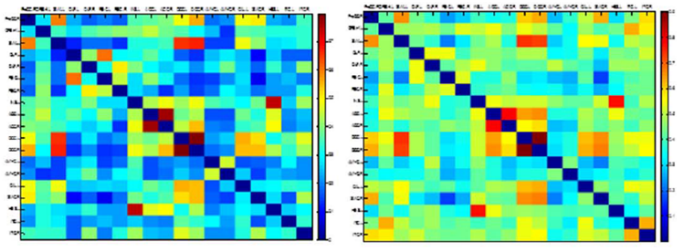


Figure 2.27: comparison between average connectivity matrix of a healthy person (on the left) and an MCI person (on the right). The colors express how strong is a connection (ascending order from red to blue).

ondary tests are conducted on the effect of threshold, preprocessing step and effect of feature extraction and selection, and each test suggest the fundamental importance of use each of that passage. It's possible to conclude that this method is a

valid proposal for detecting automatically the MCI patients with a high accuracy. The main limitations of the paper are:

- The definition of nodes and edges changes sensibly the topological property of connectivity networks
- Not analyzed the impact of different brain parcellation atlases on classifier performance
- The framework a be affected by unbalanced data (gap between healthy and MCI people)
- Not considered the distinction from AD to NC but only MCI and NC
- Limited sample size.

Chapter 3

Case study 2: clustering of high resolution sentinel-2 images

3.1 Problem definition

Land management and land planning requires a knowledge of the current state of the landscape. Understanding current land cover and how it is being used, along with an accurate means of monitoring change over time, is vital to any person responsible for land management. Measuring current conditions and how they are changing can be easily achieved through land cover mapping, a process that quantifies current land resources into a series of thematic categories, such as forest, water, and paved surfaces.
Changes in land use and land cover are pervasive, rapid, and can have significant impacts for people, the economy, and the environment. Among the organizations that will benefit from the information derived from land cover solutions are:

- governments and agribusiness who use them to assess demand, anticipate prices and plan the use of resources.
- Environment and research organizations
- Water districts
- Engineering firms
- Private forestry organizations
- Big traders and companies, providing services for farmers (For example, agricultural insurance companies, providers of fertilizers need to know what crops are grown in their region of interest and what are the areas of these crops.)
- Farmers, which do have a considerable interest, in knowing about problems in crops, and developments of the vegetation index concept could provide valuable information in stress management, for example in assessing irrigation demand, disease, pest and weed control, and crop nutrition.

This information must be delivered or made accessible in sufficient time for the user to make professional sense and use these advisories appropriately in the management process.

This topic has received great interest in recent years thanks to the fast scientific developments achieved by the two areas most correlated with land cover/land use mapping:

- Machine learning
- Remote sensing

For the first point the knowledge of new learning algorithms from the family of neural network, such as convolutional neural network or Unet, and the increasing efficiency of already well know algorithms such as Random forest and multilayer perceptron, brings great improvement in the managing and classification of big data. New information in faster time are now reachable, the training phase has improved in time and precision as well as their predictive ability. So nowadays became possible the manipulation of data that belong to very big crop areas.

Remote sensing

For what concern the second point the term "remote sensing" generally refers to the use of satellite or aircraft-based sensor technologies to detect and classify objects on Earth, including on the surface and in the atmosphere and oceans, based on propagated signals (e.g. electromagnetic radiation). It may be split into "active" remote sensing (i.e., when a signal is emitted by a satellite or aircraft and its reflection by the object is detected by the sensor) and "passive" remote sensing (i.e., when the reflection of sunlight is detected by the sensor) The most important

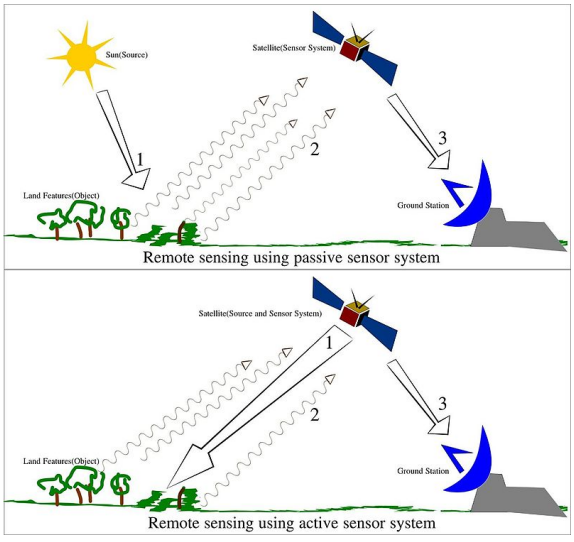


Figure 3.1: Schematizing of active and passive satellite remote sensing

applications of remote sensing in agrometeorological services are:

- Agriculture (crops): One basic information that remote sensing can provide to agriculture is data related to crop identification and area measurement under different types of crops, or acreage estimation. This enables to somewhat

estimate the total production by understanding the yield per unit area. Such information has far-reaching consequences in providing adequate food security.

- Forestry and vegetation mapping: Remote sensing can aid in providing a) information about the extent of forest cover and give a general idea of the types of forest cover; b) forest canopy density condition ; c) detection of forest hazards like fire, disease and excessive felling.

- Water resources : Understanding water resources is important from agricultural point of view. Water supply to agriculturally related sectors depends upon the available resources, both in terms of quantity and quality. Remote sensing data is useful in assessing water resources, irrigated area studies and its monitoring and determining potential ground water zones.

Remote sensing can also provide data related to ocean and coastal zones like identifying potential area of fish concentration, environmental degradation that takes place in coastal zones due to over exploitation, etc. Other promising areas of applications include, disaster assessment, drought monitoring, environmental monitoring, etc. all of which have advanced significantly[14]
For the thesis porpoise the focus is on satellite device and the most useful facilities connected to them:

- Hyperspectral cameras: Hyperspectral imaging produces an image where each pixel has full spectral information with imaging narrow spectral bands over a contiguous spectral range, so more features to give to learning algorithm

- The availability of large amount of "free and open" data images. Thanks to, becomes feasible to develop a lot of cloud services for fast access to satellite data and their products at high and medium spatial resolution scale.

- The availability of data images for different time

Figure 3.2: Logo of Copernicus the new name for the Global Monitoring for Environment and Security program, previously known as GMES, "It will provide accurate, timely and easily accessible information to improve the management of the environment, understand and mitigate the effects of climate change and ensure civil security"[4]

3.2 Past related works

In past literature numerous attempt were done in ordered to make good classification of land crop. The two mainly approach followed are:

- Pixel-based
- Object-based

In the first case the samples used are the single pixels characterized by the hyperspectral feature vector connected to it.
In the second case the samples for the learning algorithm are not the single pixel but a group of them. So in order to make a good object-based classification the step of segmentation or parcellation that came before the real classification step is fundamental.
Numerous attempt in comparison of the two categories were done[2][32][27][24]. Actually is not possible to state which one is better because it is extremely dependent on the dataset used and on the remote sensing product used (Landsat, Sentinel, Modis, Sar...). But in general what emerges from the literature is that if a good object segmentation is done, the object-oriented classification outperforms a little the pixel-based one. In particular the final mapping derived from object oriented is more homogeneous compared to pixel-based that suffer from salt and pepper effect and so in general it needs a postprocessing stage. On the other hand the pixel-based framework are easier to implement and due to the increasing information resulting from new hyperspectral camera it reaches very good performance. For example in [33] due to the unavailability of good information about field boundaries a pixel-based approach with features extracted from combination of Landsat and Sar image was implemented and additional back-scattering intensity feature is added.

For what concerns the machine learning approach almost all the literature uses a supervised approach because the ground truth information obtained by ground survey gives a big improvement in results compared to unsupervised approach that often fails in the detection of common features for each class. For what concerns the specific supervised learning algorithm numerous comparison where made and recently major importance have given to Neural Network based framework instead of more classical Random Forest. For example in [27] MLP, 1-d CNN and 2-d CNN algorithm trained with Landsat-8 and Sentinel-1 image have outperformed RF. That still remains one of the most used due to its simplicity, good knowledge and it is already implemented in various programming language. In addition it embeds the ability to extract the most important features. For example in [30] a Random Forest approach plus multitemporal information where used to select best features extracted from Modis time series, and the relevance of the NDVI band is underlined.

Another common point in literature is the increasing in accuracy by using multitemporal information. Each crop is characterized by is own phenology (represented by its spectral signature) and this information can be exploited and carried out only by augment the feature vector with information of different period of the year. In all the cited works the multitemporal approach is exploited except in [2] where only multispectral information of the same time were used.

The topic is very big and dynamic and the technologies involved are improving day by day and new technique and approaches are discovered. For example a very recent paper of Belgiu and Csillik [24] compare Random Forest with a new approach called TWDTW that was firstly used for speech recognition and now is used to recognize the spectral signature of crops. It still needs improvement since it outperforms RF only in the dataset taken from Romania and Italy and is outperformed

in the American dataset.

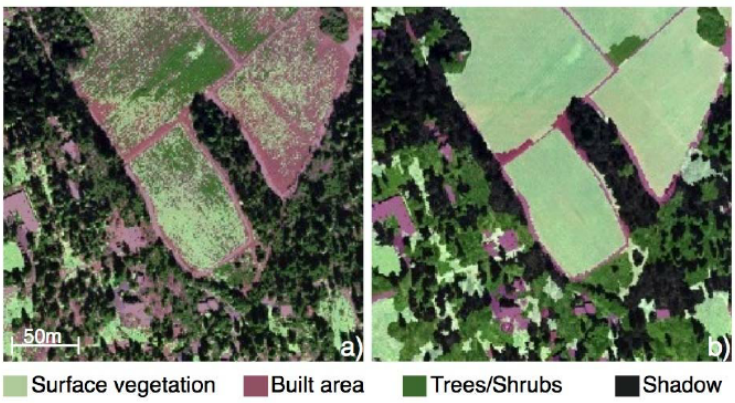


Figure 3.3: Salt and pepper effect

3.3 Proposed method

The proposed method is a pixel-oriented crop classification. The general scheme is depicted in figure 3.4.
The main aim is to reach the best accuracy using as less features as possible. Since a common wrapper approach for features selection is not suitable, due to the large amount of data and high computational cost, a different approach is exploited.
The basic features (Red,Green,Blue,Nir band) for each time period are extracted from the downloaded satellite images, the NDVI is then calculated. Since the classification accuracy is strongly dependent from the time phenology of crops different time information are evaluated. Specifically due to the impossibility of evaluate the a-priori importance of each time contributions, three scenarios are implemented:

- **scenario 1**: all monotemporal images of dataset (composed by R,G,B,Nir,NDVI features) are individually evaluated and the best monotemporal image is found.
- **scenario 2**: a temporal window made of composition of 2 monotemporal image (a multitemporal image of 2) is considered. All combinations of 2 monotemporal images are evaluated and the best multitemporal image is found.
- **scenario 3**: a temporal window made of composition of 3 monotemporal image (a multitemporal image of 3) is considered. All combinations of 3 monotemporal images are evaluated and the best multitemporal image is found.

From the preceding steps the best temporal images are extracted and the increasing performance due to the increase of number of features is analyzed.
In order to exploit also the importance of non-temporal features, all combination of Pca and textural features are combined with the best monotemporal image and the performances evaluated.
The learning alghoritm used is a Multilayer Perceptron preceded by balancing (only for Dataset A) and normalization pre-processing steps. The final classified image is then refined by using a median filter in order to remove the salt-and-pepper effect.

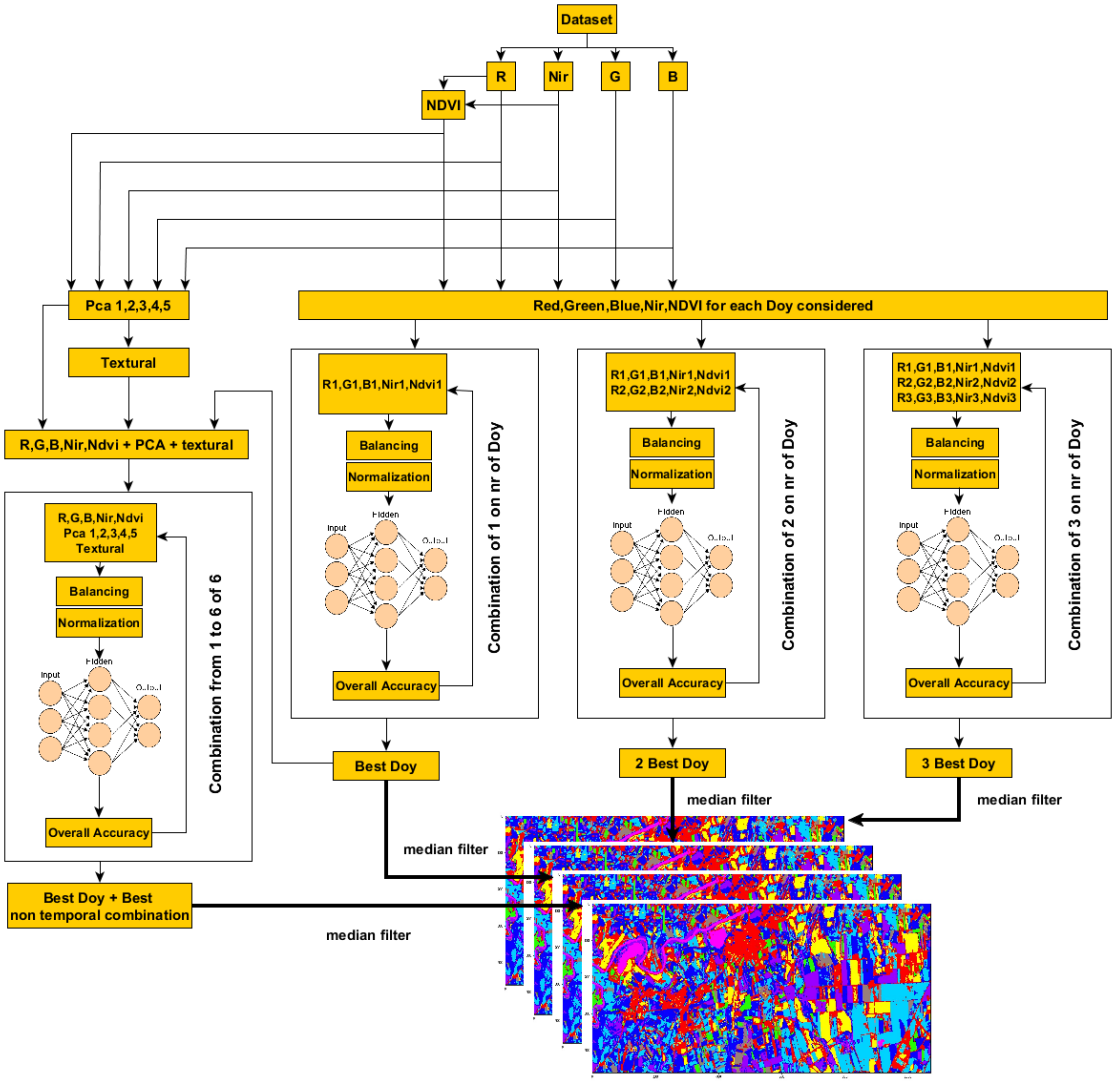


Figure 3.4: Scheme of the proposed method

3.3.1 Data acquisition

The data used for the thesis were collected from the Copernicus website [4] where is possible to download the images taken by the two Sentinel satellite. The only input needed are the localization of the area and a time range in which the images were taken. The proposed method used only the data from Sentinel-2, so some characteristics of this satellite are exploited.

Sentinel-2

Sentinel-2 mission consists of twin polar-orbiting satellite launched by European Space Agency (ESA) in 2015 and are used in various application areas such as land cover change detection, natural disaster monitoring, forest monitoring and most importantly in agricultural monitoring and management[5]. It is equipped with multi-spectral optical sensors which captures 13 bands of different wavelengths shown in table-1. It has also high revisit time (10days at the equator and 5 days with twin satellites (Sentinel-2A, Sentinel-2B). It has gain more importance due to fact that

it possesses various key features such as, free data products available at reasonable spatial resolution (which is 10m for Red, Green, Blue and Near Infrared bands), high revisit time and has very good spectral resolution among other available free data sources.

Sentinel-2 Bands	Central Wavelength (µm)	Resolution (m)
Band 1 (B1) – Coastal	0.443	60
Band 2 (B2)– Blue	0.490	10
Band 3 (B3)– Green	0.560	10
Band 4 (B4)– Red	0.665	10
Band 5 (B5)– Vegetation Red Edge	0.705	20
Band 6 (B6)– Vegetation Red Edge	0.740	20
Band 7 (B7)– Vegetation Red Edge	0.783	20
Band 8 (B8)– NIR	0.842	10
Band 8A (B8A)– Narrow NIR	0.865	20
Band 9 (B9)– Water vapor	0.945	60
Band 10 (B10)– SWIR	1.375	60
Band 11 (B11)– SWIR	1.610	20
Band 12 (B12)– SWIR	2.190	20

Figure 3.5: Sentinel-2 bands detail

Postprocessing

Once the raw images are taken two are the next step:

- Resampling: in order to make a subset of the total image and since the spatial resolution of all bands are different resampling uniform the pixel size for each band.
- Make subset: take form a bigger area just the area of interest

The two procedures were made by using SNAP tool, A common architecture for all Sentinel Toolboxes developed by Brockmann Consult, Array Systems Computing and C-S called[6].

3.3.1.1 Ground truth dataset acquisition

By following the preceding steps two areas were selected:

- Dataset A: Racconigi, Piedmont region, situated in north part of Italy with central coordinates 44°48'26"N, 7°37'37"E. Almost 46 Km^2 area covered.
- Dataset B: central coordinates near Carpi, Emilia-Romagna region, situated in center-north part of Italy with central coordinates 44°47'01"N, 10°59'37"E.Amost 2640 Km^2 area covered.

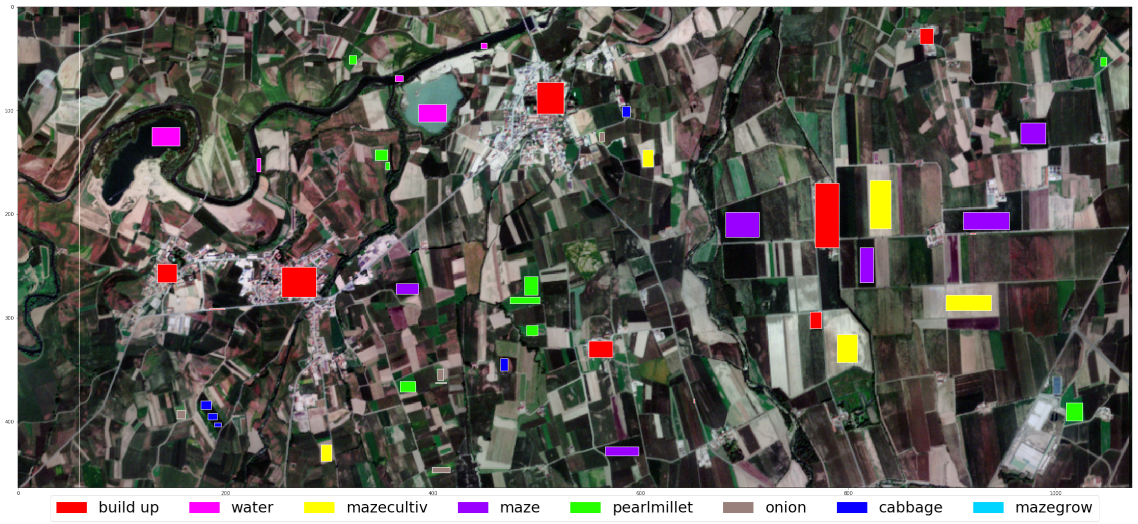


Figure 3.6: Dataset A

The next step is to collect the ground truth information necessaries for the supervising training algorithm.
For the Dataset A a ground survey was conducted by going physically in loco, observing the crops and saving thanks to GPS mobile device the geolocalization of the area investigated. For the Dataset B a physical ground survey was impossible. So information about ground truth were taken from LUCAS archive and than visualized in QGIS, an Open-source software used for visualization, editing, analysis of geographical data. In detail the LUCAS dataset just provided information about the point location, the selection of pixel were made manually by overlapping images and LUCAS data.
LUCAS (Land Use/Cover Area frame statistical Survey) is the name given to a series of ground survey decided by European Parliament and the European Statistical Office (EUROSTAT) with the aim to gather information on land cover and land use. In figure 3.6 and 3.7 the details of area selected and their ground truth data with the selected classes.

3.3.2 Feature extraction

The feature extraction step mainly consisted in selection an manipulation of the bands downloaded from Copernicus website.
This step was exploited using the already cited SNAP tool software for analysis and processing of Sentinel images provided by ESA.
Synthesizing its charachteristics:

- Common architecture for all Toolboxes
- Very fast image display and navigation even of giga-pixel images
- Graph Processing Framework (GPF): for creating user-defined processing chains
- Advanced layer management allows adding and manipulation of new overlays such as images of other bands, images from WMS servers or ESRI shapefiles
- Rich region-of-interest definitions for statistics and various plots

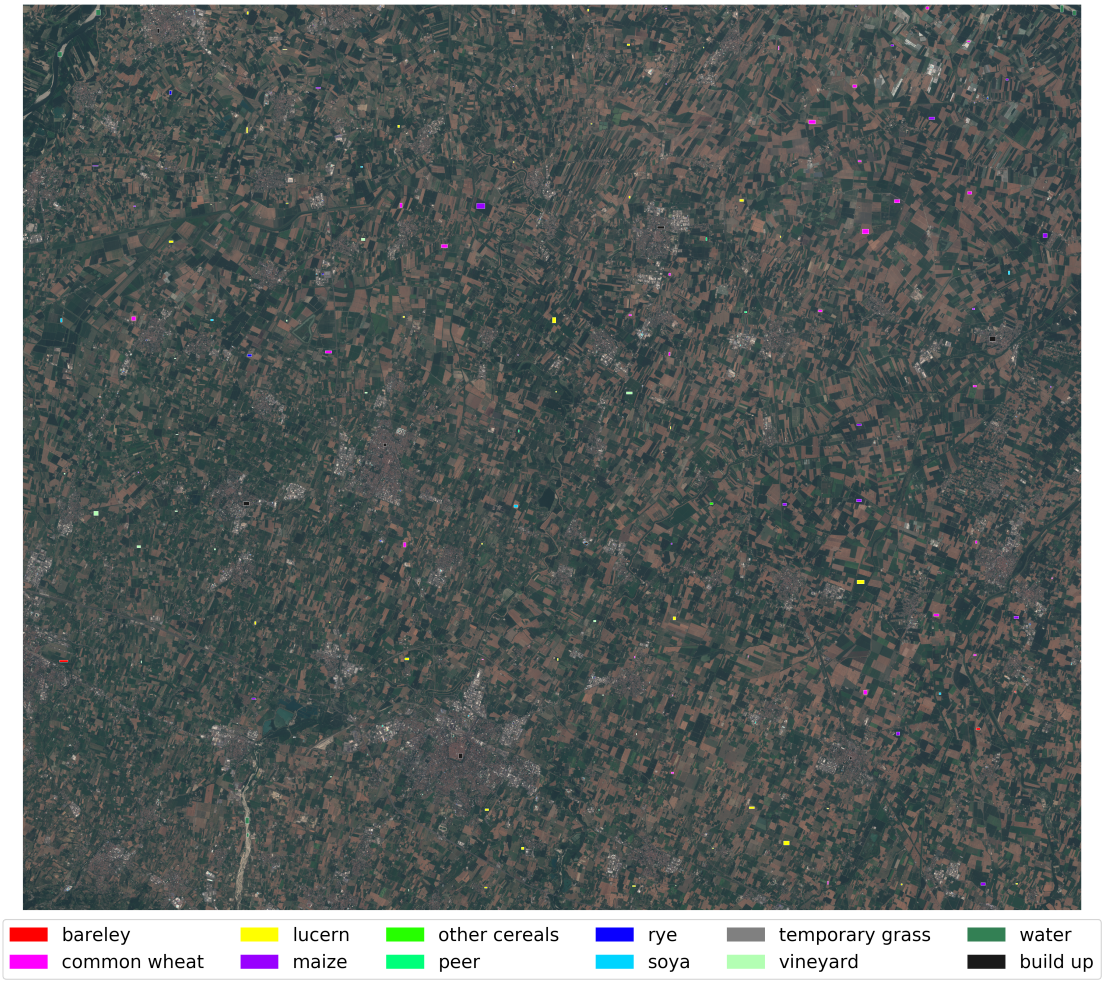


Figure 3.7: Dataset B

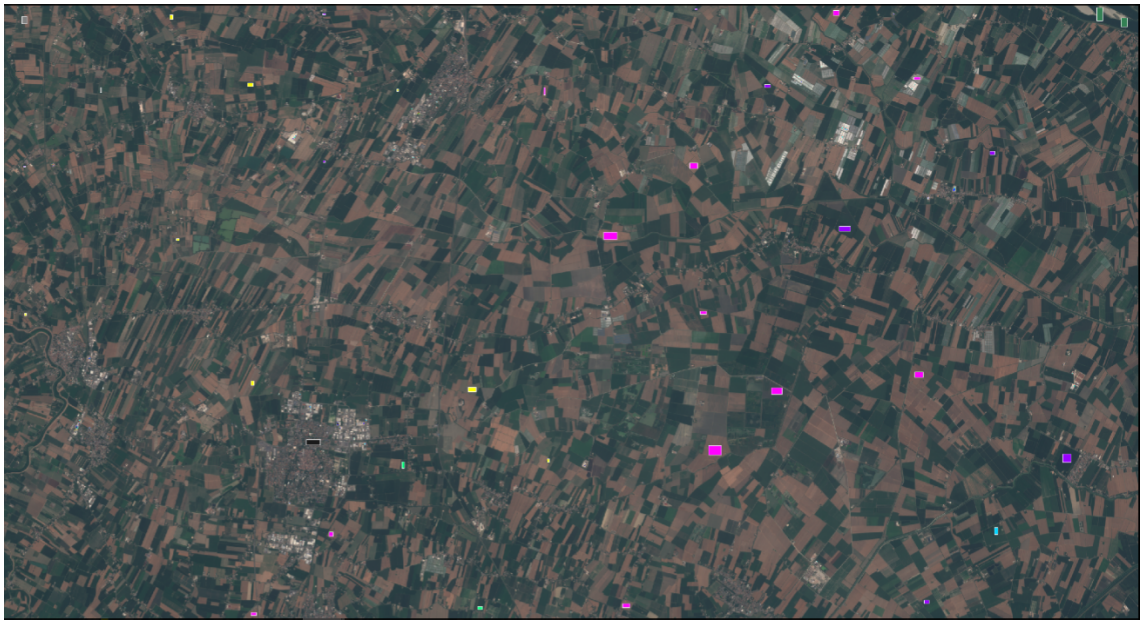


Figure 3.8: Zoom-in of the upper-right corner of dataset B

Class	Pixels	Percentage
build up	4269	26.03%
water	1075	6.56%
maze cultivated	2460	15%
maze	2946	17.97%
pearl millet	1192	7.27%
onion	271	1.65%
cabbage	326	1.99%
maze grown	3858	23.52%
Total	16397	100%

(a) Dataset A

Class	Pixels	Percentage
barley	739	2.44%
common wheat	7877	26%
lucern	5565	18.38%
maize	5198	17.16%
other cereals	202	0.67%
peer	326	1.07%
rye	499	1.66%
soya	931	3.07%
temporary grass	549	1.81%
vineyard	2204	7.28%
water	2936	9.69%
build up	3263	10.77%
Total	30289	100%

(b) Dataset B

Table 3.1: Composition of two dataset: dataset A on the left and dataset B on the right

- Easy bitmask definition and overlay
- Flexible band arithmetic using arbitrary mathematical expressions
- Accurate reprojection and ortho-rectification to common map projections,
- Geo-coding and rectification using ground control points
- Automatic SRTM DEM download and tile selection
- Product library for scanning and cataloguing large archives efficiently
- Multithreading and Multi-core processor support
- Integrated WorldWind visualisation[6]

3.3.2.1 Basic

For this thesis work will be considered basic features the 4 10m high resolution bands of Sentinel 2:

- B4 = the red band
- B3 = the green band
- B2 = the blue band
- NIR = the near infrared band

The first three bands essentially are used to build up the Natural RGB color image and so they have the ability to capture the Color of the fields. The near infrared is good for mapping shorelines and biomass content, as well as at detecting and analyzing vegetation.

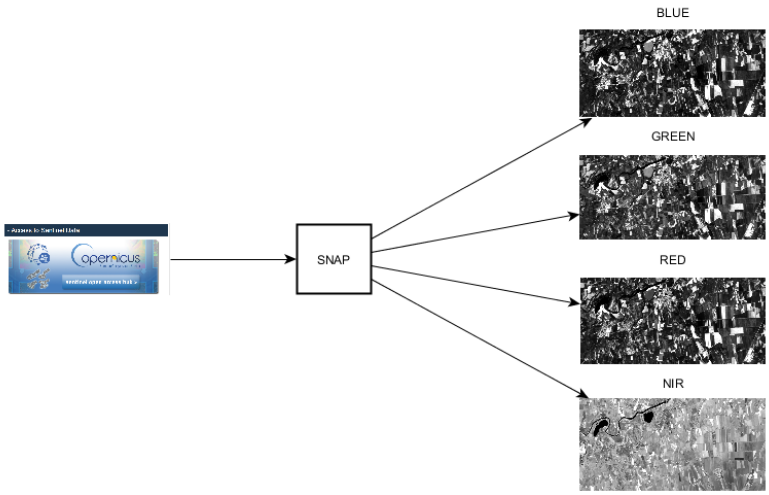


Figure 3.9: Basic feature extraction

3.3.2.2 NDVI

The normalized difference vegetation index (NDVI) is a simple graphical indicator that can be used to analyze remote sensing measurements and assess whether the target being observed contains live green vegetation or not.
NDVI is calculated on a per-pixel basis as the normalized difference between the red and near infrared bands from an image:

$$NDVI = \frac{\rho(B8) - \rho(B4)}{\rho(B8) + \rho(B4)}$$

where $\rho(B8)$ is the near infrared band value for a cell and $\rho(B4)$ is the red band value for the cell. NDVI can be calculated for any image that has a red and a near infrared band. The biophysical interpretation of NDVI is the fraction of absorbed photosynthetically active radiation, or simpler it measures the "greenness" of vegetation.

Many factors affect NDVI values like plant photosynthetic activity, total plant cover, biomass, plant and soil moisture, and plant stress. Because of this, NDVI is correlated with many ecosystem attributes that are of interest to researchers and managers (e.g., net primary productivity, canopy cover, bare ground cover). Also, because it is a ratio of two bands, NDVI helps compensate for differences both in illumination within an image due to slope and aspect, and differences between images due things like time of day or season when the images were acquired. Thus, vegetation indices like NDVI make it possible to compare images over time to look for ecologically significant changes.

The output of NDVI is a new image file/layer. Values of NDVI can range from -1.0 to +1.0. Higher values signify a larger difference between the red and near infrared radiation recorded by the sensor - a condition associated with highly photosynthetically-active vegetation. Low NDVI values mean there is little difference between the red and NIR signals. This happens when there is little photosynthetic activity, or when there is just very little NIR light reflectance (i.e., water reflects very little NIR light).

A negative value of NDVI actually means presence of water so is a very important feature for water resources detection.

Great importance has the NDVI plot over time. Where on the y axis there are some average NDVI values for some pixels that belong to the same class and on the x axis the multitemporal information. By looking at this spectral signature is possible to detect the phenological life-cycle of each crop.

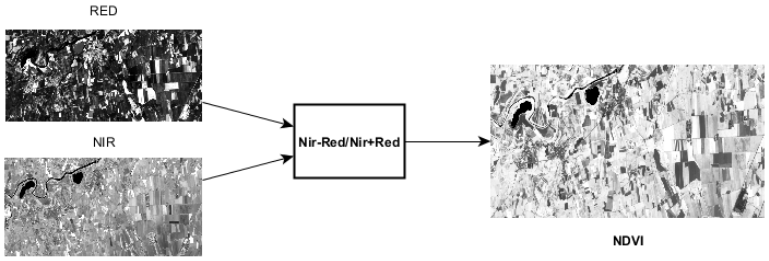


Figure 3.10: Ndvi feature extraction

3.3.2.3 Pca

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The PCA is mostly used as a dimensionality reduction technique in order to extract the most important features of a feature vector. In crop classification is mainly used in order to generate new features able to encapsulate important information as showed in [29]. The key point of the transformation is to reproject the feature vector associated to each observation into a new coordinate system which axis are the so called principal analysis and they are ordered in function of their importance that is evaluated as the variability of data. Imagine to be in the situation of figure 3.12 all the points are represented by 2 features. The major variability of the data is on the blue line and the second major variability is on the purple line. Mathematically speaking the two directions are represented by the eigenvectors associated to the eigenvalues of the covariance matrix of the points (the observations). The highest eigenvalue is connected to the eigenvector with the highest variability of data. Given the eigenvectors and the original points it's then possible to obtain the final transformation as:

$$\text{Final Data} = \text{RowfeatureVector} \times \text{RowdataAdjust}$$

where

- RowfeatureVector is the matrix with the eigenvectors in the columns transposed so that the eigenvectors are now in the rows, with the most significant eigenvector at the top

- RowdataAdjust is the mean-adjusted data transposed, ie. the data items are in each column, with each row holding a separate dimension.

The mean-adjust come out from the correlation matrix.
To reduce the final data we have to discard some eigenvectors from Rowfeaturevector. If a eigenvector associated with a low eigenvalue is removed the information loss is little otherwise is bigger. So in the end we have transformed a vector of features of dimension D to another that embed the major information of the original one but of dimension d, with $d < D$.

In image field the generation of Pca means build up the images with transformed value associated to each pixel. The number of images generated are the same of the features of the original image (one for each principal component). The Pca extraction is made by using SNAP tool that automatically embed this option. Specifically SNAP tool follow the procedure:

1. Average the pixels across the input images to compute a mean image.

2. Subtract the mean value of each input image (or image from step 1) from itself to produce zero-mean images.

3. Compute covariance matrix from the zero-mean images 3. given in step 2.

4. Perform eigenvalue decomposition of the covariance matrix.

5. Compute PCA images by multiplying the eigenvector matrix by the zero-mean images given in step2. The number of produced images are the same of the Principal Components.

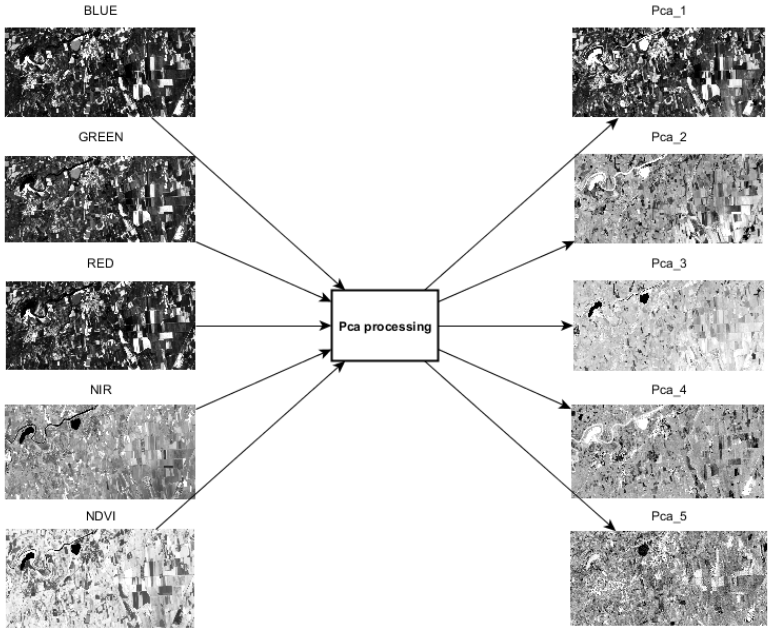


Figure 3.11: pca features extraction

In this thesis for both dataset the Pca is calculated on the best individual temporal image considering the stack of Red,Green,Blue,Nir and NDVI.

3.3.2.4 Texture features

Spatial information in the form of texture features can be useful for image classification. Texture measures can produce new images by making use of spatial information inherent in the image. Texture is the pattern of intensity variations in an image and can be a valuable tool in improving land-cover classification accuracy. Texture information involves the information from neighbouring pixels which is important to characterize the identified objects or regions of interest in an image.
The Gray Level Co-occurrence Matrix (GLCM) is one of the most widely used methods to compute second order texture measures. Several texture features can be computed from the GLCM matrix, e.g., angular second moment, contrast, correlation, entropy, variance, inverse difference moment, difference average, difference variance, difference entropy, sum average, sum variance and sum entropy[31]. Each feature models different properties of the statistical relation of pixels co-occurrence estimated within a given moving window and along predefined directions and inter-pixel distances. Also in this case the resulting image is extracted by using SNAP tool, which automatically execute the following steps:

1. Quantize the image data. Each sample on the original image is treated as a single image pixel and the value of the sample is the intensity of that pixel. These intensities are then further quantized into a specified number of discrete gray levels.

2. Create the GLCM. It will be a square matrix N x N in size where N is the Number of levels specified under Quantization. The matrix is created as follows:

   (a) Let s be the sample under consideration for the calculation.

   (b) Let W be the set of samples surrounding sample s which fall within a window centered upon sample s of the size specified under Window Size.

   (c) Considering only the samples in the set W, define each element i,j of the GLCM as the number of times two samples of intensities i and j occur in specified Spatial relationship (where i and j are intensities between 0 and Number of levels-1)
   . The sum of all the elements i, j of the GLCM will be the total number of times the specified spatial relationship occurs in W.

   (d) Normalize the GLCM:

   - Divide each element by the sum of all elements.
     The elements of the GLCM may now be considered probabilities of finding the relationship i, j (or j, i) in W.

3. Calculate the selected Feature. This calculation uses only the values in the GLCM. (i.e. homogenity, entropy, variance, correlation)
   Then snap tool give the values of the calculated feature to all pixel belonging to the chosen original window.

So the parameter to be selected by the user are:

- Window dimension

- Directions taken in consideraton for building the co occurence matrix
- Quantization of gray level
- The displacement of the window

$$\begin{matrix} 0 & 0 & 0 & 1 & 2 \\ 1 & 1 & 0 & 1 & 1 \\ 2 & 2 & 1 & 0 & 0 \\ 1 & 1 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{matrix} \qquad C = \frac{1}{16} \begin{bmatrix} 4 & 2 & 1 \\ 2 & 3 & 2 \\ 0 & 2 & 0 \end{bmatrix}$$

Figure 3.12: Example of gray-scale image with 3 tones on the left and correspective GLCM with operational position 1-dx and 1-down

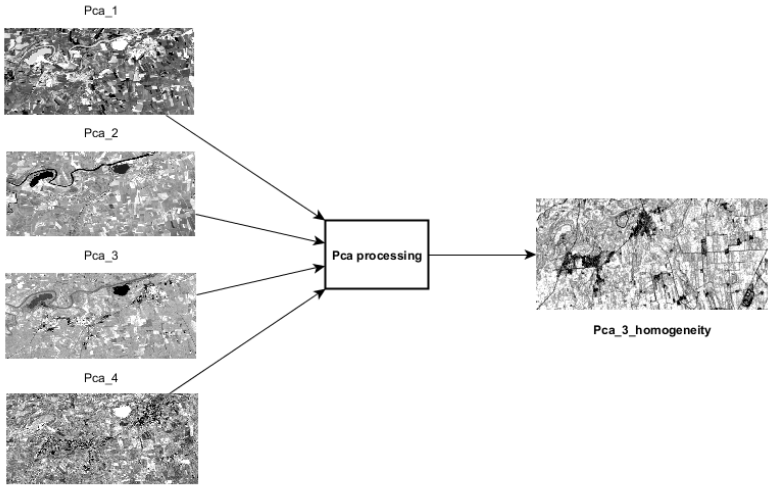


Figure 3.13: texture feature extraction

In this thesis the parameters used for building the textural feature are:

- **Dataset A**: Window dimension=5, Directions=ALL, Quantization=32, Window displacement=1
- **Dataset B**: Window dimension=7, Directions=ALL, Quantization=16, Window displacement=1

3.3.2.5 Multitemporal features

Each crop is characterized by its own phenology cycle. This information is for its nature a temporal data, since the field change its property time-by-time.
Using the basic features with monotemporal information essentially means make a classification based on color plus infrared of crop. Additional information is applied by using NDVI, but in general the information given to the classifier are not so

expressive of the property of crops.
In particular often happens that two different crops show a very similar spectral signature if no-time information is added, this means that they are hard to classify. In figure 3.14 is shown three different crops in three different period of the year. It's possible to see from visual analysis and by looking at the three different spectral signature in figure 3.15 that in 08/05 the three crops are very similar (only the nir component show a significant difference), so to increase accuracy is necessary to add the spectral information of other period.
In this thesis the feature that are added in time are the basic ones (R,G,B,NIR) and the NDVI. So adding the temporal information means adding 5 feature for each temporal images used. In table 3.2 the detail of temporal data. In figure 3.16 the plot of multitemporal NDVI values for each crop of dataset A. The plot shows the different phenological phase of each crop and the different discriminatory power of the NDVI feature in function of the different time period considered.

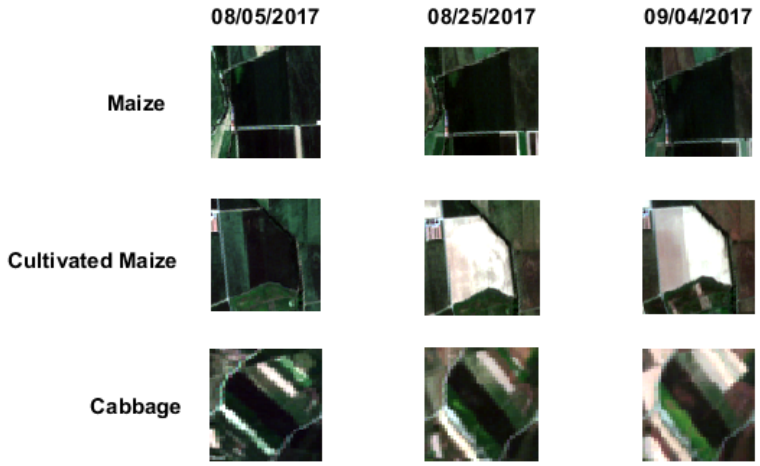


Figure 3.14: Three crops: maize,cultivated maize and cabbage in different day of the year

3.3.3 Processing raw features

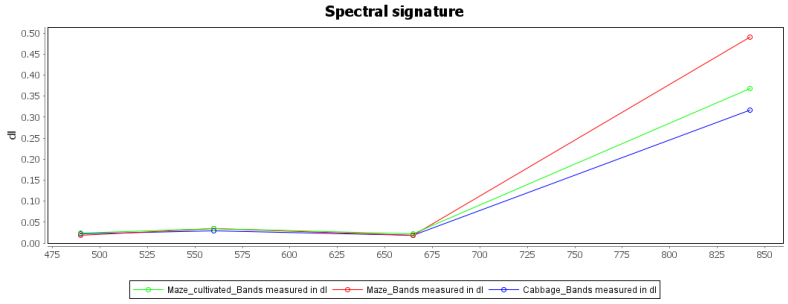
3.3.3.1 Normalization

Since the bands information values have different scales dependently from the band itself a normalization preprocessing step was made. The preprocessing normalization applied, has rescaled all the values between 0 and 1. The used formula is:

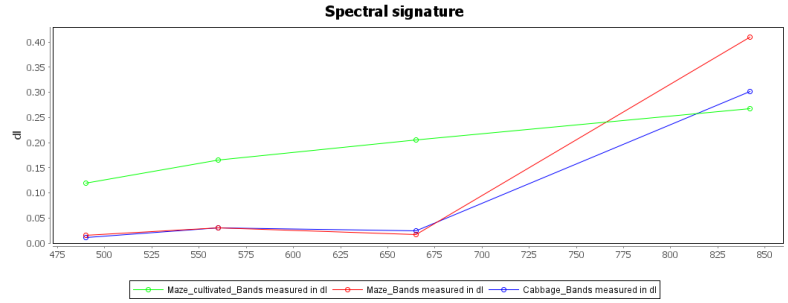
$$x' = \frac{x_i - min(x)}{max(x) - min(x)}$$

3.3.3.2 Balancing

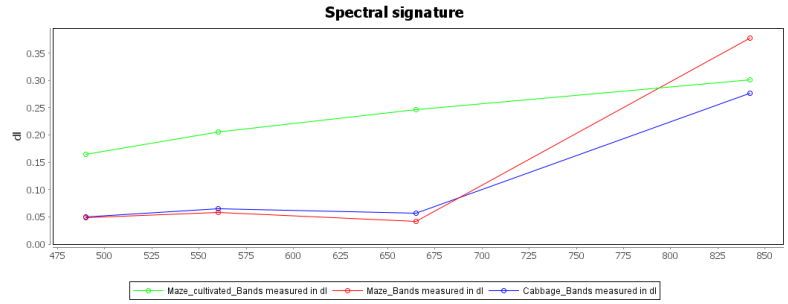
Imbalanced data refers to a situation where the number of observations is not the same for all the classes in a classification dataset. Machine learning classifiers can fail to cope with imbalanced training datasets as they are sensitive to the proportions of the different classes. As a consequence, these algorithms tend to favor the class with



(a) 08/05/2017



(b) 08/25/2017



(c) 09/04/2017

Figure 3.15: Spectral signature of crops in figure 3.14 considering basic bands

Date	Doy
09/17/17	260
09/04/17	247
08/25/17	237
08/05/17	217
07/26/17	207
07/06/17	187
06/19/17	170
05/30/17	150
05/27/17	147
05/17/17	137
04/20/17	110

(a) Dataset A

Date	Doy
07/04/15	185
08/03/15	215
09/02/15	245
09/12/15	255
10/22/15	295
02/19/16	50
03/20/16	80
04/29/16	120
06/18/16	170
07/18/16	200

(b) Dataset B

Table 3.2: Temporal data for dataset A (on the left) and dataset B (on the right)

the largest proportion of observations (known as majority class), which may lead to misleading accuracies. This may be particularly problematic when the interest

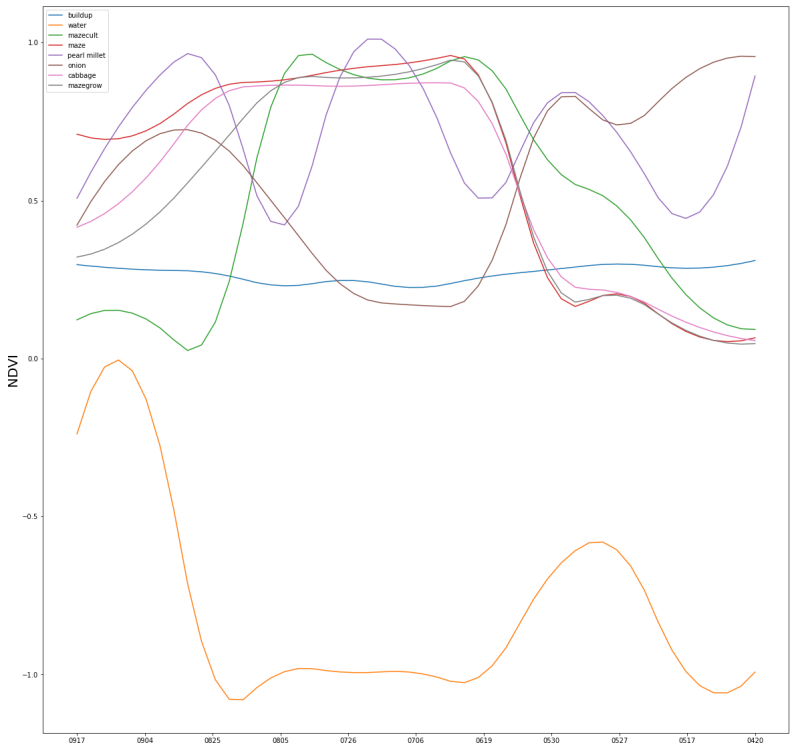


Figure 3.16: multitemporal NDVI plot for each crop of dataset A

is in the correct classification of a "rare" class (also known as minority class) but high accuracies which are actually the product of the correct classification of the majority class are found(ie, are the reflection of the underlying class distribution). Given that these algorithms aim to minimize the overall error rate, instead of paying special attention to the minority class, they may fail to make an accurate prediction for this class if they don't get the necessary amount of information about it.
In order to cope with imbalanced data some pixel from the majorities classes are removed. This is done only for Dataset A which is representative of a smaller area than the one of Dataset B, so a reduction of training samples don't change the performances of the classifier. For Dataset B this balancing step is avoided since the area to classify is very big and a removal of pixel in training dataset can reduce the performance of the classifier. In 3.3 the detail of dataset A pre and post balancing.

3.3.4 Feature selection

Looking at the general scheme in figure 3.4 it's possible to see that it leads naturally to choose best features. Actually it's possible to imagine the feature selection step as a forward wrapper method. The difference respect the classical forward feature selection is that not all of possible combinations of features are exploited; instead in the proposed method just some combinations of them are analyzed.
Using the classical forward selection step permits to analyze all the features space since the algorithm is trained, at the starting point, with a few subset of features and than each feature is added to the starting set and the result evaluate; this process is iterative until all the features are used.
Obviously this process is computationally very expensive especially when integrated

Class	Pixels	Percentage
build up	4269	26.03%
water	1075	6.56%
maze cultivated	2460	15%
maze	2946	17.97%
pearl millet	1192	7.27%
onion	271	1.65%
cabbage	326	1.99%
maze grown	3858	23.52%
Total	16397	100%

(a) unbalanced

Class	Pixels	Percentage
build up	539	16.50%
water	475	14.54%
maze cultivated	360	11.02%
maze	446	13.65%
pearl millet	392	12%
onion	271	8.30%
cabbage	326	9.98%
maze grown	458	14.01%
Total	3267	100%

(b) Balanced

Table 3.3: Composition of dataset A: unbalanced on the left and after balancing on the right

with cross validation for choosing best hyperparameters of algorithm. Implementing this step for this work is unfeasible.
The choice of subsets of features involved in the method is justified by a series of experiments where other combinations of other features where exploited but didn't lead to sensible increment in accuracy.

3.3.5 Classifier

The implemented classifier for the Task is a Multilayer Perceptron which belong to the class of feedforward artificial neural network.

A feedforward neural network is an artificial neural network wherein connections between the units do not form a cycle. As such, it is different from recurrent neural networks. The feedforward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

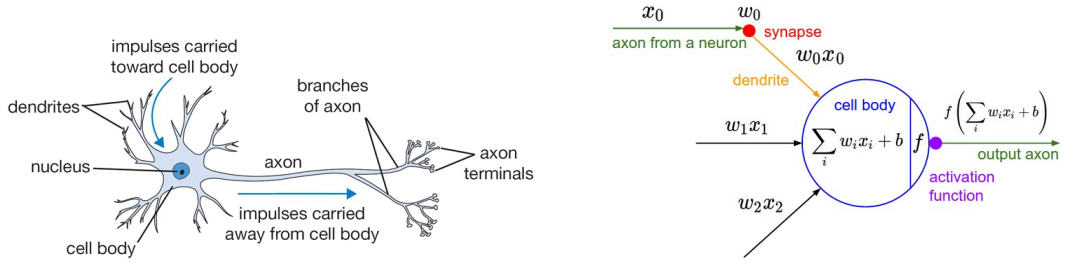
3.3.5.1 MLP

A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

Biological motivations and connections

The basic computational unit of the brain is a neuron. Approximately 86 billion neurons can be found in the human nervous system and they are connected with approximately 10^{14} - 10^{15} synapses. The figure 3.17 shows a cartoon drawing of a biological neuron (left) and a common mathematical model (right). Each neuron

receives input signals from its dendrites and produces output signals along its (single) axon. The axon eventually branches out and connects via synapses to dendrites of other neurons. In the computational model of a neuron, the signals that travel along the axons (e.g. x0) interact multiplicatively (e.g. w0x0) with the dendrites of the other neuron based on the synaptic strength at that synapse (e.g. w0). The idea is that the synaptic strengths (the weights w) are learnable and control the strength of influence (and its direction: excitory (positive weight) or inhibitory (negative weight)) of one neuron on another. In the basic model, the dendrites carry the signal to the cell body where they all get summed. If the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon. In the computational model, we assume that the precise timings of the spikes do not matter, and that only the frequency of the firing communicates information. Based on this rate code interpretation, we model the firing rate of the neuron with an activation function f, which represents the frequency of the spikes along the axon[12].



(a) neuron biological model

(b) neuron mathematical model

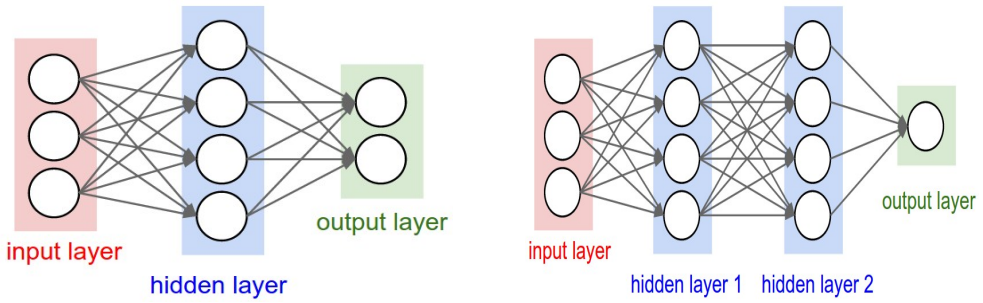
Figure 3.17: A cartoon drawing of a biological neuron (a) and its mathematical model (b).

Architecture

Neural Networks are modeled as collections of neurons that are connected in an acyclic graph. In other words, the outputs of some neurons can become inputs to other neurons. Cycles are not allowed since that would imply an infinite loop in the forward pass of a network. Instead of an amorphous blobs of connected neurons, Neural Network models are often organized into distinct layers of neurons. For regular neural networks, the most common layer type is the fully-connected layer in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connection[12]. In figure 3.18 are two example Neural Network topologies that use a stack of fully-connected layers:

Mathematical interpretations

Neural network models can be viewed as simple mathematical models defining a function $f : X \rightarrow Y$ or a distribution over X or both X and Y. Sometimes models are intimately associated with a particular learning rule. A common use of the phrase "ANN model" is really the definition of a class of such functions (where



(a) 1 hidden layer, 2 output neurons

(b) 2 hidden layer, 1 output neuron

Figure 3.18: (a): A 2-layer Neural Network (one hidden layer of 4 neurons (or units) and one output layer with 2 neurons), and three inputs. (b): A 3-layer neural network with three inputs, two hidden layers of 4 neurons each and one output layer. Notice that in both cases there are connections (synapses) between neurons across layers, but not within a layer.

members of the class are obtained by varying parameters, connection weights, or specifics of the architecture such as the number of neurons or their connectivity).

Mathematically, a neuron's network function $f(x)$ is defined as a composition of other functions $g_i(x)$, that can further be decomposed into other functions. This can be conveniently represented as a network structure, with arrows depicting the dependencies between functions. A widely used type of composition is the nonlinear weighted sum, where $f(x) = K(\sum_i w_i g_i(x))$, where K is some predefined activation function, such as the hyperbolic tangent or sigmoid function or softmax function or rectifier function. The important characteristic of the activation function is that it provides a smooth transition as input values change, i.e. a small change in input produces a small change in output. Figure 3.19 depicts such a decomposition of f, with dependencies between variables indicated by arrows.
One way to see the figure is the functional one: the input x is transformed into a 3-dimensional vector h, which is then transformed into a 2-dimensional vector g, which is finally transformed into f. This view is most commonly encountered in the context of optimization.

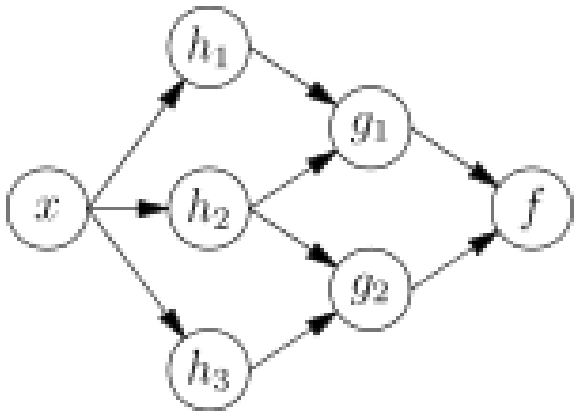


Figure 3.19: ANN dependency graph

Learning process

The parameters that the MLP has to learn are the weights that connect each neurons to all the others of next layer and the bias for each neuron. It's easy to understand that increasing the number of neurons or the number of hidden layer brings to a big increase in computational time need to learn the parameters that grows up in non linear way.

The learning process of a MLP is composed by the following steps:

1. Pick the network architecture(initialize with random weights)
2. Do a forward pass (Forward propagation)
3. Calculate the total error(we need to minimize this error)
4. Back propagate the error and Update weights(Back propagation)
5. Repeat the process(2-4)for no of epochs/until error is minimum.

So the training phase is an iterative process which consist of two different phases: Forward propagation and Backward propagation. One epoch is just one forward pass and one backward pass of all the training examples.

Forward propagation In the forward step the features (that represent the sample itself) of a sample is given to the net. This information is processed forwardly following the the direction from input layer to output layer and calculating the value of each node as:

$$a_j^i = \sigma(\sum_k (w_{jk}^i a_k^{i-1}) + b_j^i)$$

where:

- j is the node
- i is the layer
- σ is the activation function
- w_{jk}^i is the weight from the k^{th} neuron in the $(i-1)^{th}$ layer to the j^{th} neuron in the i^{th} layer
- b_{ij} is the bias of the j^{th} neuron in the i^{th} layer
- a_{ij} represents the activation value of the j^{th} neuron in the i^{th} layer.

Until the activation value of output nodes is reached.

Since this is training and labels of samples are known a cost function is defined, in literature different measures of loss are analyzed. What is important is that the cost function is a measure of how good is the output obtained after a forward propagation phase of all the samples in the used batch (subset of samples used for each forward-backward propagation).

Backward propagation So now the network is capable of calculating the total error for a given training set. The weights in the network are the only parameters that can be modified to make the error E as low as possible. Because E is calculated by the extended network exclusively through composition of the node functions, it is a continuous and differentiable function of the ‘ weights $w_1, w_2, ..., w_l$ in the network. We can thus minimize E by using an iterative process of gradient descent, for which we need to calculate the gradient

$$\Delta E = (\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, ..., \frac{\partial E}{\partial w_l})$$

Each weight is updated using the increment

$$\Delta w_i = -\lambda \frac{\partial E}{\partial w_i} \text{ for i} = 1, \ . \ . \ ., \text{l}$$

where λ represents a learning constant, i.e., a proportionality parameter which defines the step length of each iteration in the negative gradient direction. With this extension of the original network the whole learning problem now reduces to the question of calculating the gradient of a network function with respect to its weights. Once we have a method to compute this gradient, we can adjust the network weights iteratively. In this way we expect to find a minimum of the error function, where $\Delta E \approx 0$.

In literature a plethora of method to adjust the weights are proposed, some based just on first derivative of error, others use also the second, others use more hyperparameters than just the learning rate. Since is known that an high learning rate bring fast training but can miss the best minimum of the cost function (the point jumps around) and a low learning rate make all the training very slow the best practice is to anneal the learning rate over time. This is done by using a learning rate decay approach (an other hyperparameter).

At the state of the art the best methods are the ones that adaptively tune the learning rate (using a learning rate decay approach), and even do so per parameter. The most known are: Adagard,RMSprop,Adam.

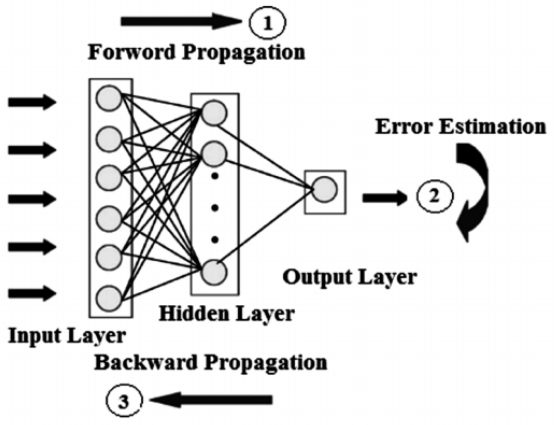


Figure 3.20: forward and backward propagation

Implemented MLP

The MLP implemented for this thesis work has the next characteristics:

- **Cost function**: Cross entropy
- **Backpropagation alghoritm**: Adam optimizer
- **nr hidden layer**: 1 for dataset A, 2 for dataset B
- **nr of neurons**: cross validation grid search between 30,40,50,60 for dataset A, cross validation grid search between 40,50,60,70 for first hidden layer and second hidden layer with the half of the neurons of the first for dataset B.
- **learning rate**: 0.001 for dataset A, 0.01 for dataset B
- **activation functions**: sigmoid for hidden layer and softmax function for output layer.

The cross entropy function is defined as

$$-\frac{1}{n}\sum_{x}\sum_{j}[y_j \ln a_j^L + (1-y_j)\ln 1 - a_j^L]$$

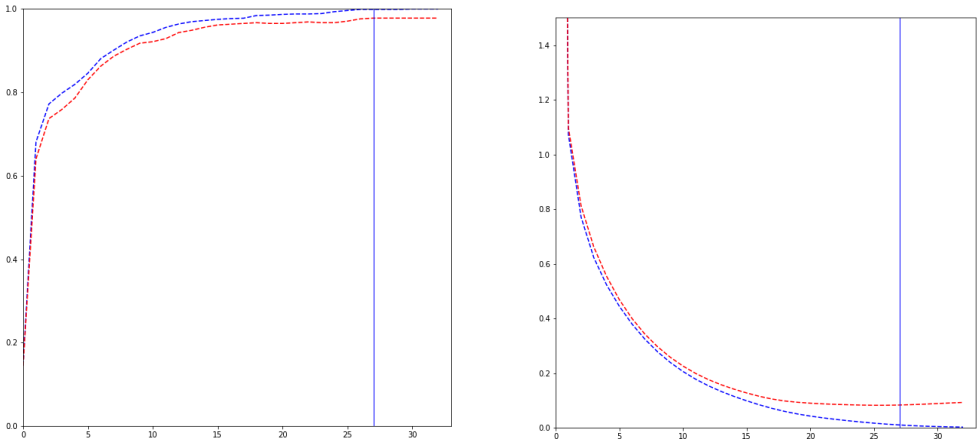
where a_j^L is the actual output value, j is the number of actual output neuron, n the total training data. It showed good performance for multiclass classification and avoid the learning slowrate. Adam optimizer belongs to the adaptive backpropagation methods and adopt also a learning decay approach, more detail in [10].
Two different architectures where utilized for each dataset. Since a 1 layer is enough to reach good accuracy for Dataset A going into a deep structure is useless. A learning rate of 0.001 is the proposed one in [10] and has good convergence time and good results. For Dataset B a deep structure with 2 layer is more suitable since it reaches the convergence in less time and give best results, for the same reason a starting learning rate of 0.01 is adopted. In both the architecture the only hyperparameters tuned are: the number of hidden neurons using a grid search and 3-fold cross validation and the number of training epochs by using the early stopping criteria.
Adam optimizer has a learning decay approach that is useful for prevent overfitting but also a early stopping strategy was adopted.

Early stopping Early stopping is a technique used in order to detect when to stop the training process when it is done in iterative way as in neural network. The procedure is:

1. Split the training data into a training set and a validation set (it can be extended also with cross validation as in the case of this thesis)
2. Train only on the training set and evaluate the per-example error on the validation set once in a while
3. Stop training as soon as the error on the validation set is higher than it was the last time it was checked.

4. Use the weights the network had in that previous step as the result of the training run.[22]

In case of this thesis instead of looking for error, the evaluation of validation set is done on accuracy (that reflect the behaviour of the cost function); and the stopping criterion is given by a patience range of 2500 epochs. This means that the MLP stops to learn if after 2500 epochs the accuracy of validation set is not improved. Example of early stopping in 3.21



(a) accuracy (b) loss

Figure 3.21: typical behaviour of early stopping in the implementation. Red stand for validation set, blue for training set

3.3.6 Postprocessing

All the classification process is done by per pixel-level instead of object based. This imply that the final classified image can suffer of salt and pepper noise.
Salt-and-pepper noise is a form of noise sometimes seen on images. It is also known as impulse noise. This noise can be caused by sharp and sudden disturbances in the image signal. It presents itself as sparsely occurring white and black pixels. In image per pixel classification instead it presents as isolated pixels classified differently from the majority of their neighboring pixels. This is a natural behaviour since not all the pixels belonging to the same class have identical features.
In order to remove such noise a filter is generally applied. The two simplest and most used ones are: the mean filter and the median filter. In particular for this thesis a median filter is applied because of the following characteristics:

- The median is a more robust average than the mean and so a single very unrepresentative pixel in a neighborhood will not affect the median value significantly.

- Since the median value must actually be the value of one of the pixels in the neighborhood, the median filter does not create new unrealistic pixel values when the filter straddles an edge. For this reason the median filter is much better at preserving sharp edges than the mean filter.

The filter is already implemented in the cv2 library of python.
The median filter considers each pixel in the image in turn and looks at its nearby neighbors to decide whether or not it is representative of its surroundings. Instead of simply replacing the pixel value with the mean of neighboring pixel values, it replaces it with the median of those values. The median is calculated by first sorting all the pixel values from the surrounding neighborhood into numerical order and then replacing the pixel being considered with the middle pixel value. (If the neighborhood under consideration contains an even number of pixels, the average of the two middle pixel values is used.) Figure 3.22 illustrates an example calculation.
The only parameter to specify is the dimension of window used to take in account the neighboring pixels. For Dataset A a window of dimension 3 is considered and for Dataset B a window of dimension 5. In figure 3.23 a comparison between original and filtered image.

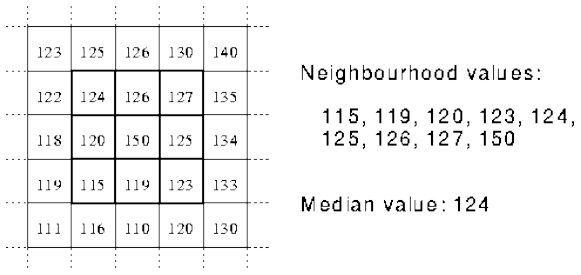
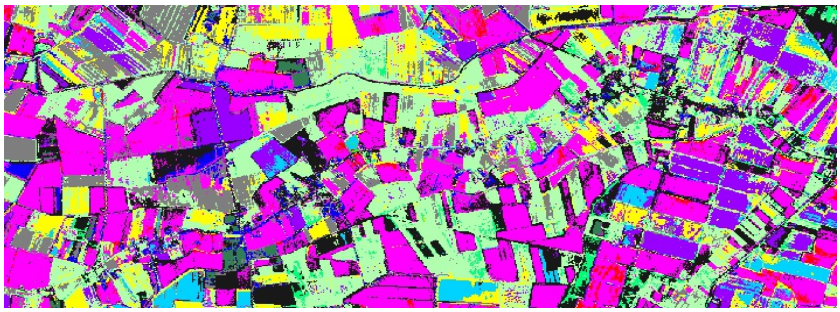
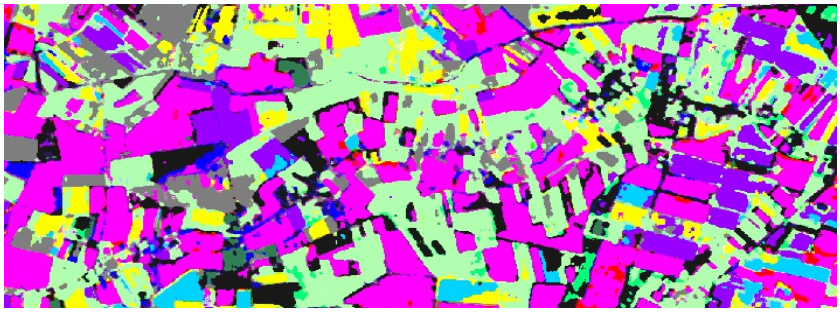


Figure 3.22: example of calculation of median filter



(a) original



(b) median filter 5x5

Figure 3.23: original clustered image in (a) and after the application of median filter 5x5 in (b)

3.4 Results

3.4.0.1 Dataset A

Scenario 1

In figure 3.24 the plot accuracy/kappascore vs Doy for monotemporal images shows that the maximum accuracy reached is 90.14% by the image of 09/04/2017. The lowest is 76.99% reached by image of 07/06/2017. With a total range of 13.55%. Same evaluations for kappascore with a maximum of 88.68% and a minimum of 73.52% with a total range of 13.16%.
In the figure 3.25 the best performances are reached by water class with 100% of PA and UA. The lowest UA is 79.11%, the lowest PA is 76.68% both of cabbage class.
In figure 3.26 the clustered image. It reflects the performances shown in the relative confusion matrix. So a perfect match of water is reached, instead by visual inspection is possible to see some overlapping for the city class. As expected the majority of fields are composed by maze with different life cycle (yellow,light blue and purple).

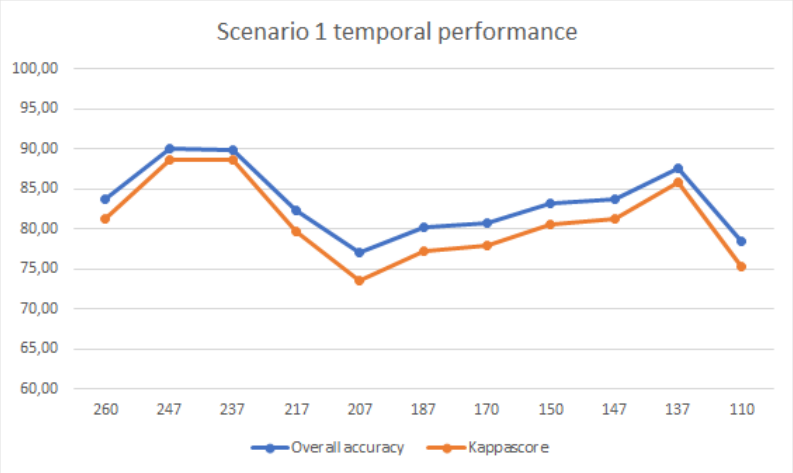


Figure 3.24: temporal performance of scenario 1

		Predicted									
	Class	0	1	2	3	4	5	6	7	All	PA(%)
Ground truth	0	**225**	0	23	3	1	1	8	9	270	83,33
	1	0	**238**	0	0	0	0	0	0	238	100
	2	3	0	**177**	0	0	0	0	0	180	98,33
	3	0	0	0	**214**	0	1	8	0	223	95,96
	4	3	0	0	0	**184**	8	1	0	196	93,87
	5	3	0	0	0	19	**110**	0	3	135	81,48
	6	9	0	0	13	0	0	**125**	16	163	76,68
	7	9	0	1	0	0	3	16	**200**	229	87,33
	All	252	238	201	230	204	123	158	228	1634	
	UA(%)	89,28	100	88,05	93,04	90,19	89,43	79,11	87,71		

0=build up, 1=water, 2=mazecultivated, 3=maze, 4=pearlmillet, 5=onion, 6=cabbage, 7=mazegrow

Figure 3.25: confusion matrix of best individual temporal information (09/04)

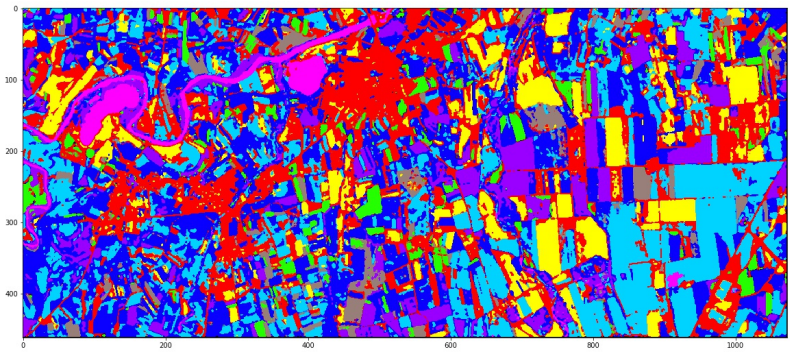


Figure 3.26: final segmentation of the image considering best individual temporal information

Scenario 2

In figure 3.27 the plot accuracy/kappascore vs Doy for multitemporal image shows that the maximum accuracy reached is 97.98% by the multitemporal image of 09/17/17_08/25/17 but since there are clouds in the images they can lead to low predictive ability. So the chosen best multitemporal image is 09/04/17_05/27/17 with an accuracy of 97.55%. The lowest is 89.66% reached by image of 07/26/17_07/06/17. With a total range of 7.89%. Same evaluations for kappascore with a maximum of 97.25% and a minimum of 88.26% with a total range of 8.99%.
In the figure 3.28 the best performances are reached by water class with 100% of PA and UA. The lowest UA is 95.18%, the lowest PA is 95.18% both of build up class. Big increase in performance can be seen for the cabbage class compared to monotemporal confusion matrix
In figure 3.29 the clustered image. It reflects the performances shown in the relative confusion matrix. So a perfect match of water is reached, instead by visual inspection is possible to see an increase of onion fields and more homogeneity in cabbage field if compared to monotemporal image. Some overlapping for the city class can still be detected. As expected the majority of fields are composed by maze with different life cycle (yellow,light blue and purple).

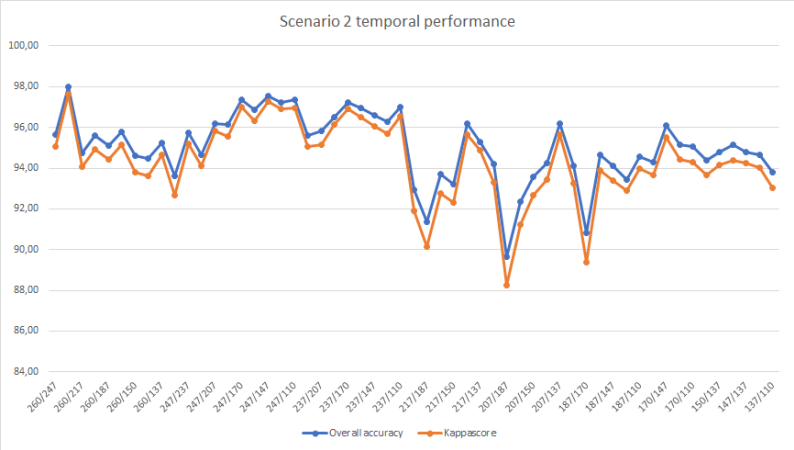


Figure 3.27: temporal performance of scenario 2

		Predicted									
Ground truth	Class	0	1	2	3	4	5	6	7	All	PA(%)
	0	**257**	0	4	2	2	2	0	3	270	95,18
	1	0	**238**	0	0	0	0	0	0	238	100
	2	2	0	**178**	0	0	0	0	0	180	98,88
	3	0	0	0	**222**	0	0	1	0	223	99,55
	4	3	0	0	1	**192**	0	0	0	196	97,95
	5	6	0	0	0	0	**126**	1	2	135	93,33
	6	2	0	0	3	1	0	**156**	1	163	95,70
	7	0	0	1	0	0	0	2	**226**	229	98,68
	All	270	238	183	228	195	128	160	232	1634	
	UA(%)	95,18	100	97,26	97,36	98,46	98,43	97,50	97,41		

0=build up, 1=water, 2=mazecultivated, 3=maze, 4=pearlmillet, 5=onion, 6=cabbage, 7=mazegrow

Figure 3.28: confusion matrix of best two temporal information (09/04_05/27)

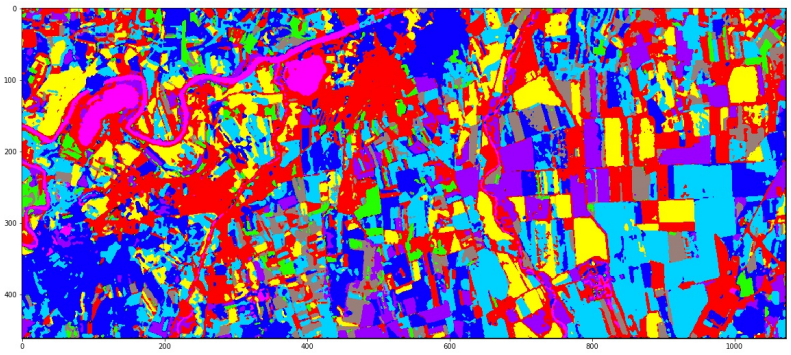


Figure 3.29: final segmentation of the image considering best two temporal information

Scenario 3

In figure 3.30 the plot accuracy/kappascore vs Doy for multitemporal image shows that the maximum accuracy reached is 98.83% by the multitemporal image of 09/04/17_06/19/17_05/17/17.

The lowest is 94.19% reached by image of 07/26/17_07/06/17_06/19/17. With a total range of 4.64%. Same evaluations for kappascore with a maximum of 98.66% and a minimum of 93.32% with a total range of 5.34%.

In the figure 3.31 the best performances are reached by water class with almost 100% of PA and UA. The lowest UA is 96.41% for build up class, the lowest PA is 97.44% for pearl millet class. Increase in performance can be seen for all the class compared to scenario 2.

In figure 3.32 the clustered image. It reflects the performances shown in the relative confusion matrix. Some overlapping for the city class can still be detected. As expected the majority of fields are composed by maze with different life cycle (yellow,light blue and purple).

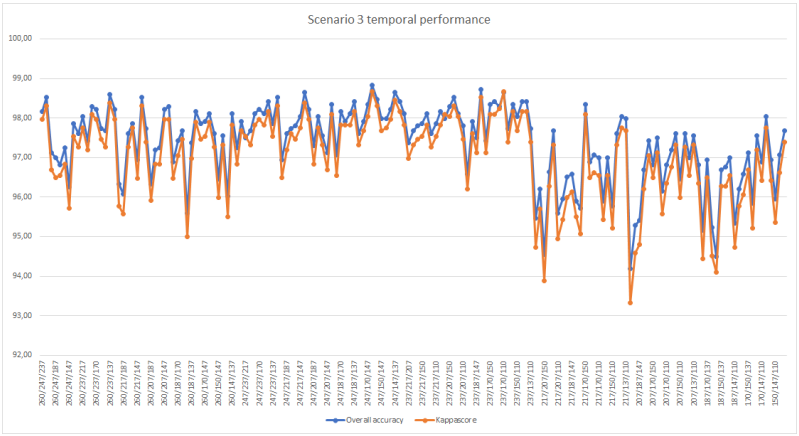


Figure 3.30: temporal performance of scenario 3

		Predicted									
	Class	0	1	2	3	4	5	6	7	All	PA(%)
Ground truth	0	**269**	0	0	1	0	0	0	0	270	99,62
	1	1	**237**	0	0	0	0	0	0	238	99,57
	2	0	0	**179**	0	0	0	0	1	180	99,44
	3	2	0	0	**220**	0	1	0	0	223	98,65
	4	2	0	0	0	**191**	0	3	0	196	97,44
	5	2	0	0	0	0	**133**	0	0	135	98,51
	6	1	0	0	1	0	0	**161**	0	163	98,77
	7	2	0	0	0	0	0	2	**225**	229	98,25
	All	279	237	179	222	191	134	166	226	1634	
	UA(%)	96,41	100	100	99,09	100	99,25	96,98	99,55		

0=build up, 1=water, 2=mazecultivated, 3=maze, 4=pearlmillet, 5=onion, 6=cabbage, 7=mazegrow

Figure 3.31: confusion matrix of best two temporal information (09/04_06/19_05/17)

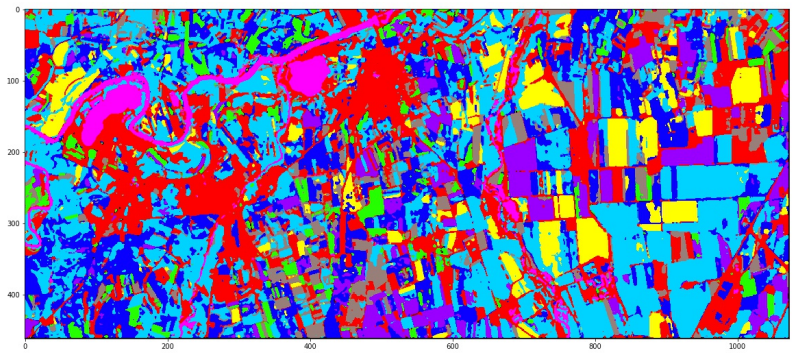


Figure 3.32: final segmentation of the image considering best three temporal information

Non-temporal features

In figure 3.33 the plot accuracy/kappascore vs non-temporal features for monotemporal image shows that the maximum accuracy reached is 94.74% by the monotemporal image with Pca1_Pca3_Txt. The lowest is 89.35% reached by image with Pca1_Pca2. With a total range of 5.35%. Same evaluations for kappascore with a maximum of 93.95% and a minimum of 87.97% with a total range of 5.98%.
In the figure 3.34 the best performances are reached by water class with almost 100% of PA and UA. The lowest UA is 86.06% for cabbage class, the lowest PA is 85.92% for onion class. Increase in performance can be seen for all the class compared to simple monotemporal image, especially the build up class have a big increase in

performance as expected.
In figure 3.35 the clustered image. It reflects the performances shown in the relative confusion matrix. By visual inspection the overlapping for the city class is reduced compared to simple monotemporal image. As expected the majority of fields are composed by maze with different life cycle (yellow,light blue and purple).

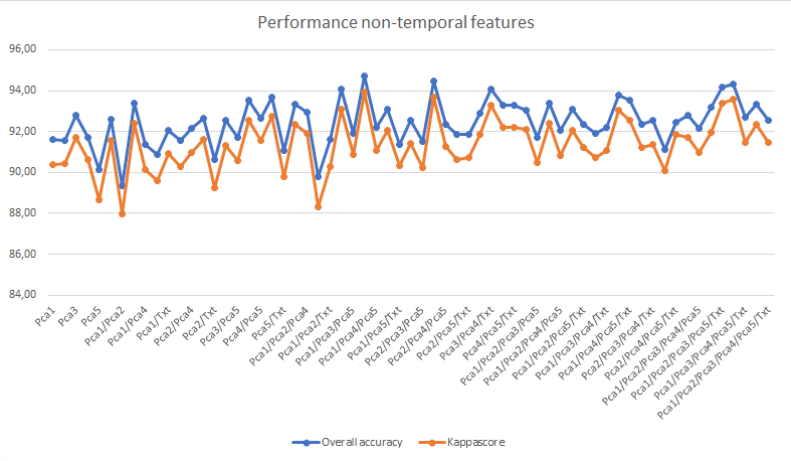


Figure 3.33: performance of combination of non temporal features

	Predicted										
	Class	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**All**	**PA(%)**
Ground truth	**0**	**262**	0	3	0	0	0	5	0	270	97,03
	1	0	**237**	0	0	0	0	0	1	238	99,57
	2	7	0	**173**	0	0	0	0	0	180	96,11
	3	0	0	0	**215**	0	0	8	0	223	96,41
	4	2	0	0	0	**188**	4	2	0	196	95,91
	5	0	0	0	0	15	**116**	3	1	135	85,92
	6	3	0	0	7	0	2	**142**	9	163	87,11
	7	5	0	0	0	0	4	5	**215**	229	93,88
	All	279	237	176	222	203	126	165	226	1634	
	UA(%)	93,90	100	98,29	96,84	92,61	92,06	86,06	95,13		

0=build up, 1=water, 2=mazecultivated, 3=maze, 4=pearlmillet, 5=onion, 6=cabbage, 7=mazegrow

Figure 3.34: confusion matrix of best non-temporal features (Pca1_Pca3_Txt)

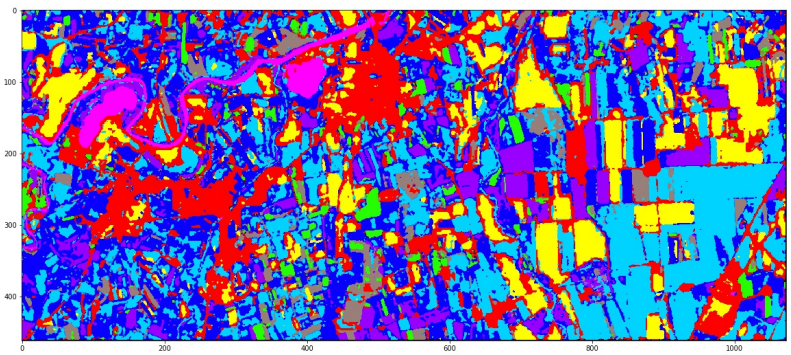


Figure 3.35: final segmentation of the image considering best non-temporal features

Features comparison
In figure 3.36 is shown the accuracy/kappascore vs features. It's possible to see that an increase in the number of features lead to an increase of both scores. With total range of 10% between monotemporal image and the multitemporal with window of 3 Doy. The right choice of non-temporal features results in a good increase of accuracy of 4% while the use of multitemporal features lead to a strongly increase of accuracy. The graph show also that a multitemporal windows of dimension 3 is enough since increasing the number of features lead to a plateau of the curve.

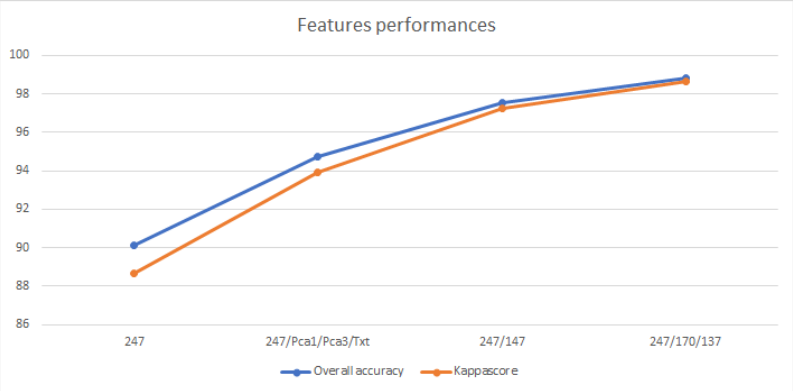


Figure 3.36: plot accuracy vs features used

3.4.0.2 Dataset B

Scenario 1
In figure 3.37 the plot accuracy/kappascore vs Doy for monotemporal images shows that the maximum accuracy reached is 86.94% by the image of 07/04/2015. The lowest is 77.83% reached by image of 10/22/2015. With a total range of 9.11%. Same evaluations for kappascore with a maximum of 84.25% and a minimum of 73.54% with a total range of 9.11%.
In the figure 3.38 the best performances are reached by water class with 98% of PA and UA. The lowest UA is 49.24%, the lowest PA is 35.63% both of temporary grass class.
In figure 3.39 the clustered image. It reflects the performances shown in the relative confusion matrix. So an almost perfect match of water is reached. The majority of fields are composed by common wheat, maize and lucern.

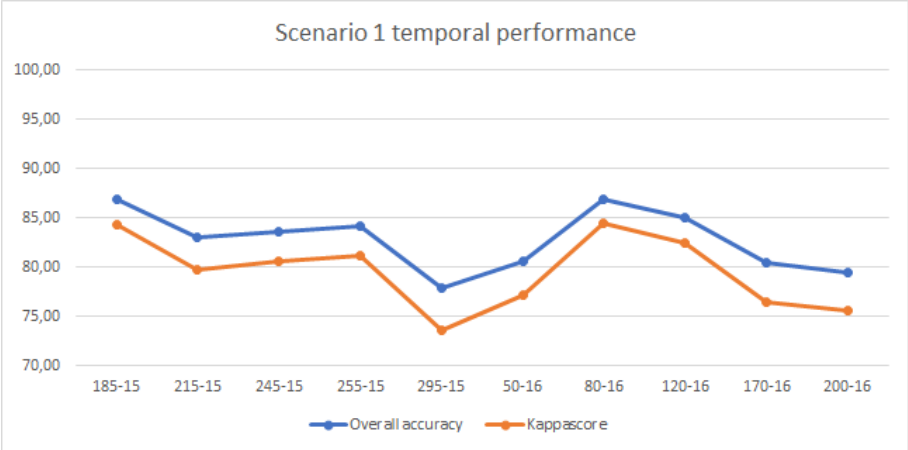
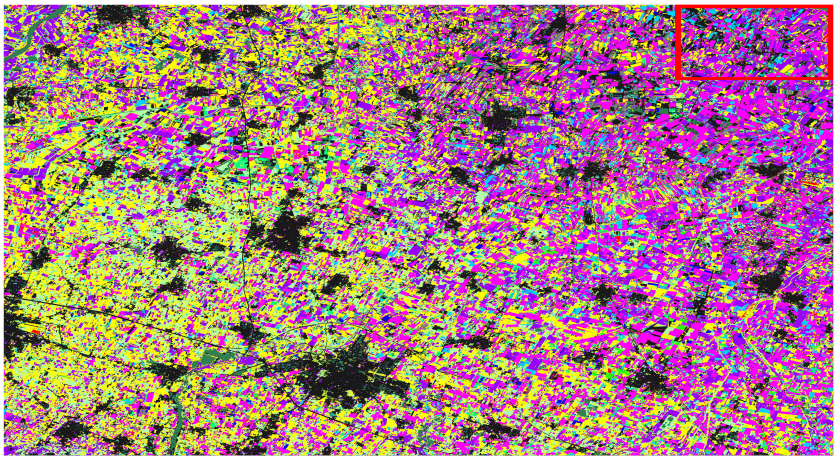


Figure 3.37: temporal performance of scenario 1

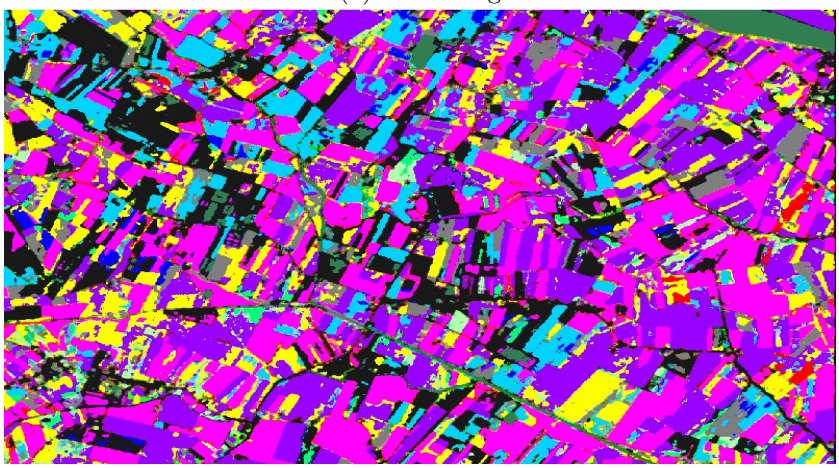
		Predicted													
Ground Truth	Class	0	1	2	3	4	5	6	7	8	9	10	11	All	PA(%)
	0	**230**	105	28	0	1	0	4	1	1	0	0	0	370	62,16
	1	37	**3529**	109	15	1	0	1	20	25	18	13	21	3789	93,13
	2	49	34	**2388**	128	2	6	14	11	38	86	2	24	2782	85,83
	3	0	3	135	**2308**	5	25	0	20	1	89	0	5	2591	89,07
	4	0	1	0	2	**93**	0	0	4	1	0	0	0	101	92,07
	5	0	2	3	21	0	**82**	0	4	0	44	0	7	163	50,30
	6	2	5	13	0	0	0	**217**	2	1	0	0	9	249	87,14
	7	11	27	101	9	2	2	0	**293**	7	12	0	1	465	63,01
	8	0	72	53	2	0	1	1	19	**98**	20	0	9	275	35,63
	9	0	12	106	14	3	10	0	6	8	**930**	0	13	1102	84,39
	10	0	1	0	0	0	0	0	0	0	0	**1452**	15	1468	98,91
	11	2	88	78	11	0	10	19	2	19	8	10	**1385**	1632	84,86
	All	331	3879	3014	2510	107	136	256	382	199	1207	1477	1489	14987	
	UA(%)	69,48	90,97	79,23	91,95	86,91	60,29	84,76	76,70	49,24	77,05	98,30	93,01		

0=barley, 1=common wheat, 2=lucern, 3=maize, 4=other cereals, 5=peer, 6=rye, 7=soya, 8=temporary grass, 9=vineyard, 10=water, 11=build up

Figure 3.38: confusion matrix of best individual temporal information (07/04/15)



(a) total image



(b) zoom upper-right corner

Figure 3.39: Total segmented image in (a) and a zoom of 600x800 px in the upper right corner

Scenario 2

In figure 3.40 the plot accuracy/kappascore vs Doy for multitemporal images shows that the maximum accuracy reached is 97.21% by the image of 07/04/15_03/20/16. The lowest is 92.70% reached by the temporal window image of 09/02/15_09/12/15 . With a total range of 4.51%. Same evaluations for kappascore with a maximum of 96.75% and a minimum of 91.30% with a total range of 5.45%.

In the figure 3.41 the best performances are reached by water class with almost 100% of PA and UA. The lowest UA is 82.03%, the lowest PA is 84.04% both of peer class.

In figure 3.42 the clustered image. It reflects the performances shown in the relative confusion matrix. So what in ground truth is water is almost perfectly predicted but by visual analysis also some overlap with this class is present. The majority of fields are composed by common wheat, maize and lucern.

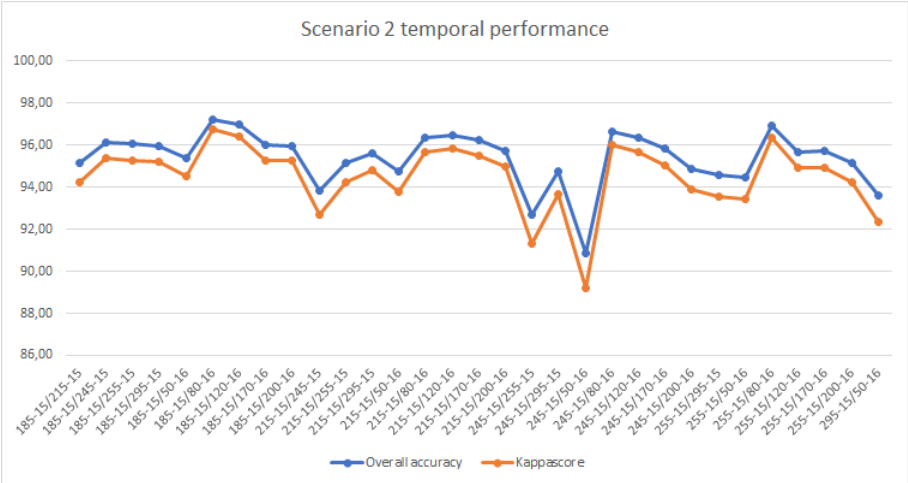
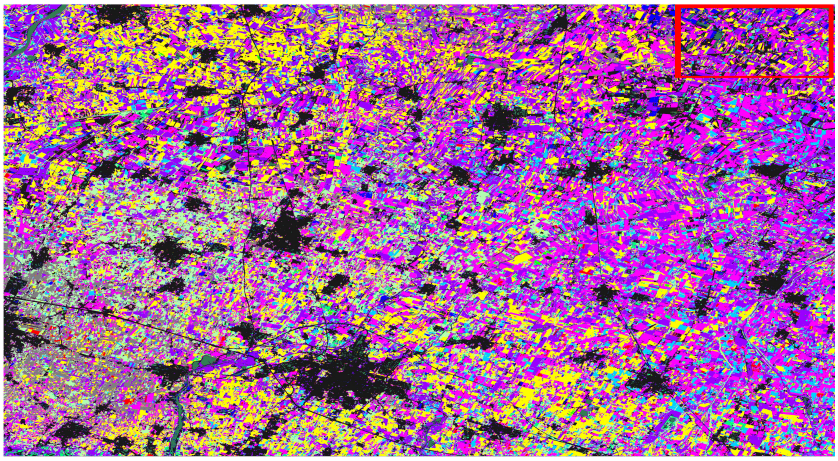


Figure 3.40: temporal performance of scenario 1

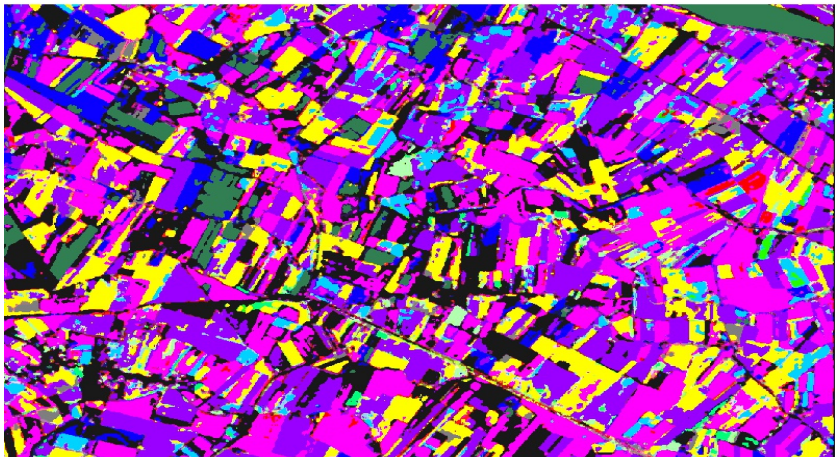
		Predicted													
Ground Truth	Class	0	1	2	3	4	5	6	7	8	9	10	11	All	PA(%)
	0	**363**	7	0	0	0	0	0	0	0	0	0	0	370	98,10
	1	3	**3705**	18	0	0	0	1	0	12	35	0	15	3789	97,78
	2	0	13	**2715**	13	1	8	0	12	3	17	0	1	2783	97,55
	3	0	0	18	**2555**	0	4	0	6	2	6	0	0	2591	98,61
	4	0	1	2	0	**94**	1	0	3	0	0	0	0	101	93,06
	5	0	0	2	4	0	**137**	0	1	0	19	0	0	163	84,04
	6	1	6	0	0	0	0	**235**	3	2	0	0	2	249	94,37
	7	0	5	17	7	2	7	0	**416**	1	10	0	0	465	89,46
	8	0	3	1	0	0	0	0	0	**270**	1	0	0	275	98,18
	9	0	12	17	8	3	8	0	3	5	**1043**	0	3	1102	94,64
	10	0	0	0	0	0	0	0	0	0	0	**1467**	1	1468	99,93
	11	2	23	4	5	0	2	3	0	3	9	3	**1577**	1631	96,68
	All	369	3775	2794	2592	100	167	239	444	298	1140	1470	1599	14987	
	UA(%)	98,37	98,14	97,17	98,57	94	82,03	98,32	93,69	90,60	91,49	99,79	98,62		

0=barley, 1=common wheat, 2=lucern, 3=maize, 4=other cereals, 5=peer, 6=rye, 7=soya, 8=temporary grass, 9=vineyard, 10=water, 11=build up

Figure 3.41: confusion matrix of best two temporal information (07/04/15_03/20/16)



(a) total image



(b) zoom upper-right corner

Figure 3.42: Total segmented image in (a) and a zoom of 600x800 px in the upper right corner

Scenario 3

In figure 3.43 the plot accuracy/kappascore vs Doy for multitemporal images shows that the maximum accuracy reached by considering the not-cloudy combinations is 98.57% by the image of 07/04/15_09/04/15_03/20/16. The lowest is 96.59% reached by the temporal window image of 09/02/15_09/12/15_10/22/15 . With a total range of 2%. Same evaluations for kappascore with a maximum of 98.30% and a minimum of 95.94% with a total range of 2.36%.

In the figure 3.44 the best performances are reached by water class with almost 100% of PA and UA but very good performances can be seen from all classes. The lowest UA is 93.02% of peer class, the lowest PA is 96.34% of soya class.

In figure 3.45 the clustered image. It reflects the performances shown in the relative confusion matrix. Less overlap in water class can be seen respect to multitemporal image of window 2 and the reduction of other cereals crops. The majority of fields are composed by common wheat, maize and lucern.

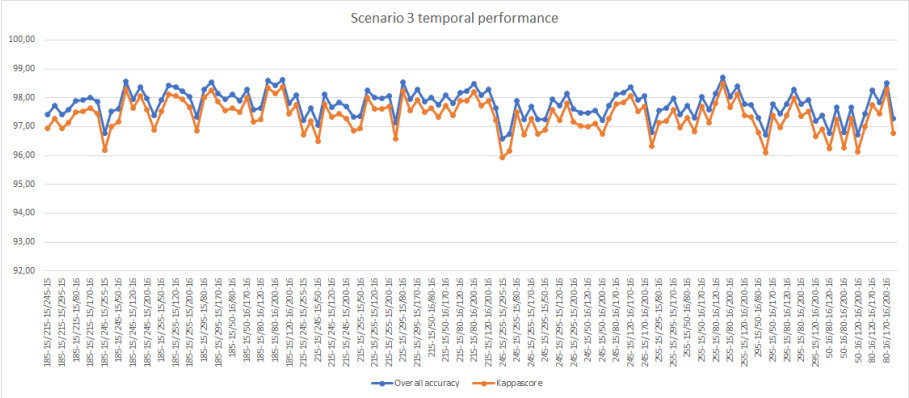
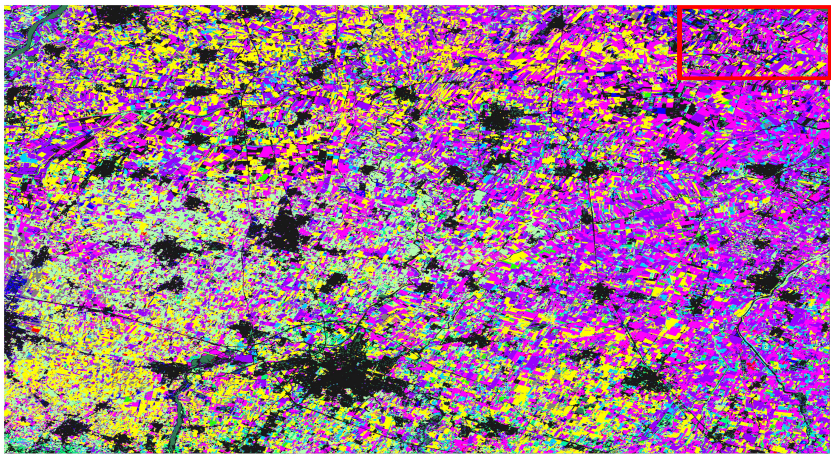


Figure 3.43: temporal performance of scenario 1

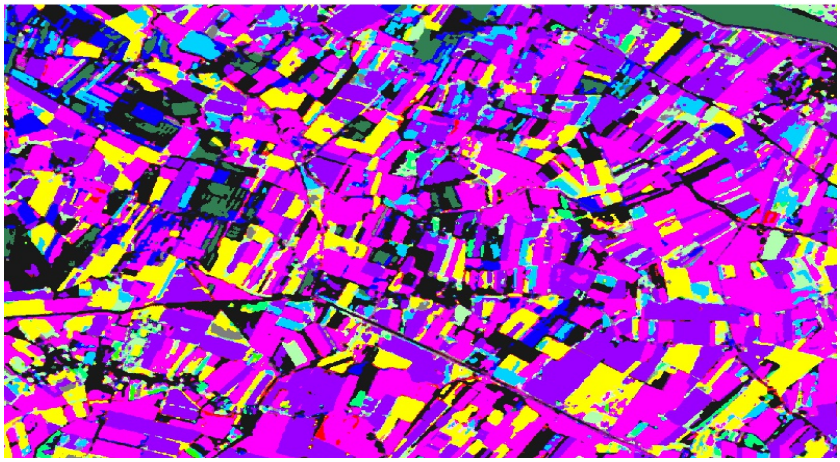
		Predicted													
	Class	0	1	2	3	4	5	6	7	8	9	10	11	All	PA(%)
Ground Truth	0	369	1	0	0	0	0	0	0	0	0	0	0	370	99,72
	1	8	3735	15	1	0	0	1	1	3	11	0	14	3789	98,57
	2	0	9	2760	1	0	0	0	4	5	3	0	0	2782	99,20
	3	0	0	11	2565	0	2	0	5	3	4	0	1	2591	98,99
	4	0	0	0	0	99	0	0	1	1	0	0	0	101	98,01
	5	0	0	1	0	0	160	0	0	0	2	0	0	163	98,15
	6	3	2	0	0	0	0	244	0	0	0	0	1	250	97,6
	7	0	5	3	2	2	3	0	448	0	2	0	0	465	96,34
	8	0	0	3	0	0	0	0	0	270	1	0	1	275	98,18
	9	0	12	12	1	0	5	1	3	5	1060	0	3	1102	96,18
	10	0	0	0	0	0	0	0	0	0	0	1468	0	1468	100
	11	0	11	1	7	0	2	5	0	3	4	4	1594	1631	97,73
	All	380	3775	2806	2577	101	172	251	462	290	1087	1472	1614	14987	
	UA(%)	97,10	98,94	98,36	99,53	98,01	93,02	97,21	96,96	93,10	97,51	99,72	98,76		

0=barley, 1=common wheat, 2=lucern, 3=maize, 4=other cereals, 5=peer, 6=rye, 7=soya, 8=temporary grass, 9=vineyard, 10=water, 11=build up

Figure 3.44: confusion matrix of best two temporal information (07/04/15_09/04/15_03/20/16)



(a) total image



(b) zoom upper-right corner

Figure 3.45: Total segmented image in (a) and a zoom of 600x800 px in the upper right corner

Non-temporal features

In figure 3.46 the plot accuracy/kappascore vs non-temporal features for monotemporal image shows that the maximum accuracy reached is 90.39% by the monotemporal image with Pca1_Pca5_Txt. The lowest is 85.72% reached by image with Pca2. With a total range of 4.67%. Same evaluations for kappascore with a maximum of 88.55% and a minimum of 82.95% with a total range of 5.55%.

In the figure 3.47 the best performances are reached by water class and city class. The lowest UA is 67.40% for peer class, the lowest PA is 47.81% for temporary grass class. Increase in performance can be seen for all the class compared to simple monotemporal image, especially the build up class have a big increase in performance as expected.

In figure 3.48 the clustered image. It reflects the performances shown in the relative confusion matrix. So what in ground truth is water is almost perfectly predicted but by visual analysis also some overlap with this class is present. The majority of fields are composed by common wheat, maize and lucern. Respect to the simple monotemporal image the predicted lucern fields are less.

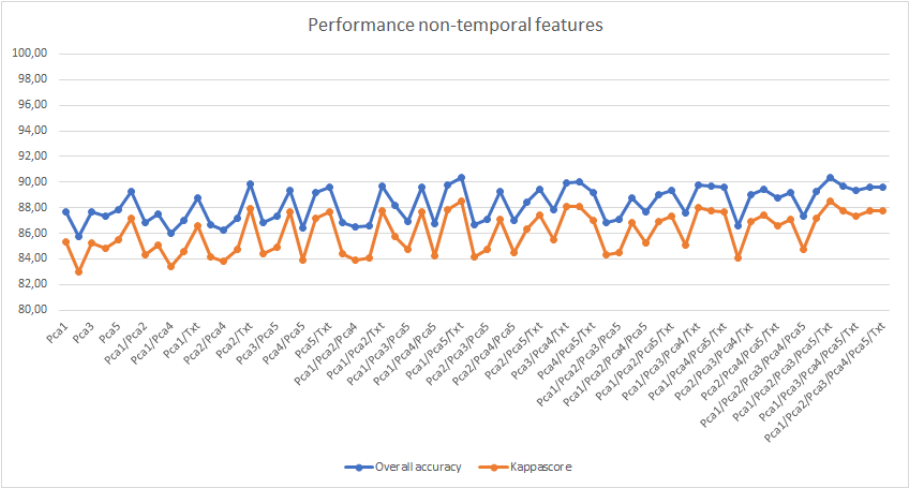
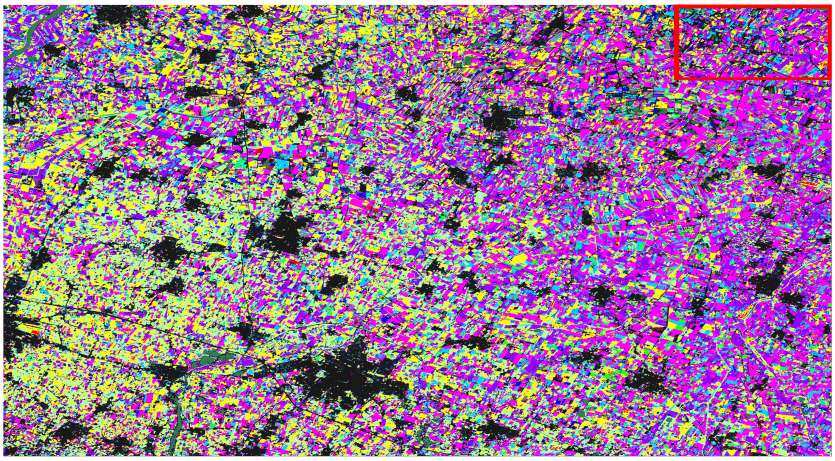


Figure 3.46: performance of combination of non temporal features

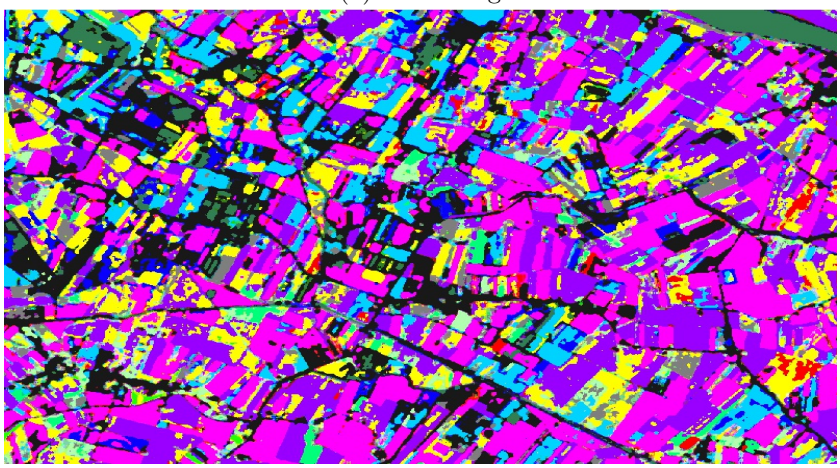
		Predicted													
Ground Truth	Class	0	1	2	3	4	5	6	7	8	9	10	11	All	PA(%)
	0	**234**	104	32	0	0	0	0	0	0	0	0	0	370	63,24
	1	23	**3598**	84	5	0	1	0	11	18	28	0	21	3789	94,95
	2	31	22	**2577**	45	1	3	5	18	17	60	0	3	2782	92,63
	3	0	3	107	**2323**	6	28	0	15	4	94	0	11	2591	89,65
	4	0	3	0	7	**89**	0	0	2	0	0	0	0	101	88,11
	5	0	1	3	9	1	**91**	0	5	1	52	0	0	163	55,82
	6	0	0	10	0	0	0	**226**	5	2	0	1	6	250	90,4
	7	9	35	67	18	1	2	3	**307**	6	14	0	3	465	66,02
	8	0	53	38	0	0	5	2	7	**131**	30	0	8	274	47,81
	9	0	19	93	13	0	3	0	6	4	**963**	0	1	1102	87,38
	10	0	3	0	0	0	0	2	0	0	0	**1457**	6	1468	99,25
	11	0	26	21	4	0	2	9	9	6	1	4	**1550**	1632	94,97
	All	297	3867	3032	2424	98	135	247	385	189	1242	1462	1609	**14987**	
	UA(%)	78,78	93,04	84,99	95,83	90,81	67,40	91,49	79,74	69,31	77,53	99,65	96,33		

0=barley, 1=common wheat, 2=lucern, 3=maize, 4=other cereals, 5=peer, 6=rye, 7=soya, 8=temporary grass, 9=vineyard, 10=water, 11=build up

Figure 3.47: confusion matrix of best non-temporal features (Pca1_Pca5_Txt)



(a) total image



(b) zoom upper-right corner

Figure 3.48: Total segmented image in (a) and a zoom of 600x800 px in the upper right corner

Features comparison

In figure 3.49 is shown the accuracy/kappascore vs features. It's possible to see that an increase in the number of features lead to an increase of both scores. With total range of 11.63% between monotemporal image and the multitemporal with window of 3 Doy. The right choice of non-temporal features results in a good increase of accuracy of 3.45% while the use of multitemporal features lead to a strongly increase of accuracy. The graph show also that a multitemporal windows of dimension 3 is enough since increasing the number of features lead to a plateau of the curve.

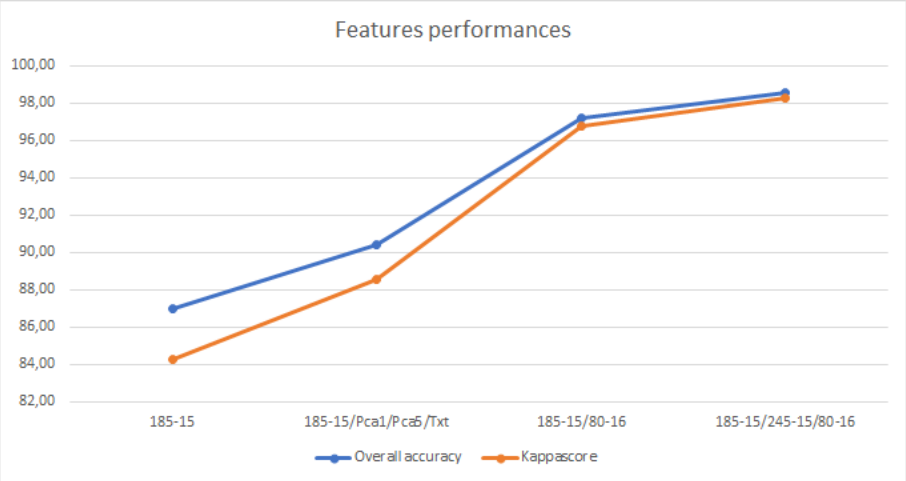


Figure 3.49: plot accuracy vs features used

Chapter 4

Conclusion

The thesis is composed by three main parts, each of them has a different aim.

- The aim of the first chapter is to give a general overview of machine learning framework focused on a supervised approach, in particular the branch applied to classification is exploited. The main steps of machine learning (data acquisition, pre-processing, feature extraction, feature selection, training algorithm) are explained and for each of them the main concepts are investigated in a general way. In the same section the most common problems connected to the topic are exposed and at the same time a general overview of the best known solutions is given. Big importance is given also to the different criteria of evaluation of a classifier; so the main measures for both binary (ROC, sensitivity, specificity) and multiclass(confusion matrix, kappascore, producer and user accuracy) classification are analyzed. Very important is also the assessment methods section since contains information about underfitting and overfitting, the two biggest problem in machine learning and also contains the description of hold out and cross validation methods both used in the thesis implementations. Summarize, this chapter can be seen as a collection of procedures for machine learning applied to classification that represent a theoretical basis for the next chapters. I did not find during my research all these information grouped in one file; but its main limitation is due to the general level of discussion; so I suggest to use this chapter just as an overview and then use other sources to go deep in the topics.

- The aim of the second chapter is to explain in detail the paper "Topological Graph Kernel on Multiple Thresholded Functional Connectivity Networks for Mild Cognitive Impairment Classification" written by Biao Jie et al of which I have implemented the sub-methods that compose the general structure. Analyzing in the detail the general framework and also the techniques used for detection of MCI is possible to extrapolate useful methods applicable in different fields covered by machine learning; especially the ones that has to deal with connectivity matrix that are very common in bio-medical field for study brain activity. I analyzed in detail each step of the first chapter from the point of view of the paper. The most significant techniques analyzed are: the Weisfeiler-Lehman graph kernel, the RFE-GK, the ttest for feature selection and the learning algorithm used is SVM with different kernel approach

validated by a nested cross-validation. A general overview of the results and limitations obtained by the paper is synthesized in the end of the chapter and them demonstrate that the general framework that uses 5 threshold has a very good capability of detection the MCI in subjects. Comparison between different and more classic techniques have shown that the best results are given by the use of RFE-GK method combined with 5 thresholds. The presented techniques in this chapter were implemented. The environment used is Matlab with STATISTICAL toolbox and the libsvm library. The code is modularized so for each step or sub-step I built a function able to give the correct output; such functions have been tested by using random data. I was not able to test the general framework because the total complexity of the code is too high and it was not possible to obtain results in acceptable computational time using commercial hardware. To solve this problem is necessary to use proper hardware and optimize the software. Anyway the codes can represent a little library composed of method such RFE-GK, the Weisfeiler-Lehman kernel, Ttest... useful for further analysis that involves such methods.

- My major contribution is demonstrated in the third chapter of the thesis and a new general method for crop classification using satellite image is exploited. Information about the two dataset used were given and the structure of the first chapter is followed in order to explain all the fundamental steps of the procedure. The learning algorithm used is a MLP with a different architecture for each dataset. The used raw features are extracted from the Copernicus website that give direct access to the products of Sentinel-2 satellite. The results obtained confirm the fundamental importance of using multitemporal features instead of a monotemporal one with an accuracy increased (from best monotemporal to best 3 window multitemporal) of around 9% for dataset A and around 11% for dataset B. Not only the importance of multitemporal information is confirmed but it is shown that the different crop phenology strongly influence the final accuracy, so the right choice of best multitemporal features is of maximum importance. In this thesis the best choice derived from combinations of multitemporal windows of dimension one,two and three. Since the use of multitemporal features imply bigger effort in data acquisition and more computational complexity due to adding more features; an analysis of monotemporal information plus some non temporal features (Pca and textural)is done. It showed a considerable increase in accuracy respect simple monotemporal image. The increase for dataset A is around 5% and for dataset B is around 4%. Looking at image 3.36 and 3.49 is possible to see that for both datasets the window range of 3 Doy (in 1 year) is enough to obtain very good results since the trend of the graph shows that a plateau is reached. The combination approach is very computational expensive and the results (at least for Dataset B) are obtained by using the computational resources provided by hpc@polito (http://www.hpc.polito.it). In particular 24 CPUs and 1 gpu were used and the computational time varied dependently from number of combinations, number of features and dataset. This approach limited the use of cross validation only for hyperparameters selection and an outer cross validation for generality assessment wasn't implemented. This imply that the numerical results showed are consequence of a 50-50 hold out

method and they are subject to fluctuation, negligible for the treated topic, when the experiments are repeated and the best chosen multitemporal image can change; but the general methodology and extracted results are valid and lead to good accuracies that outperform some literature's approaches. A limit of the machine learning approach in crop classification is that once a trained network is obtained, cannot garantee the same results in terms of accuracy for areas different from the considered dataset, this is due to the strong influence of spatial location in crop phenology which lead to strong differences in intra-spectral information that cause misleading predictions.
The main knowledge acquired in this chapter concern:

- The use of Python programming language and specifically Anaconda and Spyder IDE
- The use of of tensorflow library that is the actual state of the art for construction of machine learning algorithms
- The use of SNAP tool provided from ESA for elaboration of Sentinel-2 images
- The use of Qgis for construction of Dataset B
- The use of a linux-based cluster computer for running the most computational expensive programs.

Although good results and important evaluations are extracted from this thesis, some further improvements can be added in future work. In the following, some suggestions are listed:

- Taking into account also the spatio-temporal crops information, that means consider the different life-cycle for the same crop in the same monotemporal image. Since same crops have different behaviour at the same time, due to the presence of a lot of farmers with their own crop life-cycle strategy.
- Reduce or remove the computational effort of using combinations, for example by detection of important information of spectral signature of crops.
- Implementation of a nested-cross validation as consequence of a reduced computational cost.
- Make a comparison with object-based methods.

Acronyms

ACC Accuracy. 24

AD Alzheimer disease. 40

ANN Artificial neural network. 86

AUC Area under the curve. 27

B Blue band. 81

BOLD Blood oxygenation level dependent. 42

CNN Convolutional neural network. 69

CSF Cerebrospinal fluid. 42

CV Cross validation. 33

EEG electroencephalograph. 39

ESA European Space Agency. 71, 73

EUROSTAT European Parliament and the European Statistical Office. 73

FMRI Functional Magnetic Resonance Imaging. 39

FN False negative. 24

FP False positive. 24

FPR False positive rate. 26

G Green band. 81

GK Graph kernel. 40

GLCM Gray level co-occurrence matrix. 79

GM Gray matter. 42

GPF Graph Processing Framework. 73

GPS Global positioning system. 73

LOOCV Leave one-out cross validation. 33

LUCAS Land Use/Cover Area frame statistical Survey. 73

MCI Mild cognitive impairment. 38

MKL Multiple kernel learning. 61

MLP Multilayer Perceptron. 69

MRI magnetic resonance imaging. 39

NC Normal condition. 40

NDVI Normalized difference vegetation index. 69, 76

NIR Near infra red. 76, 81

OCR Optical character reconison. 12

PA Producer accuracy. 30

PCA Principal Component Analysis. 77

QGIS Quantum geographic information system. 73

R Red band. 81

RF Random Forest. 69

RFE Recursive features elimination. 40

ROC Receiver operating characteristic. 28

ROI Region of interest. 42

SNAP Sentinel application platform. 72

SVM Support vector machine. 22

TN True negative. 24

TP True positive. 24

TPR True positive rate. 26

TWDTW Time-Weighted Dynamic Time Warping. 69

UA User accuracy. 29

WM White matter. 42

Appendix A

Insights

A.1 From brain images to fRMI time series

Neuronal clusters involved in brain activity consume more oxygen compared to their baseline state. Due to neurovascular coupling, blood flow and volume are increased and lead to a significant overcompensation of the oxygen demands, i.e., the ratio of oxygenated and deoxygenated hemoglobin is altered. Deoxygenated hemoglobin is paramagnetic and acts as an endogenous contrast agent since it alters the T2*-weighted MR images. This gives rise to the blood-oxygen-level-dependent (BOLD) signal, discovered in the 1990s, which has allowed MRI to become functional (fMRI) and to observe the brain at work. MRI allows sampling a three-dimensional (3-D) volume of the brain at millimetric spatial resolution every 1–3 s (or faster with recent sequences). In this way is obtained a multivariate time series of brain activity. In particular from fMRI the resulting 3-D image is composed by voxels. The extension in 3-D of a pixel. And it is the minimum spatial unit which has a fMRI time series. A voxel typically contains a few million neurons and tens of billions of synapses, with the actual number depending on voxel size and the area of the brain being imaged. So for each voxel a BOLD time series is associated. Raw fMRI signals suffer from low signal-to-noise ratio and need to be processed heavily to be amenable to analysis. Several preexisting open source software packages allow reliable results to be obtained rapidly. The main preprocessing steps, illustrated in figure A.2, are to realign the volumes to compensate for subject motion and ensure voxel-to-voxel correspondence across time, coregister functional images to a high-resolution structural image, and normalize the data into a common reference space so that subjects can be compared and existing anatomical knowledge can be leveraged. Once this is achieved, representative time series can be extracted from different brain regions (called ROIs) and serve as a basis for brain graph construction. In literature different ways of extracting brain region are analyzed (one of the most preferred is Atlas-based).
Synthesizing, thanks to fMRI and parcellation techniques, a time series for each individuated Region of interest is extracted.

A.2 Connectivity matrix

For a weighted graph with vertex set V, the connectivity matrix (also called adjacency matrix) is a square $|V| \times |V|$ matrix A such that its element A_{ij} is the weight

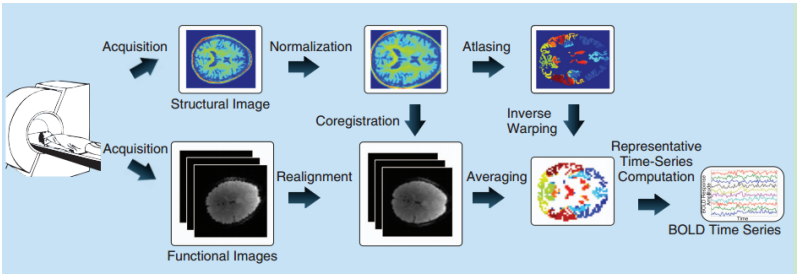


Figure A.1: Example of preprocessing for time series extraction from fMRI data, including atlas-based parcellation of the brain.

of the edge from vertex i to vertex j, and zero when there is no edge. The diagonal elements of the matrix are all zero, since edges from a vertex to itself (loops) are not allowed in simple graphs.
For this thesis, given the fMRI time series for each ROI, the Pearson Correletion Coefficient for each ROI is calculated. So the connectivity matrices considered represent as vertices the ROIs and as edges of graph the Pearson Coefficient between each ROI. This imply that the values of entries in the matrices range between -1 and 1. Specifically the matrices are symmetric with ones on the diagonal.

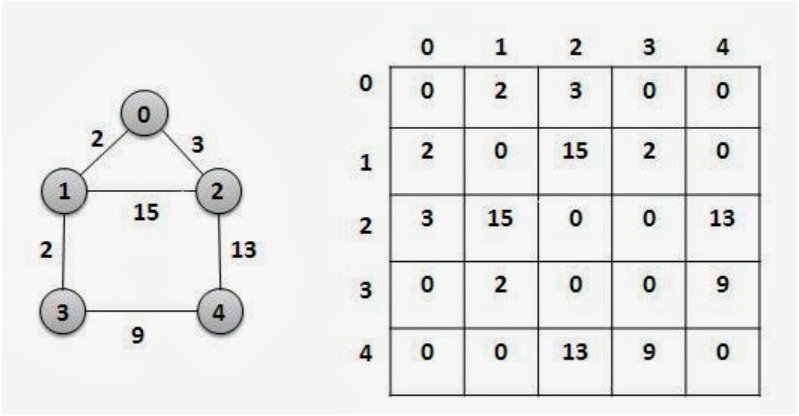


Figure A.2: Example adjacency matrix

A.3 Combinations

a combination is a selection of items from a collection, such that the order of selection does not matter. More formally, a k-combination of a set S is a subset of k distinct elements of S. If the set has n elements, the number of k-combinations is equal to the binomial coefficient

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots 1}$$

which can be written using factorials as $\frac{n!}{k!(n-k)!}$ whenever $k < n$, and which is zero when $k > n$. The set of all k-combinations of a set S is often denoted by $\binom{S}{k}$.

A.4 Covariance matrix

Given the column vector

$$\mathbf{X} = \begin{bmatrix} X_1 \\ . \\ . \\ . \\ X_n \end{bmatrix}$$

where $X_1 \cdots X_n$ are random variables, each with finite variance, then the covariance matrix Σ is the matrix whose (i, j) entry is the covariance $\Sigma_{ij} = cov(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)] = E[X_i X_j] - \mu_i \mu_j$ where the operator E denotes the expected (mean) value of its argument, and $\mu_i = E(X_i)$ is the expected value of the i th entry in the vector X. In other words,

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1 - \mu_1)(X_1 - \mu_1)] & \mathrm{E}[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_1 - \mu_1)(X_n - \mu_n)] \\ \mathrm{E}[(X_2 - \mu_2)(X_1 - \mu_1)] & \mathrm{E}[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_2 - \mu_2)(X_n - \mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n - \mu_n)(X_1 - \mu_1)] & \mathrm{E}[(X_n - \mu_n)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix}.$$

The covariance matrix generalizes the notion of variance to multiple dimensions. The covariance is a measure used to indicate the extent to which two random variables change in tandem. It can be positive or negative and its values can range between $-\infty$ and $+\infty$. Because the covariance of the ith random variable with itself is simply that random variable's variance, each element on the principal diagonal of the covariance matrix is the variance of one of the random variables. Because the covariance of the ith random variable with the jth one is the same thing as the covariance of the jth random variable with the ith one, every covariance matrix is symmetric. In addition, every covariance matrix is positive semi-definite.

Bibliography

[1] Wales Alzheimers's Society operates in England and Northern Ireland. *What is mild cognitive impairment (MCI)?* 2015. URL: https://www.alzheimers.org.uk/download/downloads/id/1773/factsheet_what_is_mild_cognitive_impairment_mci.pdf.

[2] Ozdarici-Ok Asli, Ali Ozgun OK, and Schindler Konrad. "Mapping of Agricultural Crops from Single High-Resolution Multispectral Images—Data-Driven Smoothing vs. Parcel-Based Smoothing". In: *Remote sensing* (2015).

[3] Stam CJ et al. "Graph theoretical analysis of magnetoencephalographic functional connectivity in Alzheimer's disease". In: *Brain* (2009).

[4] European Commission and European Space Agency. *Copernicus observing the heart.* 2018. URL: https://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Overview3.

[5] European Commission and European Space Agency. *Sentinel-2 MSI Technical Guide.* URL: https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-2-msi.

[6] European Commission and European Space Agency. *Step, Science toolbox exploitation platform.* URL: http://step.esa.int/main/toolboxes/snap/.

[7] Wee CY, Yap PT, and Li W. "Enriched white matter connectivity networks for accurate identification of MCI patients". In: *Neuroimage* (2011).

[8] Zhang D et al. "Multimodal classification of Alzheimer's disease and mild cognitive impairment". In: *Neuroimage* (2011).

[9] Dai Dai, Wang Jieuiong, and He Huiguang. "Classification of ADHD children through multimodal magnetic resonance imaging". In: *The frontiers in Neuroscience* (2012).

[10] Kingma Diederik P. and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (2014).

[11] Cawley Gavin C. and Talbot Nicola L. "On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation". In: *Journal of Machine Learning Research* (Oct. 2010).

[12] Github. *CS231n Convolutional Neural Networks for Visual Recognition.* URL: http://cs231n.github.io/neural-networks-1/.

[13] Isabelle Guyon and Andre Elisseeff. *The elements of statistical learning.* Vol. 207. Berlin, Heidelberg: Springer, 2006. ISBN: 978-3-540-35487-1.

[14] Das H. "Satellite based agro advisory services". In: *Satellite remote sensing and GIS applications in agricultural meteorology, Proceedings of the Training Workshop* (2003).

[15] Shen H et al. "Discriminative analysis of resting-state functional connectivity patterns of schizophrenia using low dimensional embedding of fMRI". In: *Neuroimage* (2010).

[16] Mark A. Hall. "Correlation-based Feature Selection for Machine Learning". MA thesis. Hamilton, New Zeland: The University of Waikato, Apr. 1999.

[17] Inc Harris geospatial solutions. *Harris geospatial solutions*. 2018. URL: `http://www.harrisgeospatial.com/docs/CalculatingConfusionMatrices.html`.

[18] Richiardi J et al. "Classifying minimally disabled multiple sclerosis patients from resting state functional connectivity". In: *Neuroimage* (2012).

[19] Biao Jie et al. "Integration of Network Topological and Connectivity Properties for Neuroimaging Classification". In: *IEEE Trans Biomed Eng* (2014).

[20] Biao Jie et al. "Topological Graph Kernel on Multiple Thresholded Functional Connectivity Networks for Mild Cognitive Impairment Classification". In: *Wiley Periodicals, Inc.* (Sept. 2013).

[21] Onnela Jukka-Pekka et al. "Intensity and coherence of motifs in weighted complex networks". In: *American Physical Society* (2005).

[22] Prechelt Lutz. "Early Stopping-But When?" In: *Proceeding Neural Networks: Tricks of the Trade* (1996).

[23] Rubinov M and Sporns O. "Complex network measures of brain connectivity: Uses and interpretations." In: *Neuroimage* (2010).

[24] Belgiu Mariana and Csillik Ovidiu. "Sentinel-2 cropland mapping using pixel-based and object-based timeweighted dynamic time warping analysis". In: *Remote Sensing of Environment* (2018).

[25] Gonen Mehmet and Alpaydın Ethem. "Multiple Kernel Learning Algorithms". In: *Journal of Machine Learning Research* (2011).

[26] Tom Mitchell and Andre Elisseeff. *Machine learning*. McGraw Hill., 1997. ISBN: 0-07-042807-7.

[27] Kussul Nataliia et al. "Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data". In: *IEEE GEOSCIENCE AND REMOTE SENSING LETTERS* (2017).

[28] Shervashidze Nino et al. "Weisfeiler-Lehman Graph Kernels". In: *Journal of Machine Learning Research* (2011).

[29] Dalmau Oscar S, Alarcon Teresa E., and Oliva Francisco E. "Crop Classification in Satellite Images through Probabilistic Segmentation Based on Multiple Sources". In: *Sensors* (2017).

[30] Hao Pengyu et al. "Feature Selection of Time Series MODIS Data for Early Crop Classification Using Random Forest: A Case Study in Kansas, USA". In: *remote sensing* (2015).

[31] Haralick R.M., Shanmugam K., and I. Denstien. "Textural features for image classification". In: *IEEE Trans Syst Man Cybern* (1973).

[32] Weith Robert and Riggan Norman D. "OBJECT-BASED CLASSIFICATION VS. PIXEL-BASED CLASSIFICATION: COMPARITIVE IMPORTANCE OF MULTI-RESOLUTION IMAGERY". In: *Remote Sensing and Spatial Information Sciences* ().

[33] Skakun Sergii et al. "Efficiency Assessment of Multitemporal C-Band Radarsat-2 Intensity and Landsat-8 Surface Reflectance Satellite Imagery for Crop Classification in Ukraine". In: *IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING* (2016).

[34] Parikshit Sondhi. "Feature Construction Methods: A Survey". MA thesis. 201 North Goodwin Avenue Urbana,IL: Univeristy of Illinois at Urbana Champaign.

[35] Hatie Trevor, Tibshirani Robert, and Friedman Jerome. *An Introduction to Feature Extraction*. Vol. 207. Springer, 2008.

[36] Humboldt state university. *GPS 216 introduction to remote sensing*. 2015. URL: `http://gsp.humboldt.edu/olm_2016/Courses/GSP_216_Online/lesson6-2/metrics.html`.

[37] Mu Zhu. "Feature Extraction and Dimension Reduction with Applications to Classification and the Analysis of Co-occurrence Data". dissertation. Stanford University, 2001.