Sapienza
Università di Roma

DEPARTMENT OF INFORMATION ENGINEERING, ELECTRONICS AND TELECOMMUNICATIONS

Distributed Learning for Multiple Source Data

Ph.D. in Information and Communication Technology – XXX Cycle

Candidate
Rosa Altilio

Thesis Advisor
Prof. Massimo Panella

Thesis not yet defended

Distributed Learning for Multiple Source Data
Ph.D. thesis. Sapienza – University of Rome
ISBN: 000000000-0

.

Version: October 31, 2017
Author's email: rosa.altilio@uniroma1.it

Abstract

Distributed learning is the problem of inferring a function when data to be analyzed is distributed across a network of agents. Separate domains of application may largely impose different constraints on the solution, including low computational power at every location, limited underlying connectivity (e.g. no broadcasting capability) or transferability constraints related to the enormous bandwidth requirement. Thus, it is no longer possible to send data in a central node where traditionally learning algorithms are used, while new techniques able to model and exploit locally the information on big data are necessary.

Motivated by these observations, this thesis proposes new techniques able to efficiently overcome a fully centralized implementation, without requiring the presence of a coordinating node, while using only in-network communication. The focus is given on both supervised and unsupervised distributed learning procedures that, so far, have been addressed only in very specific settings only. For instance, some of them are not actually distributed because they just split the calculation between different subsystems, others call for the presence of a fusion center collecting at each iteration data from all the agents; some others are implementable only on specific network topologies such as fully connected graphs. In the first part of this thesis, these limits have been overcome by using spectral clustering, ensemble clustering or density-based approaches for realizing a pure distributed architecture where there is no hierarchy and all agents are peer. Each agent learns only from its own dataset, while the information about the others is unknown and obtained in a decentralized way through a process of communication and collaboration among the agents. Experimental results, and theoretical properties of convergence, prove the effectiveness of these proposals.

In the successive part of the thesis, the proposed contributions have been tested in several real-word distributed applications. Telemedicine and e-health applications are found to be one of the most prolific area to this end. Moreover, also the mapping of learning algorithms onto low-power hardware resources is found as an interesting area of applications in the distributed wireless networks context. Finally, a study on the generation and control of renewable energy sources is also analyzed.

Overall, the algorithms presented throughout the thesis cover a wide range of possible practical applications, and trace the path to many future extensions, either as scientific research or technological transfer results.

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Background Material

Chapter 1

Introduction

In recent years, new technologies have reshaped the ICT world with a massive volume of both structured and unstructured data that is so large to make difficult to processing it by using traditional technologies and software. Data generally comes from multiple sources, such as the collected results of health care researches, scientific experiments on real-time sensors, business sales records, financial values or information on a supply chain system.

All of the previous applications are concerned with an increasing amount of data that needs to be stored and processed with innovative techniques on a daily basis. It is referred to 'Big Data'. When treating with big data, we must be able to handle three kinds of characteristics, generally identified as the Three "V", whose expansion is illustrated in Fig.1.1, and whose description is the following:

- Volume: the acquired data is so large and complex that it can no longer be possible to save or analyze it using conventional data processing methods. In fact, in recent years, we have been able to deal with terabytes, petabytes, and exabytes of data acquired in a real-time or near-real time context;
- Variety: data could be heterogeneous, so we should be able to deal with both structured and unstructured data as well as text documents, email, video, audio, speech recordings, financial transaction and so on. Additionally, many sources of big data are relatively new, so it is necessary an ad-hoc treatment to handle them. Without considering that data can be highly inconsistent with periodic peaks, like something trending, daily, seasonal, and event-triggered peak data loads.
- Velocity: it should remain the same even if the volume of data increases. This characteristic is often even more important than the volume because the ability to achieve information in a faster way could bring a company in a superior condition compared to competitors. Data is acquired at unprecedented speed and must be dealt, when possible, in a timely manner; just think of RFID tags, sensors, smart metering, or all of the near real-time applications.

Recently, two new items have been added to describe Big Data's features:

- Volatility: in addition to the increasing velocity and volume of data, its volatility needs to be carefully considered. It refers to how long is data valid and how long should it be stored, by determining what point is data no longer relevant for the current analysis. In this way, the rapid retrieval of information when the cost and the complexity of storage are infeasible, is allowed.

- Value: the last feature of big data is maybe the most important one since is related to the information that can be extracted. Substantial value can be found from it, including understand customers better, optimizing processes or improving business performance. To this end, suited techniques must be able to gathering and turn out data into huge value for who is treated with it.

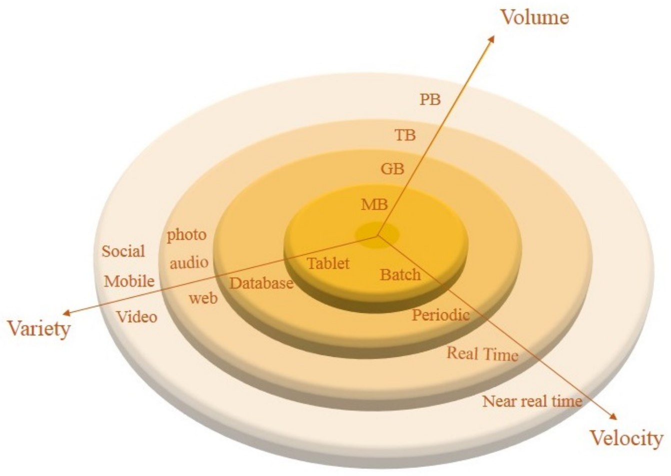


Figure 1.1. Big Data: Expanding on 3 fronts at an increasing rate

Traditional techniques are not well suited to capture the full big data's value, allowing only a small percentage of data to be actually analyzed. Machine learning approaches are promising methods for exploiting the opportunities hidden in big data, thriving with growing dataset. In particular, a lot of useful features should be extracted to learn the underlying relationship among results. In biomedical context, for example, it could be interesting extract a subcategory of pathologies from records acquired on patients. While, in biological context, it could be useful find out the grouping of data among several biological species. Again, in the weather forecast could be helpful to know what are the elements of both the atmospheric and oceanic pressure on the weather.

Thus, extracting knowledge from big data poses significant research and challenges. However, the problem could be complicated when data is spread across distinct logical or physical locations, and could not be sent to a central node for reason of security, processing or privacy. In these contexts, the range of potential correlations and relationships could makes complex the test of all hypothesis and the successive extraction of information buried in data.

A first attempt to deal with this scenario consists in scaling the data to be processed horizontally, allowing the processing of data in parallel. However, the synchronization of data across nodes may not be in real time.

For all of the previous reasons, a purely distributed scenario, when the agents work on a local dataset that could be shared only with the neighbors, concerns the main topic of this thesis. In a pure distributed architecture there is no hierarchy and all agents are peers. In this case, the data is partitioned among the agents (equally or proportionally to the calculation power) and each agent learns only from its own dataset. The idea is to reach a solution similar to the centralized one (where all the learning takes place in a central node that collects all the data) only through a process of communication and collaboration among the local agents.

1.1 Structure of the Thesis

The thesis is organized in 3 big sections. A schematic categorization of each one is provided as follows:

Part I Introduces the required background material:

Chapter 2 Offers a schematic overview of the fully centralized machine learning techniques. Specifically, regression, classification and more in general supervised and unsupervised learning approaches are presented.

Chapter 3 Provides a formal definition and formulation of the distributed learning scenario. Additionally, a comprehensive overview of state-of-the-art algorithms regarding the distributed learning is presented. It combines works from multiple research fields and gives a unified discussion on the main drawbacks and advantages of the existing approaches.

Part II Introduces the several proposed approaches to distributed supervised and unsupervised learning problems.

Chapter 4 Introduces the Validated Distributed Ensemble Clustering. It extends the traditional cluster ensemble techniques overcoming the requirement that each agent has to work with the same dataset.

Chapter 5 In this chapter, it is introduced a fully distributed procedure for performing spectral clustering over networks of computing agents. In particular, the equivalent problem of completing the matrix containing the pairwise distances among all datapoints is analyzed and solved by using a distributed gradient procedure, interleaving gradient descent steps with point-to-point diffusion of information.

Chapter 6 Introduces a distributed on-line learning for random weight fuzzy neural networks. Data is allowed to arrive continuously, one-by-one or chunk-by-chunk, at every node. In this approach, the concept of epoch disappears, allowing the algorithm to avoid retraining whenever new data is received.

Chapter 7 Introduces an unsupervised algorithm for solving density based approaches. A non-convex distributed optimization in multi-agent networks with time-varying (not symmetric) connectivity is applied to the well-known Expectation Maximization approach. The local solutions are found through which a sequence of strongly convex, decoupled, optimization subproblems, while the agreement is reached by a consensus step.

Part III In this section, different possible applications of the distributed scenarios have been presented. A centralized analysis is firstly realized in order to extend it, and investigates what happens when these approaches will be placed in a distributed setting.

Chapter 8 In this chapter, it has been considered the application of data mining methods in medical contexts, wherein the data to be analyzed is distributed among multiple clinical parties. It extends the idea of distributed learning, in the perspective of a tele-rehabilitation context. Machine learning approaches are firstly used to extract important features, helping clinicians in evaluating the impairment's degree of a patient. Successively, an extension of the algorithm presented in Chapter 6, where two techniques of privacy constraints are added to deal with this sensible data, is described.

Chapter 9 In order to realize a complete tool able to work in a distributed context, like the one of tele-rehabilitation, two low-cost devices are tested and assessed in capturing all of the parameters used in the previous chapter.

Chapter 10 Introduces another application of distributed learning that is the one of sensors networks and low-power devices. A finite precision implementation of a Random Vector Functional Link is applied to distributed signal processing scenario, where a low computational power of a simple and cheap hardware is often required.

Chapter 11 Is related to the ability to forecast the power produced by renewable energy plants in the context of a highly connected network of agents. In this chapter, a new embedding approach based on neural and fuzzy neural networks is presented. The idea is to test the capability of model identification and the accuracy of time series prediction, by which the results could be improved sharing the data from different cabins of the same plants, thus achieving a multivariate space-time prediction by using multiple sources of data.

Chapter 12 Finally, this chapter summarizes the main contributions of this thesis, along with the further developments.

Appendix A gives a general overview of the unsupervised validity indexes, focusing the attention on the internal validity measures and the external ones used in the overall work in order to judge the validity of the approaches.

Appendix B introduces some statistical tests that have been used in the thesis to perform inference on random variables.

Appendix C introduces several classification algorithms used to solve specific feature selection problem.

Part of this thesis is adapted from material published (or currently under review) in several journals and conferences.

1.2 Notation

Throughout the thesis, vectors are denoted by boldface lowercase letters, e.g. $\mathbf{a}$, while matrices are denoted by boldface uppercase letters, e.g. $\mathbf{A}$. The notation $\mathbf{a}_i$ denotes the (i)th entry of the vector $\mathbf{a}$, which is assumed to be column one, with $\mathbf{a}^T$ denoting the transpose of $\mathbf{a}$. Similarly for the matrix, the notation $\mathbf{A}_{ij}$ denotes the (i, j)th entry of matrix $\mathbf{A}$. The operator $vect(\mathbf{a})$ is used for the column-wise vectorization of a matrix, while the operator $diag(\mathbf{a})$ deals with diagonal matrices $\mathbf{A}$, such that $\mathbf{A}_{ii} = a_i$. $\mathbf{0}$ indicates the zero vector or matrix, while $\mathbf{1}$ the all-ones vector or matrix.

The L_p-norm of a generic vector $\mathbf{a}$ is indicated with $\|a\|_p$:

$$\|\mathbf{v}\| = (\sum_{i=1}^{n} |v_i|^p)^{\frac{1}{p}} \tag{1.1}$$

The standard Euclidean one, for $p = 2$, is indicated $\|a\|$, while for $p = 1$ we have $\|a\|_1 = \sum_i a_i$.

In some cases, vectors will depend to a time-instant. It is indicated as $a[n]$ both for time-varying signals (in which case n refers to a time-instant) and for elements in an iterative procedure (in which case n is the iteration's index).

$A \geq 0$ denotes a positive semi-definite (PSD) matrix, i.e. a matrix for which $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$ for any vector $\mathbf{x}$ of suitable dimensionality.

A function is generally indicated with an italic letter f and is defined using the notation $f : \mathbb{R}^m \to \mathbb{R}$. Let $\mathbf{x} \in \mathbb{R}^m$ a free variable, then the partial derivative of f with respect to x_i is denoted by $\frac{\partial f}{\partial x_i}$. The gradient of f, evaluated at q, is denoted as $\nabla_q f(\mathbf{q})$ and is identified as a column vector as follows:

$$\nabla_{\mathbf{q}} f(\mathbf{q}) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1}(\mathbf{q}) \\ \vdots \\ \dfrac{\partial f}{\partial x_n}(\mathbf{q}) \end{bmatrix} \tag{1.2}$$

Sometimes the dependency on $\mathbf{q}$ is omitted when it is obvious from the context, consequently, $\nabla_{\mathbf{q}} f(\mathbf{q})$ could be indicated as $\nabla f(\mathbf{q})$ or ∇f.

The set is usually written using upper-case letters (e.g $\mathcal{D}$). The usual sets of numbers are denoted using the blackboard font, like $\mathbb{R}$ for real numbers or $\mathbb{Z}$ for the integers. The explicit definition of a set is denoted by using curly brackets (like $\mathcal{D} = \{2^{-7}, 2^{-6}, \cdots, 2^{6}, 2^{7}\}$. Real intervals are denoted using brackets, let $\mathbf{x}$ as a real variable, then the set of values such that $aleqx \leq b$ is indicated as $[a, b]$.

1.3 Publications Related to the Thesis

The main publications written within the research work associated with this PhD thesis are listed in the following:

- R. Altilio, L. Liparulo, M. Panella, M. Paoloni and A. Proietti, ”Multimedia and Gaming Technologies for Telerehabilitation of Motor Disabilities”, *IEEE Technology and Society Magazine*, Vol. 34, No.4, pp. 23-30, ISSN: 0278-0097, DOI: 10.1109/MTS.2015.2494279, IEEE, USA, December 2015.
- R. Altilio, M. Paoloni and M. Panella, “‘Selection of clinical features for pattern recognition applied to gait analysis”, *Medical & Biological Engineering & Computing*, Vol. 55, No. 4, pp. 685-695, ISSN: 0140-0118, DOI: 10.1007/s11517-016-1546-1, Springer Berlin Heidelberg, Germany, April 2017 (online publication: 19 July 2016).
- A. Rosato, R. Altilio, R. Araneo and M. Panella, “Prediction in Photovoltaic Power by Neural Networks”, *Energies*, ISSN: 1996-1073, MDPI, Switzerland, in printing.
- R. Altilio and M. Panella, “A smartphone-based Application Using Machine Learning for Gesture Recognition”, *IEEE Consumer Electronics Magazine Editorial Office*, accepted in minor revision.
- R. Altilio, A. Rossetti, Q. Fang, X.fu and M. Panella, “A smartphone gait analysis for post-stroke lower limb rehabilitation”. *Journal of Biomedical and Health Informatics*, submitted in 2017.
- A.Rosato, R. Altilio and M. Panella, “An unsupervised learning algorithm for distributed environment”, Soft Computing, submitted in 2017.
- A. Rosato, R. Altilio and M. Panella, “Recent Advances on Distributed Unsupervised Learning”, in *Advances in Neural Networks: Computational Intelligence for ICT, Smart Innovation, Systems and Technologies (WIRN 2015)*, Vol. 54, pp. 77-86, ISBN: 978-3-319-33746-3, ISSN: 2190-3018, DOI: 10.1007/978-3-319-33747-0_ 8, *Springer International Publishing*, Switzerland, June 2016.
- S. Scardapane, R. Altilio, V. Ciccarelli, A. Uncini and M. Panella, “Privacy-preserving data mining for distributed medical scenarios”, in *Multidisciplinary Approaches to Neural Computing (WIRN 2016), Smart Innovation, Systems and Technologies, Springer International Publishing*, Switzerland, in printing.
- A. Rosato, R. Altilio, R. Araneo and M. Panella, “Embedding of Time Series for the Prediction in Photovoltaic Power Plants”, *Proc. of IEEE International Conference on Environment and Electrical Engineering (IEEE EEEIC 2016)*, pp. 1-4, ISBN: 978-1-5090-2320-2, 978-1-5090-2319-6, DOI: 10.1109/EEEIC.2016.7555872, IEEE, Florence, Italy, 7-10 June 2016.

- S. Scardapane, R. Altilio, M. Panella and A. Uncini, "Distributed Spectral Clustering based on Euclidean Distance Matrix Completion", *Proc. of International Joint Conference on Neural Networks (IJCNN 2016)*, pp. 3093-3100, ISBN: 978-1-5090-0620-5, ISSN: 2161-4407,DOI: 10.1109/IJCNN.2016.7727593, IEEE, Vancouver, Canada, 24-29 July 2016.

- R. Altilio, L. Liparulo, A. Proietti, M. Paoloni and M. Panella, "A Genetic Algorithm for Feature Selection in Gait Analysis", *Proc. of IEEE Congress on Evolutionary Computation (IEEE CEC 2016)*, pp. 4584-4591, ISBN: 978-1-5090-0623-6, DOI: 10.1109/CEC.2016.7744374, IEEE, Vancouver, Canada, 24-29 July 2016.

- A. Rosato, R. Altilio, R. Araneo and M. Panella, "Takagi-Sugeno Fuzzy Systems Applied to Voltage Prediction of Photovoltaic Plants", *Proc. of IEEE International Conference on Environment and Electrical Engineering (IEEE EEEIC 2017)*, pp. 1-6, ISBN: 9781538639177, IEEE, Milan, Italy, 6-9 June 2017.

- R. Fierimonte, R. Altilio and M. Panella, "Distributed On-line Learning for Random-Weight Fuzzy Neural Networks", *Proc. of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2017)*, pp. 1-6, IEEE, Naples, Italy, 9-12 July 2017.

- R. Altilio, A. Rosato and M. Panella, "A New Learning Approach for Takagi-Sugeno Fuzzy Systems Applied to Time Prediction", *Proc. of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2017)*, pp. 1- 6, IEEE, Naples, Italy, 9-12 July 2017.

- A. Rosato, R. Altilio and M. Panella, "Finite Precision Implementation of Random Vector Functional-Link Networks", *Proc. of International Conference on Digital Signal Processing (DSP 2017)*, pp. 1-5, IEEE, London, UK, 23-25 august 2017, in printing.

- R. Altilio, A. Rosato and M. Panella, "A Nonuniform Quantizer for Hardware Implementation of Neural Networks", *Proc. of European Conference on Circuit Theory and Design (ECCTD 2017)*, pp. 1-4, IEEE, Catania, Italy, 4-6 September 2017, in printing.

Chapter 2

Centralized Machine Learning

The problem of extracting information and the automatic discovery of regularities in data with computer algorithms is increasingly widespread in a lot of scientific endeavors:

- Computational finance for credit scoring and algorithmic trading;
- Image processing and computer vision for face recognition, motion detection, object detection;
- Computational biology for tumor detection, drug discovery and DNA sequencing;
- Energy production for price and load forecasting;
- Medical Context for classifying patient among diseased and not diseased.

In all of these cases, the underlying question is how can learn from such data. Machine learning approaches are promising methods able to learn and make predictions on data, by exploiting the functioning by which animals and humans learn from the experience. Given L number of patterns, a set of input data $[\mathbf{x}_1, \dots, \mathbf{x}_L] \in \mathbb{R}^n$, and a set of output data $[\mathbf{y}_1, \dots, \mathbf{y}_L] \in \mathbb{R}$, the aim is to model the underlying relationship among them in order to make better decisions and predictions. The algorithms adaptively improve the performances and, particularly, as the number of available samples increases also the learning capability increases. Machine learning approaches are essentially based on four different kinds of techniques that are represented in Fig. 2.1, and summarized below:

- Supervised Learning: the corresponding output and label to each input data are known beforehand. The task is to model the relationship from the input and the output.
- Unsupervised Learning: the input data consists of a set of vectors $\mathbf{x}$ without any corresponding target value. The aim is to find the structure of the input.
- Semisupervised Learning: there is an incomplete input vector, with some (often many) of the target output missing. These problems sit in between supervised and unsupervised learning, so a mixed of techniques suitable to solve both approaches can be used.

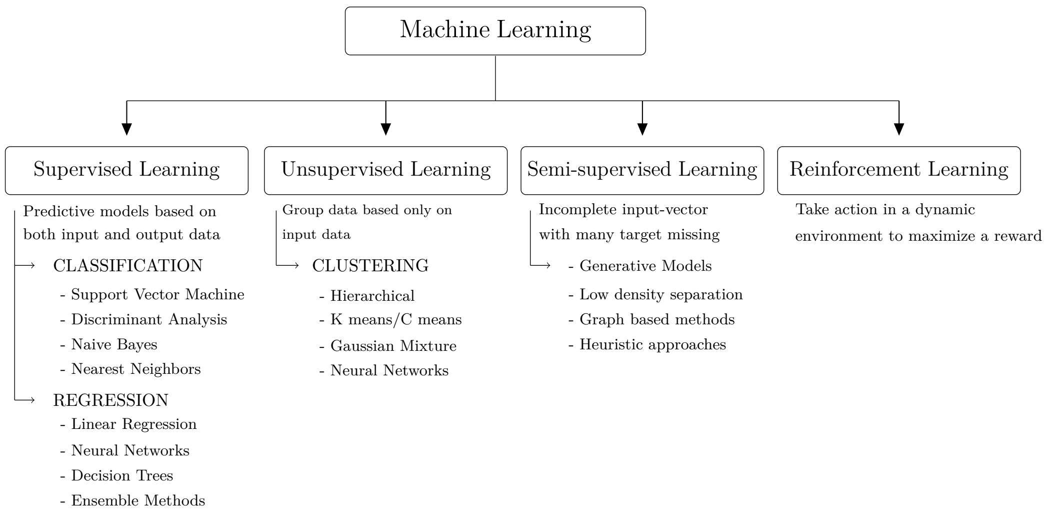


Figure 2.1. Machine Learning techniques

- Reinforcement Learning: tries to find out the suitable actions that have to been taken in a given situation in order to maximize a reward. Typically, there is a sequence of states and actions where the learning algorithm has to interact with the environment. It tries to find a trade-off between exploration, where the system tries out new kinds of actions to see how effective they are, and exploitation, in which the system makes use of actions that are known to bring a high reward.

Another categorization that can be performed on machine learning algorithms involves the desired output of a system:

- Classification: where the inputs are divided among two or more classes and the algorithm must be able to assign each input to the correct one. If there are two classes, we talk of binary classification whereas if they are greater, we talk about a multiclass classification. The output data takes only a discrete amount of values, and represents a category, such as "read" or "blue", or "disease" and "no disease". This is typically tackled in a supervised way.

- Regression: where the output is a real value, and the aim is to predict a quantitative information instead of a qualitative one. The elements of the output are infinite. This is a supervised problem.

- Clustering: this is an unsupervised problem where the groups are not known beforehand, and the aim is to divide a set of inputs in several clusters.

- Density estimation: is based on finding the distribution of the input in some spaces.

In the following sections, some techniques related to the supervised and unsupervised approaches are presented.

2.1 Supervised Learning

The aim of the supervised learning is to find out a model able to make a prediction even when uncertainty and fortuity characterize data. Let us suppose to have a d-dimensional vector of real numbers $x \in \mathbb{R}^{nxd}$, where each component represents a feature, while each row represents a pattern, and a single scalar output $\mathcal{Y} \subseteq \mathbb{R}$ (that can be easily considered and extended to the multidimensional case). The goal of the supervised learning problem, in the regression case, is to approximate the mapping function between an input vector x and the corresponding output Y.

$$Y = f(x) \tag{2.1}$$

The derivation of the model in the case of binary classification proceeds identically, with the difference that the output is not a real value, but a discrete one: $y \in \{-1, +1\}$. In this case, the actual class of the pattern can be evaluated as:

$$\text{Class of } \mathbf{x} = \text{sgn} f(\mathbf{x}) \tag{2.2}$$

Similarly, in the case of of multiclass classification, the output can take values in a set of M different labels $\{1, 2, \ldots, M\}$ where M is the number of classes. In this case, a common encoding association consists in assign to each pattern $\mathbf{x}_i$, a single output vector $\mathbf{y}_i$ of M bits: if $y_{ij} = 1$ and $y_{ij} = 0, \forall k \neq j$, then the corresponding pattern is of the class j. Also in this case, a common way to proceed consists to assume $y \subseteq \mathbb{R}^M$, $f(\mathbf{x}) \in \mathbb{R}^M$ and retrieval the actual class as:

$$\text{Class of } \mathbf{x} = \underset{j=1,\ldots,M}{\arg\max} f_j(\mathbf{x}) \tag{2.3}$$

In all cases, the process of inferring a function $f(\cdot)$ from a dataset is called *training*. Supposing that the unknown function belongs to a functional space $\mathcal{H}$, a loss function $l(y, f(\mathbf{x})) : \mathcal{Y}x\mathcal{Y} \to \mathbb{R}^+$ could be introduced to quantify the error incurred in estimating $f(\mathbf{x})$ instead of the true y. Thus, the supervised learning problem can be defined as:

Definition 2.1. Given an hypothesis space $\mathcal{H}$ and a loss function $l(\cdot, \cdot)$, the solution of a supervised learning problem consists in finding the function $f(\cdot)$ able to minimize the expected risk functional:

$$I_{exp}[f] = \int l(y, f(\mathbf{x})dp(\mathbf{x}, y), f \in \mathcal{H} \tag{2.4}$$

The probability distribution is unknown, so the solution to the problem (2.4) can be given by an approximation realized by a generic dataset $\mathcal{D}$, which is called empirical risk function:

$$I_{emp}[f] = \sum_{i=1}^{N} l(y_i, f(\mathbf{x}_i)) \tag{2.5}$$

However, minimizing (2.5) instead of (2.4) could lead the risk of overfitting, making the function unable of generalizing efficiently on new data presented to the network. A common solution consists in the inclusion of a regularizing term able to take into account some features of the unknown function like smoothness, sparsity and so on:

Definition 2.2. Given a dataset $\mathcal{D}$, an hypothesis space $\mathcal{H}$, a loss function $l(\cdot;\cdot)$, a regularization functional $\phi[f] : \mathcal{H} \to \mathbb{R}$, and a scalar coefficient $\lambda > 0$, the regularized SL problem is defined as the minimization of the following functional:

$$I_{reg}[f] = \sum_{i=1}^{N} l(y_i, f(\mathbf{x}_i)) + \lambda\phi[f] \tag{2.6}$$

The problem in (2.6) can be analyzed from a wide variety of viewpoints, including statistical learning theory or Bayes's theory.

Before this, however, few words on the choice of the outspace $\mathbf{y}$ need to be said. As stated before, the model can be used to solve the problem of both classification and regression. As far as the classification approach is involved, the most important techniques used to solve it will be described as follows:

Support Vector Machines (SVM) [1]: is a model used for classification purpose able to assign new examples to one category or others, making use of a non-probabilistic binary linear classifier. It is based on a representation of the examples as points in the space, and it tries to find out the hyperplane able to maximize margin among the patterns, as well as minimize the mistakes on the training set. More precisely, it is based on the distance from an example x_i to the separator, the ones closest to them are called support vectors. Therefore, the aim of the SVM is to maximize the margin between support vectors. This can be formulated as an optimization problem, which is particularly hard to solve in its original formulation, but that can be rewritten in a convenient form by using its dual form. The main advantage of this kind of system is that it has the possibility to obtain non-linear boundaries between samples. In particular, the idea is to use particular structures, called Kernel, which allow to transform the data from the input space to a higher dimensional one that is called feature space. The resulting advantage is that the non linear operation of the input space becomes linear in the feature one.

Neural Networks [2, 3] is a model inspired by the biological neural system. It is an adaptive method, where the learning is obtained through examples. Generally, it is based on a collection of single units, called neurons, organized in several layers (an input, an output, and several hidden ones) and connected between synapses. In each layer, the nodes perform a transformation of the data received from the previous one and send an aggregate result in the successive one. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables. The main idea is to perform a learning similar to the one obtained by the human brain. Often, the backpropagation procedure is used to adjust the parameters of the network by propagating the error from the output to the hidden and input layers.

Naive Bayes classifier [4, 5, 6], based on the Bayes' theorem, provides a way of calculating posterior probabilities by assigning a new observation to the most probable class. It is easy and fast in predicting the class of a data set. Furthermore, it performs well in the multi class prediction. However, it requires the assumption of independent predictors, which is often difficult to find in real context problems.

Decision tree [7] is based on a diagram to predict responses on data. It starts from the root node and comes down to the leaf one, which contains the response. In particular, it gives responses that are nominal, such as 'true' or 'false'. Each step, during the prediction, involves checking the value of one predictor (variable).

Discriminant analysis [8] is a classification method assuming that data is generated according to different Gaussian mixtures. At first, the classifier is trained by estimating the parameters of each Gaussian distribution. Successively, the new data is assigned to the class able to produce the smallest misclassification cost.

k Nearest neighbors [9] is a non parametric method where the output is a membership class. An object is classified by using the majority vote of its neighbors. Specifically, a data is assigned to the most common class among its k nearest neighbors.

As far as the regression algorithms are concerned, the most common approaches are the following ones:

Linear Regression [10] is a statistical model used to describe the relationship between output values and one or more predictor variables. In particular, it supposes a linear model among the response and the input. It can be one of a different kind of linear regression models: simple (when the model has only one predictor), multiple (when the model has multiple predictors) and multivariate (when the model has multiple response variables).

Non linear regression [11] is a statistical technique that assumes a non linear relationship between the input data and the output response. It is generally not parametric, and the output is modeled as a combination of non linear parameters and one or more independent variables.

Decision trees [7] are based on a independent variable where the target values are not associate to classes. Data is divided into several split points, where at each one of them the sum of squared error between the predicted value and the actual one is updated.

Neural network [2, 3] is similar to its corresponding classification version, the only difference being on the output evaluation. In this case, the target becomes continuous and not discrete.

2.2 Unsupervised learning

In this section, the focus is changed in all the situations where the targets of the output values are unknown. The aim is to discover groups of similar examples, in order to maximize the intraclass similarity and minimize the interclass similarity. In this way, observations within the same cluster are similar according to some predesignated criterion or criteria, while observations drawn from different clusters are dissimilar. The aim is to learn and establish baseline behavioral profiles for various patterns, as well as determine the distribution of data within the input space or know the density estimation of data. Since the difficulties that the unsupervised context carries with it, different assumptions about the structure of the data are mandatory. In particular, a similarity metric must be defined and the internal compactness and separation among cluster have to be evaluated to judge the results:

Definition 2.3. Supposing to have an input dataset $S = (x_1, \ldots, x_n)$ with samples i.i.d with respect to $p(\mathbf{x})$, then an aim of the unsupervised learning could be minimize the empirical (reconstruction) error:

$$I_S[C] = \frac{1}{n} \sum_{i=1}^{n} d^2(x_i, C); \tag{2.7}$$

where C is a subset of X and $d^2(x_i, C) = \min_{v \in C} \|x - v\|^2$ is the square distance of a point x to the set C.

The error measure above depends on the distribution, we can alternatively consider regularized optimization functions:

$$\min_{C \in \mathcal{H}} I_S[C] + \lambda R(C) \tag{2.8}$$

where $\mathcal{H}C|C \subset X$ is essentially an unconstrained hypotheses space and $R(C)$ a regularization term that penalizes complex/large sets.

Many techniques have been used to approach this kind of problem:

Hierarchical clustering [12]: groups data by creating a cluster tree or a dendrogram. The agglomerative methods start with n clusters containing one object, and aggregating the most similar pair (C_i, C_j) in the next level to form a new cluster. The procedure is stopped when the result is the more appropriate for the scope of the application. A lot of similarity measurements can be used to evaluate the clustering results and decide when the procedure has to be stopped. Conversely, the divisive approach follows a bottom-up approach, where all of the datapoints originally belong to the same cluster that is recursively split in the successive steps. The procedure continues until each cluster is formed by exactly one pattern or until a stopping condition, on the particular structure of the model, is reached.

Partitioning Methods [13] considering n objects in the network, the partitioning method realizes k partition of the data. Each partition represents a cluster, which must contain at least one object and each object must belong to exactly one group. All of these approaches start with an initial randomly partition, which is used to iteratively improve the partitioning by moving points from one cluster to another. A usual example of this approach is the K-Means algorithm, which tries to groups data into k distinct clusters. The main idea is to define a priori the number of centroids, one for each cluster, and then try to assign data to each cluster in an iterative procedure by minimizing the distance among the patterns. Another partitioning method is the spectral clustering approach. For a given dataset, it finds a set of data on the basis of a spectral analysis on the similarity graph. It uses the spectrum (eigenvalues) of the similarity matrix of data to perform dimensionality reduction before clustering in fewer dimensions. The clustering problem is defined in terms of a complete graph G, where the vertices represent the data point, whereas the edges represent the similarity between them. The main difficulty consists in evaluating the similarity matrix at the beginning of the procedure.

Gaussian Mixture Models [14] is a probabilistic model based on the Expectation maximization algorithm where is assumed that data arrives from an environment that can be described by a probability density function (PDF) of one or more multivariate Gaussian distribution components. Each multivariate component is defined by its mean and covariance, while the number of components, for a given object, is fixed. The mixture parameters are evaluated in an iterative way by performing an estimate of the maximum likelihood through an expectation and a maximization step.

Self-organizing maps [15] is a type of artificial network used both for clustering data and reducing its dimensionality. The input are discretized by using a similarity map able to assign the same number of instances to each class.

Hidden Markov model [16] performs inference on data by assuming it to be extracted from a Markov process with unobserved states, and by using a sequence of states from the observed data.

For a particular dataset, it may happens that several partitions are obtained when different clustering algorithms or different initialization parameters for the same algorithm, or simply different samples of the data are used. In these situations, finding out a common structure among all the results, the so-called *Ensemble Methods* [17], is the best way of proceeding. They can be based on median partition to find the most similar result among all of the others in the ensemble. Or they can be based on a weighted graph, which represents the multiple clustering results from the ensemble where the minimum cut is used to find an agreement.

2.3 Structure of the learning process

Machine learning techniques aim at finding out the underlying relationship among data in order to give information on the output parameters. A schematic depiction of this process is given in Fig.2.2. The core of the approach is what happens in the central box. In particular, each learning algorithm, to model the relationship among the input-output data, follows a succession of steps showed in Fig. 2.3 and summarized as follows:

Data acquisition: the data acquisition is the first step involved in the model. The real data is often incomplete, noisy, or inconsistent for that a preprocessing step performed on raw data, is necessary to make it suitable and easily to threat. Sampling, to select a representative subset from a large population of data, denoising,

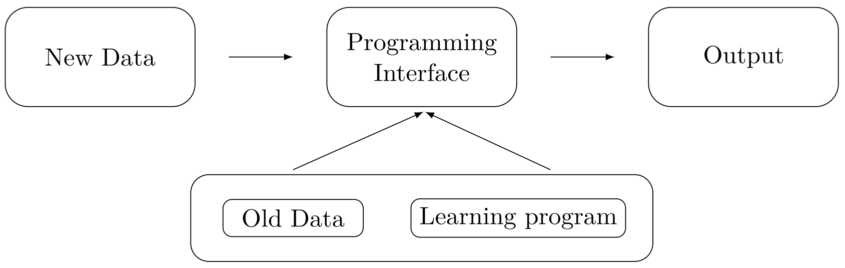


Figure 2.2. Machine Learning model

to remove noise from data, normalization, to scale attribute values to fall within a specified range are only a small subset of functions that can be applied at the beginning of the learning procedure.

Feature selection: especially in those dataset composed by many attributes, it may happens that groups of features are correlated. During the learning phase, their presence can affect the accuracy of the model. That is why a dimensionality reduction becomes necessary to remove all of the redundant information, and represents the dataset only in terms of the relevant elements:

- Feature selection: has the aim of finding the most significant predictors by selecting a subset of the original ones. PCA, ICA, Matrix Features Factorization is a list of techniques able to reduce the most discriminative components and, thus, allow a representation of the dataset with a small subset of values.

- Features extraction: involves the transformation of the features into a lower dimensional space. Unlike feature selection, which selects and retains the most significant attributes, feature extraction actually transforms the attributes by performing, for example, a linear combinations of the original ones. In this way, the output becomes smaller and richer but the process becomes not reversible because some information is lost during the dimensionality reduction.

Model Selection is the task of selecting a statistical model from a set of candidate ones. The aim is to perform the minimum error when the relationship among the data has to be modeled. For the supervised context, the aim is to maximize the generalization capability when new data is presented to the model, while for the unsupervised one, the aim is to create a compact and well-separated cluster of points.

Training: for the supervised context, the original dataset is normally divided into two subsets: the training data and the test data. In general, the data is presented as a “gold standard”, and the model is trained by presenting the input with the expected output. Successively, the learned model is tested on new data, which is the one of the test set. In the unsupervised context, the algorithms proceed by recursively adapt some statistical parameters related to the properties of the cluster that has to be determined. Generally, it depends on the distribution from which data has been extracted.

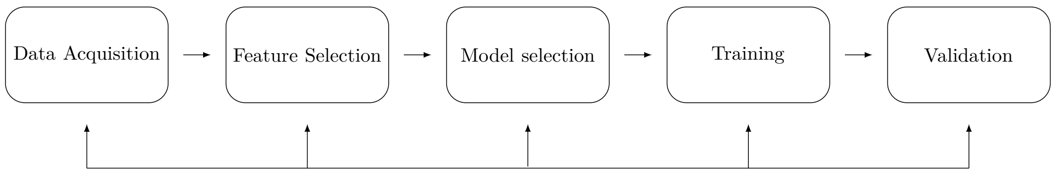


Figure 2.3. Learning process

Validation: in the supervised context, the algorithm calculates the output on the basis of the learned parameters. The validation is used after training to generalize parameters and avoid problems of overfitting or underfitting. Whereas, in the unsupervised context, the validation is used to test the performances of the trained model by using some cluster validation indexes able to minimize the interclass similarity and maximize the intraclass one.

2.3.1 Supervised validation techniques

The last step is the more important one concerning the learning capability of the model. Hence, it is explained in more details how it works for both the supervised and unsupervised approach.

In the supervised learning context, the aim is to maximize the generalization capability of the model by analyzing what happen when new data is presented to the structure. If the new values are well predicted then the algorithm was trained successfully, otherwise problems of overfitting or underfitting may have been occurred. The first one occurs when details and noise of the data are learned with the relevant information, producing accurate prediction on training data, but losing generalization performance on new ones. Conversely, the underfitting occurs when the model can neither describe the training data nor the new ones. If an algorithm is well trained, during the learning phase there is a decreasing value of both the training and the testing error. However, if the learning phase continues for too long time, the model starts to overfitting the training data, producing a low error on the training data but making bigger the one on the test set. Ideally, the correct point for stopping the learning phase is the instant just before the one in which the error on the test dataset starts to increase. In that point, the model has good performance on both the training test and the test one. In Fig. 2.4 is presented a simple example where it can be visualized the phenomena described above. In the top three diagrams, we have data and models. From the left-hand side to the right-hand side, the models have been trained longer and longer increasing theirs complexity. As it can be seen from the graph on the bottom of the figure, the training error gets better as the complexity increases, however the error on the test set has an opposite behavior. In fact, the generalization capabilities on new data start decreasing beyond a particular point.

So the question is: how does the algorithm has to be trained to avoid problems on the above-mentioned problems? The answer is not so obvious and can be given by adapting it to each particular problem that has to be solved. As it has been introduced before, a regularization term will help us in avoiding overfitting, while

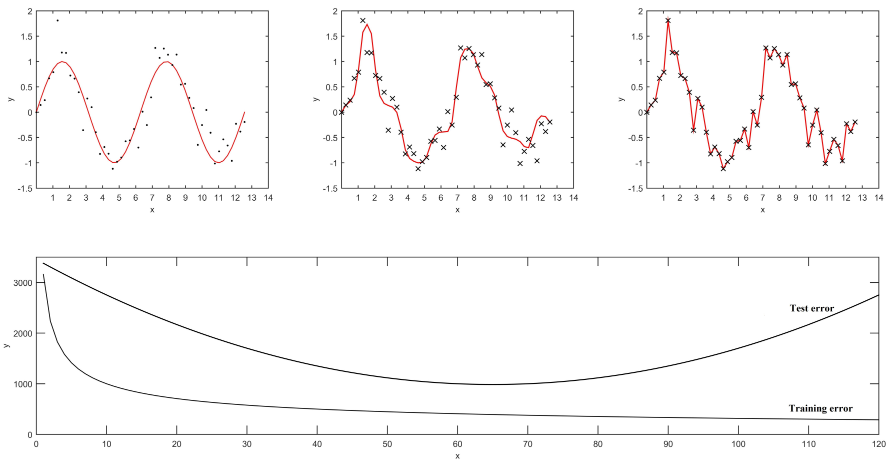


Figure 2.4. Overfitting and underfitting of a trained model

other techniques can be used to tune the regularization term. One of the most simple techniques, able to assess how the results of a statistical analysis will generalize to independent data, is the holdout method. Data is randomly assigned to two sets $d0$ and $d1$, usually called training and test set. In particular, a reasonable percentage of data to be assigned to one of the two set is 70% for training and the remaining 30% for the test. Then the algorithm is trained on $d0$, and the realized function approximator is asked to predict the output values for the data in $d1$. The results are evaluated in terms of mean absolute test set error. However, the evaluation may heavily depends on which data points ends up in the training set, and which ends up in the test set. Hence, the holdout method could be generalized to a more complex validation technique that is called k-fold cross validation.

It is based on an additional dataset that is generally called validation set. Generally, the k-fold cross validation allows to train and test the model k-times on different subsets of training data, and it realizes an estimate of the performance on unseen data. In particular, the original data is randomly partitioned into k equal sized subsamples, one of which is used as validation test to test model, whereas the others $k-1$ are used to train data. The process is repeated, $k$times, for all the validation set by using it exactly once for test the model. At the end of the procedure, results are averaged to produce a single estimation. In this way, the global dataset is used for both training and validation, while each observation is used for validation exactly once. The advantage of this approach is that you can independently choose how large each test set is and how many trails you average over. Leave-one-out cross validation is a particular case of k-fold cross validation where k is equal to n (the overall number of patterns). In this way, the model is trained n separate times. In other words, it is trained on all data except for one point on which the prediction is made. This approach confirms a good evaluation, but it can seem very expensive to compute. Fortunately, it takes no more time than computing the residual error, and it is a much better way to evaluate models.

2.3.2 Unsupervised validity techniques

For the unsupervised context, the validation procedure is a little different because there are no labels able to give information about the generalization performance of the algorithm. The way of validating the model depends on which class of unsupervised algorithm we are referring to.

For example, dimensionality reduction techniques are generally validated by computing the reconstruction error. In these cases, the validation can be performed by using similar approaches based on k-fold cross validation or holdout test set.

The density estimation approaches are difficult to evaluate, but there is a wide range of techniques used for tuning the parameters [18]. The main problem in these contexts is that there is not a general formula for the optimal smoothing parameters. It can be difficult to determine which components of X are relevant and which not. Often, cross validation overcomes these difficulties, and it automatically determines which components we have to include in the analysis.

As for clustering algorithms, one of the most important issues is the evaluation of results in order to find the partition that best fits the underlying data. The aim is to discover significant groups in a data set by searching for clusters whose elements are similar to each other, and well separated from the elements of others clusters. The process of evaluating the results of a clustering algorithm is the validity assessment, and generally two measurement are used to this end:

- Compactness: the members of each cluster should be as close to each other as possible.
- Separation: the clusters should be widely spaced. There are three common approaches usually used to measure the distance between two different clusters:
  - Single linkage: it measures the distance between the closest members of the clusters. It is based on statistical tests, and the major drawback is the high computational cost. Generally it is used to measure how much a dataset confirms an a priori specified scheme.
  - Complete linkage: it measures the distance between the most distant members.
  - Comparison of centroids: it measures the distance between the centers of the clusters. It tries to find the best clustering algorithm under certain assumptions and parameters.

These parameters are used to investigate the cluster validity, measured in terms of:

- External criteria: the results of a clustering algorithm are based on a pre-specified structure, which is imposed on a data set, and that reflects the intuition about the clustering itself. Precisely, the idea is to verify whether data satisfies a null hypothesis about their random structure. Statistical approaches, such as Monte Carlo techniques, are generally used to check this assumption.
- Internal criteria: the results are evaluated in terms of quantities that involve data set's vectors. These techniques are generally used with hierarchical clustering or single clustering schemes.

- Relative criteria: the basic idea lies in the evaluation of a clustering structure compared to other clustering schemes. Based on statistical testing, theirs major drawback is the high computational demand. There can be different cases for their application, described as follows:
  - The number of clusters (k) is not an algorithm's parameter. In this case, the algorithm is run for a wide range of parameters, choosing the largest one when k remains constant.
  - The number of clusters (k) is a parameter. In this case, the procedure consists in two different steps. Firstly, the algorithm runs for k's values that belong to a range $[k_{min}, k_{max}]$. These extreme values are set beforehand by the user. Then, for each value of k, the best-obtained result is selected.

Finally, unsupervised strategies are sometimes used in a more complex work-flow. For instance, if clustering is used to create meaningful classes, it is possible to create an external dataset by hand-labeling, and to test the accuracy, by the so-called gold standard approach. In the same way, if pre-processing step is used to perform dimensionality reduction in a supervised learning procedure, its accuracy can be used as a proxy performance measure for the dimensionality reduction technique.

Appendix A.1 shows some validity indexes used to test the results gained from some of the clustering approaches proposed in this thesis. They are essentially based on relative criteria.

The next chapters introduce some few techniques on the distributed unsupervised learning approaches, which are the main topic of this thesis.

Chapter 3

Distributed Machine Learning

This chapter is devoted to the analysis of the Distributed Learning problem. Firstly, the problem will be introduced, and the requirements of a distributed context will be discussed. Successively, a state of the art review of the recently distributed machine learning approaches will be presented followed by an introduction on the structure of a typical distributed problem.

In particular, preliminary concepts on graph theory will be provided, focusing on the aspects that will be used in the rest of this thesis such as notions on neighbors, and diffusion of information among them.

3.1 Introduction on the problem

All of the supervised and unsupervised learning approaches described in the previous chapter have been applied in several contexts. However, in recent years they have become inadequate since data comes from several locations spread logically or physically in a network of agents. In particular, with the advent of big data, cloud storage, social networks and so on we must face a huge amount of information and data. Separate domains of application may largely impose different constraints on the solution, including low computational power at every location, limited underlying connectivity (e.g. no broadcasting capability), or transferability constraints related to the enormous bandwidth requirement. The computational power may not be sufficient to analyze a too high quantity of information and it could be required to much time to reach a solution.

Thus, for the sake of security, processing, and privacy, it is no longer possible to send all of the data in a central node where traditionally algorithms will find out the optimal partition, while new techniques able to model and manage these kind of information are mandatory. In this scenario, it would be challenging to avoid that a central node detains all of the information and achieve a solution that is similar to that obtained by a centralized approach through a process of communication and collaboration among the agents. Specifically, the lefthand side panel of Fig. 3.1 shows the centralized solution: all of the data are sent to a central authority where the three overall existing clusters are correctly identified; the distributed solution is shown on the right panel. It is important to underline that each node is able to reach the same solution of the centralized approach, even if some missing clusters are presented in the local dataset, by only exploiting the communication with the neighbors. Therefore, the question is, would it be possible to perform inference in a

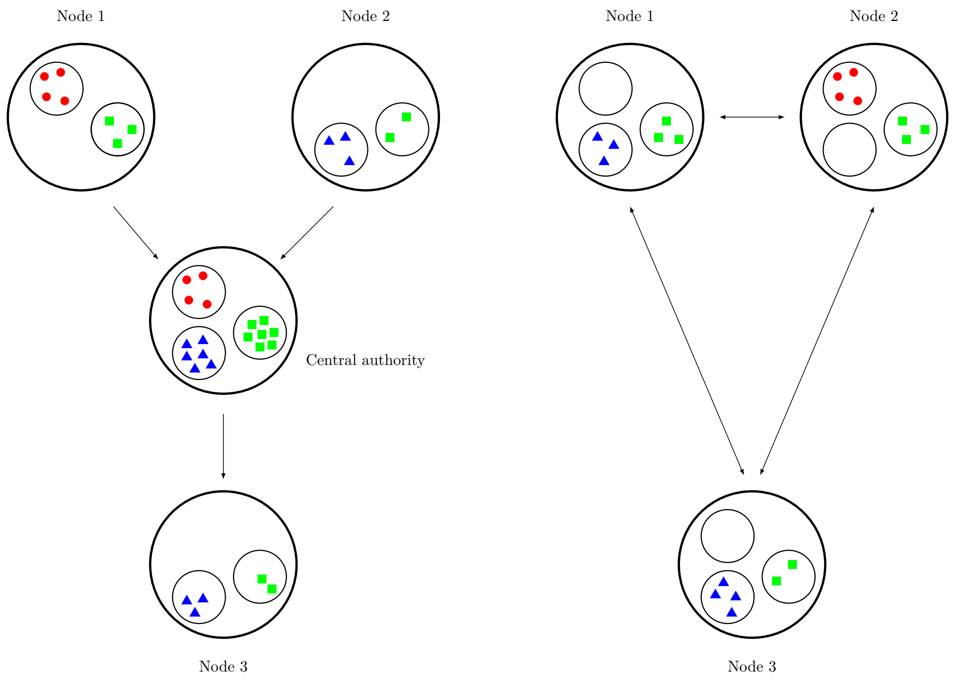


Figure 3.1. Centralized Vs Distributed approach

decentralized fashion (i.e. without a central coordinator), leaving out the exchange of training data? This is known as the *distributed learning* problem:

Distributed Learning: is the problem of inferring a function whenever the training data is distributed throughout a network of agents.

Generally, the distributed learning must verify the following constraints and characteristics:

- **Coordination**: in a fully distributed scenario, the agents must be able to communicate with the others without allowing any agent to coordinate the training process. In this way, fewer data needs to be arranged and analyzed.
- **Connectivity**: the assumption to perform in a distributed scenario requires a connection of the overall network, which means that each node can be reached from any other node in a finite number of steps.
- **Communication**: the communication infrastructure is a tool used to discriminate between distributed scenarios. The information can be exchanged via one-hop or multi-hop connectivity. In the latter, messages can be routed from any node to any other one of the network, while in the former, the node can exchange messages through the neighbors only. Obviously, the extreme case,

where each node is allowed to communicate with a single other one, is also admitted.

- **Privacy**: often, data could not be transferred from one site to another because it is confidential. In this case, a privacy violation refers to the need of exchanging local training patterns to other nodes in the network. For these reasons, security protocols must be added to avoid possible interceptions, and to preserve the sensitivity of the data.

- **Primitives**: an algorithm can be categorized according to specific mathematical primitives that are requested on the network. In some cases, when the message is exchanged from one site to another, vector sums, Hamiltonian cycles or more complex operations may be required. Moreover, they can be differently implemented based on the specific network's technology, for example with a DAC protocol.

- **Synchronization**: the algorithms differentiate on whether synchronization approach is used. Particularly, we could have a synchronous strategy during the communication procedure, or an asynchronous one where an external clock is not required to diffuse the information.

The distributed learning problem has gained significant importance in the last few years, drawing more and more attention of several authors. The new scenario has led to the realization of more suitable machine learning approaches by extending the traditional centralized algorithms [19].

The following section will provide a summary review of such approaches. In order to make things clearer, the exposition will have the same structure of the previous chapter, focusing on different learning approaches, which are the main topic of the thesis. The review aggregates works coming from multiple interdisciplinary fields.

3.2 State of-the-art

3.2.1 Distributed Hierarchical approaches

The Hierarchical approach will be our start of the analysis of the state-of-the art distributed learning. This kind of approach is particularly suited for multi-dimensional spaces, for that it naturally fits a parallel implementation of its centralized version [20]. Consequently, the basic idea has been discussed multiple times in the literature.

The work presented in [21] is among the first ones exploiting this concept. In particular, a parallel version of the Kruskals minimum spanning tree algorithm is presented. There are several processors that store the distance between one cluster and every other ones, and when new clusters are agglomerated there is no an updating step with the new ones. Unfortunately, this appears to cause incorrect clustering operation.

In successive works, there is the introduction of clusterheads. In particular, in [22] a distributed, randomized clustering algorithm able to organize the data of a wireless sensor network into clusters is presented. It is based on the creation of a hierarchy of clusterheads able to increase the saving energy, but invaliding the access protocol. Specifically, this strategy could affect the optimal probabilities of becoming a clusterhead.

A similar idea is derived in [23] where a different approach for creating the clusterhead is proposed. In particular, is reported a distributed weight-based energy efficient hierarchical clustering protocol (DWEHC). Each node first locates its neighbors (in its enclosure region) then, calculates its weights which are based on the residual energy and on the distance from the neighbors. The node with largest weight in a neighborhood may become a clusterhead.

Generally, the algorithms based on the hierarchical clustering are just a parallel version of the centralized ones or are extremely depending on the initial creation of the hierarchic communication structure. In particular, the node can be classified as forwarder or aggregators depending on the adopted strategy. In a distributed scenario, we want that each node has the same importance in the network and that the optimization strategy will be independent of any node adding or removal in the graph. For that, the presence of clusterhead may invaliding this purpose.

3.2.2 Distributed k-Means and Fuzzy C-Means

K-Means and Fuzzy C-Means are two of the simplest approaches used for clustering. The number of classes is fixed a priori and a pattern is assigned to one cluster in respect to a metric measuring the distance from that pattern to each cluster centroid. Many applications refer to extensions of these approaches able to face the distributed context.

In [24], for example, a distributed clustering algorithm called K-D Means is developed. It is based on a master-slave architecture applied in an Ethernet network. Data is divided into different subsets, one for each agent, whereas locally evaluated and the central points are sent between each others to diffuse the information. Successively, each agent determines the clusters by evaluating the distance between its own subset and the central points of the others agents. It is shown that if some data do not belong to the centroid of the local agent, then they will be transferred to the right cluster of another agent. The process will be repeated until a cost function becomes less than a certain threshold. This process requires high computation and communication costs, in addition to the still presence of a master slave architecture. In a successive work [25], the algorithm is improved by requiring only the exchange of central points and a limited number of patterns.

An important extension of the distributed k means is showed in [26] and is based on the implementation of the Optimistic Concurrency Control (OCC) approach, able to reduce the number of exchanged messages between local agents. In particular, it is able to better solve the conflicts of partial clustering results when is compared to a standard parallelised K-Means.

Similar concepts are also explored in [27] where a distributed clustering for the k-means approach is described. It is based on a distributed method for constructing a global corset to be used as a proxy for the entire dataset. The advantage lies on the construction of a small set of points, especially in those contexts where the communication is restricted to the edges of an arbitrary graph. Subsequently, the Principal Component Analysis (PCA) is used to improve the performance of the algorithm over large networks [28]. In this way, authors are able to obtain an algorithm whose communication costs are independent from the size and the dimension of the original dataset.

An alternative approach is applied in [29] where the authors extend the $K-$windows algorithm in a distributed scenario. The $K-$windows algorithm has the ability to endogenously determine the number of clusters. In this algorithm, no data exchange is allowed among the local nodes. The algorithm is executed locally in each dataset, and then the results are sent to a central node responsible of both the final merging of the windows and the construction of the final results.

The K-Means and Fuzzy C-means approaches are particularly suited for the distributed context but often, in literature, we can just find a parallel version of the corresponding centralized ones or versions depending on a coordination node. Thus, they have to be extended to make them suited for a fully distributed scenario.

3.2.3 Distributed Density based algorithm

In distributed clustering algorithms based on density models, the clustering process iterates until a neighbors density exceeds a given threshold. In this case, the density refers to the number of objects or data point to be clustered. One of the most important density-based clustering algorithms is the so called Density Based Spatial Clustering of Applications with Noise (DBSCAN).

The distributed density based algorithms are essentially different versions and extensions of the traditional DBSCAN approach. The first one here analyzed is a parallel version of DBSCAN [30]. The algorithm firstly uses R*tree to organize data in the central site, then stores the preprocessed data in each local agent, which communicates with the others by exchanging messages.

An alternative version is presented in [31], where a distributed version of DBSCAN, called DBDC is detailed. It relies on a central node that organizes the results and resolves conflicts. These works is extended in [32], where the LDBDC is able to fit noisy and high dimensional datasets.

Similar approaches, with the addition of privacy preserving protocol, have been extended in [33]. Each local node finds a clustering of its local data based on the kernel density function that is computed over all the data. The latter is approximated by exploiting statistical density estimation and sampling theory. The drawback is that the algorithm requires a central site to compute the necessary metrics to find an agreement on the parameters. The privacy is preserved not transmitting data values, but kernel based density estimation samples.

An alternative method is the Distributed Density Based Clustering (DDC) [34]. It is particularly suited for analyzing large, heterogeneous and distributed datasets. In each site, the local models are created by using the DBSCAN. Then they are aggregated by using tree based on topologies to construct global models.

All of these approaches are essentially a distributed version of the original DBSCAN, which guarantees the correct diffusion of the information by using a central node able to coordinate the communication protocol. However, as stated in the previous chapter, the distributed scenario has to be able to work without any reference node.

3.2.4 Distributed Gaussian Mixture Models

In this context, data is supposed to be sampled from an environment that can be described by a probability density function (PDF) as a mixture of elementary conditions. Generally, the Expectation-Maximization approach (EM) is used to estimate the mixture parameters in an iterative way.

An important work, which approaches this kind of scenario is presented in [35], where a parallel implementation of the EM, called DEM is presented. In particular, each node performs an incremental E-step and M-step by cycling through the network and using data provided by the previous node in the cycle. So, in other words, each node performs local computation on the sensor data and passes statistics to the next one in the network, through a node-to-node iteration process. The algorithm is able to converge to a stationary point of the log likelihood.

In [36], an iterative EM-like method is used to test different methods for clustering in the so-called clustering ensembles approaches. Specifically, it uses three different approaches: Iterative Voting Consensus (IVC), Iterative Probabilistic Voting Consensus (IPVC) and Iterative Pairwise Consensus (IPC). The first approach evaluates the clusters' center and assigns each point to the closest one. The second one differs only on the adopted metric, whereas the last one evaluates a similarity matrix by using information of the previous iterations.

The work in [37] shows a decentralized expectation-maximization algorithm applied in a distributed learning context, where data is acquired by a series of wireless sensors. The E-step relies on local information of the individual sensors, while in the M-steps, information is exchanged with the one hop neighbors to reach consensus. Unlike the others approaches, this scheme does not allows class conditional pdfs to be Gaussian. However, it requires bridge sensors to reach a consensus on the estimated parameters among the neighboring. Compared to the DEM approach, the last one is certainly more desirable without requiring an incremental looping through all nodes of the network. However, it is not allowed to be applied in a fully distributed scenario where no authority nodes have to been imposed to diffuse information among the agents.

The Expectation-Maximization approaches have been carefully studied to be applied in a distributed scenario. Being particularly suited for this context, in this thesis a version able to work in a purely distributed environment is proposed.

3.2.5 Distributed Self-organizing maps

The self organizing map algorithm is extremely simple in its description and practical implementation. However, it has been poorly investigated in the distributed scenario.

A preliminary work can be seen in [38], where a new self-organizing map is presented. The code vectors are distributed on a regular low-dimensional grid. It consists of a modified version of the widely used fuzzy c-means.

An alternative approach is discussed in [39], where a self-organized agent based architecture for power aware intrusion detection is provided. It describes an efficient way for selecting the network and monitoring, when the power consumption is the main limit in the architecture. The approach is able to preserve bandwidth.

3.3 Formulation of the problem

The previous section has presented a summary review of the recent approaches in the distributed scenario. In this paragraph, the problem is better formalized by making use of the graph theory. The distributed problems that will be introduced in the successive chapters, will always refer to a scenario similar to the one introduced as follows. Let us consider a set of L nodes in the network that could represent agents as well as sensors, hospitals, photovoltaic panels or more than this. The model can be formulated as a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, L\}$ is the set of the agents (nodes) and $\mathcal{E}$ is the set of edges (links) where $e_{ij} \in \mathcal{E}\ \forall \{i, j\}$ with $e_{ij} = e_{ji}$. The existence of a connection implies a communication between the two agents. A visual representation of this system can be given in Fig. 3.2. The common idea is that

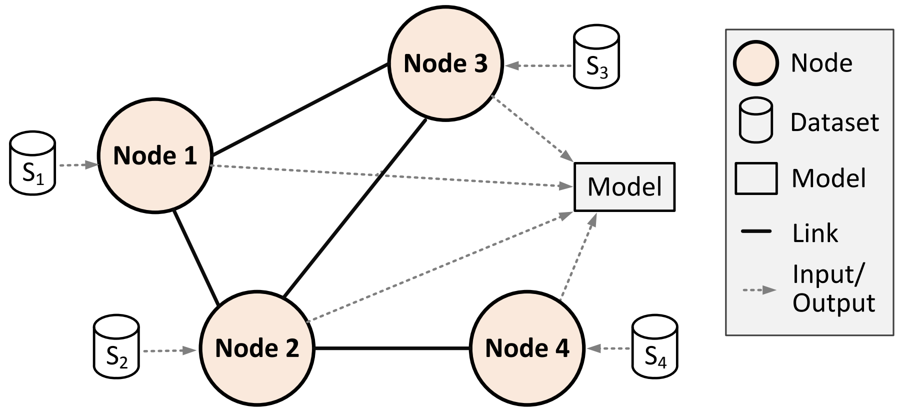


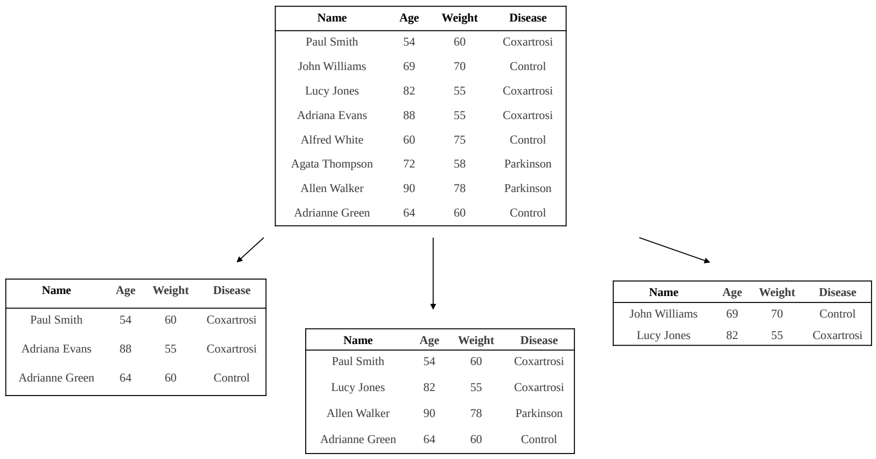
Figure 3.2. Distributed network of agents

each agent has access to a local training dataset $D_k \in \mathcal{D}$, such that $\bigcup_{k=1}^{L} D_k = D$. Generally, the standard input database is a matrix of data $\mathcal{D} \in \mathbb{R}^{nxM}$, having n number of distinct data points, also called vectors, transactions, or records, and M columns, also called dimensions, features, or attributes. When data is distributed among the several agents, the layout can be different depending on the row-wise or column-wise approach. Certain mining operations are more efficient considering a horizontal format, while others are more efficient using a vertical one. Generally, they can be described in the following way:

- *horizontal distribution:* the dataset $\mathcal{D}$ is partitioned in a local number of dataset $\mathcal{D}_i$ where $\mathcal{D}_i \in \mathbb{R}^{n_i xM}$ is such that the $\bigcup_{i=1}^{L} \mathcal{D}_i = \mathcal{D}$. In other words, different records are collected at different sites, but each record contains all of the attributes for the object it describes. In effect, each agent stores, as a unit, each transaction, along with the attribute values for that transaction. This is the most common and natural way in which data may be distributed. For example, a multinational company deals with customers of several countries, collecting data about different customers in each country. Understanding its customers around the world is important to set up a global advertising campaign.

- vertical distribution: the dataset $\mathcal{D}$ is partitioned in a local number of dataset $\mathcal{D}_i$ where $\mathcal{D}_i \in \mathbb{R}^{nxM_i}$ such that the $\bigcup_{i=1}^{L} \mathcal{D}_i = \mathcal{D}$. In other words, different attributes of the same set of records are collected at different sites and each dataset has the same number of rows, but different number of attributes. In these contexts, each agent combines each attribute with a list of all tids containing the item, and with the corresponding attribute value in that transaction. Consequently, each site will have a different view of the data. For example, a credit-card company may collect data about transactions by the same customer in different countries, and may want to deal with the transactions in different countries as different aspects of the customer total card usage.

In Fig. 3.3 and in Fig. 3.4 a visual representation of an horizontal and a vertical partitioning is illustrated. In the distributed context, we can also perform a



Name	Age	Weight	Disease
Paul Smith	54	60	Coxartrosi
John Williams	69	70	Control
Lucy Jones	82	55	Coxartrosi
Adriana Evans	88	55	Coxartrosi
Alfred White	60	75	Control
Agata Thompson	72	58	Parkinson
Allen Walker	90	78	Parkinson
Adrianne Green	64	60	Control

Name	Age	Weight	Disease
Paul Smith	54	60	Coxartrosi
Adriana Evans	88	55	Coxartrosi
Adrianne Green	64	60	Control

Name	Age	Weight	Disease
Paul Smith	54	60	Coxartrosi
Lucy Jones	82	55	Coxartrosi
Allen Walker	90	78	Parkinson
Adrianne Green	64	60	Control

Name	Age	Weight	Disease
John Williams	69	70	Control
Lucy Jones	82	55	Coxartrosi

Figure 3.3. Horizontal distributed data

discrimination on the kind of algorithm:

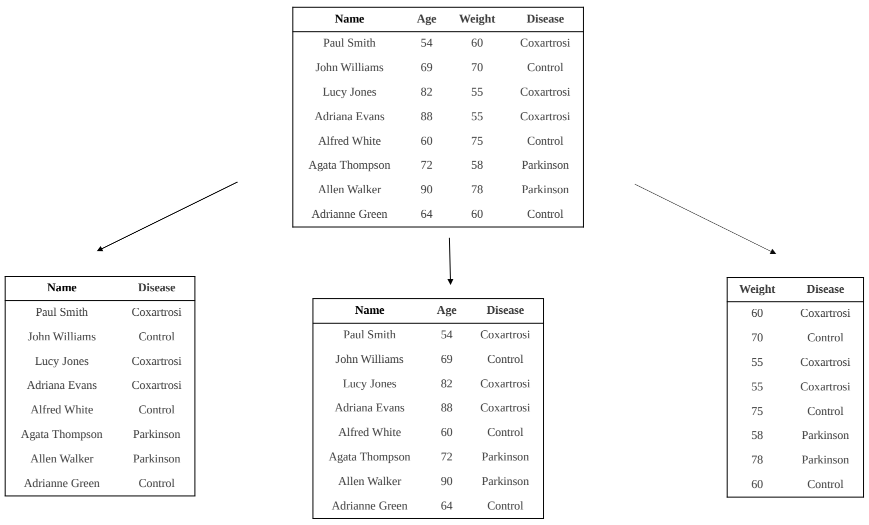
- *Parallel algorithms*: are applied in coupled systems, in architectures with multiple processors and in contexts where data is available on a centralized memory. The aim is to reduce the computational performances, making a particular computation as fast as possible by exploiting multiple processors.

- *Distributed algorithms:* are applied in a contexts where data is localized in a distributed memory and, unlike parallel processing, they do not assume shared memory since they are based on message passing.

In the rest of the thesis, a distributed algorithm in a horizontal scenario is always considered: each node represents a separate processor, and acquires the same number of attributes for each pattern, but a different number of instances.

Let us now focus on the distributed optimization problem.

Definition 3.1. Suppose that the kth agent has to minimize a generic objective function $\mathcal{J}_k(\mathbf{w})$, parameterized by the vector $\mathbf{w}$. Now, in the distributed scenario,



Name	Age	Weight	Disease
Paul Smith	54	60	Coxartrosi
John Williams	69	70	Control
Lucy Jones	82	55	Coxartrosi
Adriana Evans	88	55	Coxartrosi
Alfred White	60	75	Control
Agata Thompson	72	58	Parkinson
Allen Walker	90	78	Parkinson
Adrianne Green	64	60	Control

Name	Disease
Paul Smith	Coxartrosi
John Williams	Control
Lucy Jones	Coxartrosi
Adriana Evans	Coxartrosi
Alfred White	Control
Agata Thompson	Parkinson
Allen Walker	Parkinson
Adrianne Green	Control

Name	Age	Disease
Paul Smith	54	Coxartrosi
John Williams	69	Control
Lucy Jones	82	Coxartrosi
Adriana Evans	88	Coxartrosi
Alfred White	60	Control
Agata Thompson	72	Parkinson
Allen Walker	90	Parkinson
Adrianne Green	64	Control

Weight	Disease
60	Coxartrosi
70	Control
55	Coxartrosi
55	Coxartrosi
75	Control
58	Parkinson
78	Parkinson
60	Control

Figure 3.4. Vertical distributed data

the problem becomes that of minimizing the global joint cost function given by:

$$J(\mathbf{w}) = \sum_{k=1}^{L} J_k(\mathbf{w}) \tag{3.1}$$

subject to the constraints that $\mathbf{w} \in \mathbb{R}^d$:

$$\min_{\mathbf{w}} \sum_{k=1}^{L} J_k(\mathbf{w}) \text{ subject to } \mathbf{w} \in \mathcal{W} \tag{3.2}$$

For a single-agent, there exist several ways to find a solution to this kind of problem.

The Gradient Descent Procedure is the best representative way for minimizing a cost function in the form of (3.2). It is an optimization algorithm, used to find the parameters' value of a function, especially in those situations where they cannot be calculated analytically (e.g. using linear algebra). It computes the minimum of $J_k(\mathbf{w})$ by iteratively moving along the best direction to minimize the cost function, which is the gradient:

$$\mathbf{w}_k[n+1] = \mathbf{w}_k[n] - \eta_k \nabla_w J(\mathbf{w}_k[n]) \tag{3.3}$$

where η_k is the local step-size at time k, whose sequence should be sufficiently small in order to guarantee convergence to the global optimum, allowing $J(\mathbf{w}[n+1]) \leq J(\mathbf{w}[n])$. Many works exploit the additive property of the gradient update, finding a solution for the problem (3.2) by summing the gradient contributions from each local node. Specifically, an application of convex unconstrained problems can be found in [40, 41], while in [42, 43, 44] an extension on convex constrained problems or non-convex problem [45].

Besides the gradient descent approaches, other representative methods used to solve (3.2) can be found in subgradient descents methods [41]. Unlike the gradient ones, they are convergent even when applied to a non-differentiable objective function. The general iterative step can be expressed as follows:

$$\mathbf{w}_k[n+1] = \mathbf{w}_k[n] - \eta_k \mathbf{g}_k[n]) \tag{3.4}$$

where $\mathbf{g}_k$ denotes the subgradient of $J_k(\mathbf{w})$.

Additionally, the dual averaging [43] algorithm is generally designed for minimizing a potentially nonsmooth convex function by using a proximal function ϕ that is assumed to be 1-strongly convex compared to some norm. In particular, the proximal function satisfies:

$$\Psi(y) \geq \Psi(x) + \langle \nabla\Psi(x), y - x\rangle + \frac{1}{2}\|x-y\|^2 \text{ for all } x, y \in \mathcal{X} \tag{3.5}$$

The generic iterations at time $[n]$, when the algorithm receives a subgradient $\mathbf{g}_k[n] \in \partial J_k(\mathbf{w})$, can be formalized as follows:

$$\begin{aligned} \mathbf{z}_k[n+1] &= \mathbf{z}_k[n] + \mathbf{g}_k[n]; \\ \mathbf{w}_k[n+1] &= \Pi_{\mathcal{W}}^{\Psi_k}(\mathbf{z}_k[n+1], \alpha_k[n]) \end{aligned} \tag{3.6}$$

where $\{\alpha_k[n]\}_{t=0}^{\infty}$ is a non-increasing sequence of positive stepsizes and:

$$\Pi_{\mathcal{W}_k}^{\Psi}(\mathbf{z}_k, \alpha_k) := arg \min_{\mathbf{w}_k \in \mathcal{W}} \{\langle \mathbf{z}_k, \mathbf{w}_k\rangle + \frac{1}{\alpha\, k} \Psi_k(\mathbf{w})\} \tag{3.7}$$

is a type of projection. Specifically, the underlying intuition of this approach is the following: given the current iteration $(\mathbf{w}_k[n], \mathbf{z}_k[n])$ the next one $(\mathbf{w}_k[n+1])$ is evaluated by minimizing an averaged first-order approximation of the function J. Finally, another important approach used to solve convex optimization problem, like the one in (3.2), can be the alternation direction method of multipliers (ADMM) [44] that breaks the first original formulation, into subproblems that are simpler to be handled. Considering the problem:

$$\begin{aligned} \min_{\mathbf{w}_k, \mathbf{z}_k} \quad & J_k(\mathbf{w}) + g_k(\mathbf{z}) \\ \text{s.t} \quad & A\mathbf{w}_k + B\mathbf{z}_k = c \end{aligned} \tag{3.8}$$

the augmented Lagrangian of the problem can be defined as:

$$L_k(\mathbf{w}, \mathbf{z}, \mathbf{y}) = J_k(\mathbf{w}) + g_k(\mathbf{z}) + \mathbf{y}_k^T(A\mathbf{w}_k + B\mathbf{z}_k - c) + \frac{1}{2}\|A\mathbf{w}_k + B\mathbf{z}_k - c\|_2^2 \tag{3.9}$$

In this way, the generic iterations of the ADMM procedure can be defined as:

$$\begin{aligned} \mathbf{w}_k[n+1] &= \arg\min_{\mathbf{w}_k} L_k(\mathbf{w}, \mathbf{z}[n], \mathbf{y}[n]) \\ \mathbf{z}_k[n+1] &= \arg\min_{\mathbf{z}_k} L_k(\mathbf{w}_k[n+1], \mathbf{z}_k, \mathbf{y}_k[n]) \\ \mathbf{y}_k[n+1] &= \mathbf{y}_k[n] + \rho(A\mathbf{w}_k[n+1] + B\mathbf{z}_k[n+1] - c) \end{aligned} \tag{3.10}$$

The ADMM approach can be related to the single Gauss-Seidel pass over $\mathbf{w}_k$ and $\mathbf{z}_k$.

3.4 Notion of Algebraic Graph Theory

In this section, some preliminary concepts of graph theory that will be used in the rest of the thesis will be introduced. In particular, there will be introduced the necessary structures to perform communication in a network of agents, as well the related adjacency matrix, weight matrix, number of neighbors and so on. The first concept is about the connection among agents. Specifically, we will say that if two vertices are connected by an edge, then they will be adjacent. Extending the concept to all of the agents, the adjacency matrix can be defined as:

Definition 3.2. The adjacency matrix $A \in \mathbb{R}^{VxV}$ in a undirected graph is a matrix such that:

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge between node } i \text{ and } j \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

If there are no self edges the diagonal values are all zero. In this case, it is a symmetric matrix: if there is an edge between node i and node j, there will be an edge between node j and node i. Otherwise, in a directed network the adjacency matrix could not be symmetric.

Definition 3.3. The adjacency matrix $A \in \mathbb{R}^{VxV}$ in a directed graph is a matrix such that:

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge from node } j \text{ and } i \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

Another importation concept related to the graph is the neighborhood of each node. In particular, once set the neighbors of each node i as $\mathcal{N}_i = \{j \in \mathcal{V} : e_{ij} \in \mathcal{E}\}$ for all $i = 1, \ldots, N$, the Degree matrix can help us in finding out how many neighbors each node has:

Definition 3.4. The Degree Matrix of the node $v_i \in \mathcal{V}$ provides information on the number of edges attached to it:

$$D_i = \sum_{j=1}^{V} A_{ij} \tag{3.13}$$

It is a diagonal matrix, where the elements on the main diagonal represent the degree of the single node, while the others ones are set to zero. The concept of a number of neighborhood can be extended by analyzing the number of two or more hops neighbors.

Definition 3.5. The two hops neighbors from node i and node j can be defined as:

$$\sum_{k=1}^{V} A_{ik} A_{jk} \tag{3.14}$$

This can be repeated to find out the neighbors of three or more hops:

Definition 3.6. The three hops neighbors from node i and node j can be defined as:

$$\sum_{k=1}^{V} A_{il}A_{lk}A_{kj} \tag{3.15}$$

Until now, an unweighted graph has been considered, but often each edge is important when is related to the others. Therefore, let us consider how the previous notations change when we consider a weighted graph.

In the rest of the thesis an undirected and weighted graph $\mathcal{G}$ is always considered, so each edge between two vertices v_i and v_j carries a non-negative weight $w_{ij} \geq 0$. In this way, the weight adjacency matrix becomes:

Definition 3.7. The weight adjacency matrix $W \in \mathbb{R}^{VxV}$ can be defined as:

$$W_{ij} = \begin{cases} w_{ij} & \text{if there is an edge between node } i \text{ and } j \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

If $w_{ij} = 0$ then the vertices v_i and v_j are not connected by an edge. While, the degree matrix of a vertex can be redefined as follows:

Definition 3.8. The Degree Matrix of the node $v_i \in \mathcal{V}$ can be defined as:

$$D_i = \sum_{j=1}^{V} w_{ij} \tag{3.17}$$

The adjacency matrix and the degree matrix capture the entire structure of the network and theirs properties tell us a variety of useful things about the graph, but to have a complete vision of the system, the Laplacian matrix must be added.

Definition 3.9. Given a graph $\mathcal{G}$, its (weighted) weighted matrix $W(\mathcal{G}) = (w_{ij})$ and its degree matrix $D(\mathcal{G})$, then the unnormalized graph Laplacian is defined as:

$$L(\mathcal{G}) = D(\mathcal{G}) - W(\mathcal{G}) \tag{3.18}$$

The elements of L can be given by:

$$L_{ij} = \begin{cases} deg(v_i) & \text{if } i == j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & otherwise. \end{cases} \tag{3.19}$$

Additionally the Laplacian matrix can be formulated in an alternative way:

Definition 3.10. Given a graph $\mathcal{G}$, its (weighted) weighted matrix $W(\mathcal{G}) = (w_{ij})$ and its degree matrix $D(\mathcal{G})$, then there are two matrices which are called normalized graph Laplacian in the literature:

$$\begin{aligned} L_{sym} &:= D^{-\frac{1}{2}}LD^{-\frac{1}{2}} = I - D^{-\frac{1}{2}}W^{-\frac{1}{2}} \\ L_{rw} &:= D^{-1}L = I - D^{-1}W \end{aligned} \tag{3.20}$$

.

3.5 Distributed Average Consensus

Distributed Average Consensus (DAC) is an iterative distributed protocol for computing the global average of a series of local measurements over a network of agents without the need of a master node, but relying on local communications between neighbor nodes [46, 47]. This simplicity of implementation, coupled with the robustness to failures of single nodes, makes the DAC protocol suitable to be implemented on a variety of networks, including those with restrictive constraints in terms of computational capability.

Let us suppose that each agent i of the network has a vector of measurements θ_i which represents its estimate of the quantity θ.

The goal of the protocol is to converge for all the agents to the global average:

$$\hat{\theta} = \frac{1}{V}\sum_{i=1}^{V}\theta_i \tag{3.21}$$

For discrete-time distributed systems the $(n+1)$th iteration of the protocol is defined by a set of linear updating equations:

$$\theta_l[n+1] = \sum_{i=1}^{V}\theta_i[n] \tag{3.22}$$

which can be represented in form of a linear system:

$$\theta[n+1] = W\theta[n] \tag{3.23}$$

The convergence behavior of DAC is asymptotic [46], but in practical applications, the procedure is completed or stops when a maximum number of iterations is reached, or when for each agent, the variation from the current estimate is lower (in norm) than a tolerance threshold ϵ:

$$\|\theta_i[n+1] - \theta_i[n]\|_2^2 < \epsilon \forall i = 1, \ldots L. \tag{3.24}$$

The convergence properties of the algorithm are strictly linked to the matrix W. If the weights W are chosen appropriately, the iterations defined in (3.24) converge locally to the average (3.21). In particular, the previous inequality is true if and only if:

$$\lim_{t\to\infty W^t} = \frac{\mathbf{1}\mathbf{1}^t}{n} \tag{3.25}$$

In particular (3.25) holds, if and only if:

$$\begin{aligned} \mathbf{1}^T W &= \mathbf{1}^T, \\ W\mathbf{1} &= \mathbf{1}, \\ \rho(W - \mathbf{1}\mathbf{1}^T/\eta) &< 1 \end{aligned} \tag{3.26}$$

where $\rho(\cdot)$ denotes the spectral radius of a matrix [48].

In case of undirected, connected networks, there are several ways to make the weight matrix able to satisfy the (3.26) equation. Different strategies for the DAC protocol correspond to the different choices of the weight matrix. Clearly, the choice depends on the available information at every node about the network topology and on their specific computational requirements. The list below will describe the most used weight matrices able to satisfy the previous requirements.

Definition 3.11. The **Maximum degree weighting** [48] method is defined as:

$$w_{ij} = \begin{cases} \frac{1}{(d+1)} & i \neq j, \{i,j\} \in E \\ 1 - \frac{d_i}{(d+1)} & i = j \\ 0 & i \neq j, \{i,j\}, \notin E \end{cases} \tag{3.27}$$

where d_i is the degree of the i-th node, and d is the maximum degree of the network.

Definition 3.12. The **Metropolis-Hasting** weights [49] for the connectivity matrix W is defined as:

$$w_{ij} = \begin{cases} \frac{1}{\max\{d_i,d_j\}+1} & i \neq j, \{i,j\} \in E \\ 1 - \sum_{j \in \mathcal{N}_i} \frac{1}{(max\{d_i,d_j\}+1)} & i = j \\ 0 & i \neq j, \{i,j\} \notin E \end{cases} \tag{3.28}$$

where $\mathcal{N}_i$ is the set of nodes indexed directly connected to node i.

Definition 3.13. The **Minimum Asymptotic** method [48]is realized in order to minimize the asymptotic convergence factor $\rho(\mathbf{W} - \mathbf{1}\mathbf{1}^T/N)$ where $\rho(\cdot)$ denotes the spectral radius operator. This is achieved by minimizing the following constrained optimization problem:

$$\begin{aligned} \text{minimize} \quad & \rho(\mathbf{W} - \mathbf{1}\mathbf{1}^T/N) \\ \text{subject to} \quad & W \in \mathcal{W}, \mathbf{1}^T\mathbf{W} = \mathbf{1}^T, \mathbf{W}\mathbf{1} = \mathbf{1} \end{aligned}$$

As it has been ascertained in [48], the problem is not convex, but it can be associated with a semidefinite programming (SDP), and being solved using efficient ad-hoc algorithms.

Definition 3.14. The **Laplacian Heuristic** is based on a heuristic approach [48] that uses a constant edge weights matrix:

$$\mathbf{W} = \mathbf{I} - \alpha\mathbf{L} \tag{3.29}$$

where $\alpha \in \mathbb{R}$ is a parameter defined by the user, while $\mathbf{L}$ is the Laplacian Matrix associated to the network. The weights obtained by (3.29) satisfy the following relationship when they converge:

$$\begin{aligned} \rho(\mathbf{W} - \mathbf{1}\mathbf{1}^T/N) &= max\{\lambda_2(\mathbf{W}), \lambda_n(\mathbf{W})\} \\ &= max\{1 - \alpha\lambda_{n-1}(\mathbf{L}), \alpha\lambda_1(\mathbf{L}) - 1\} \end{aligned} \tag{3.30}$$

where $\lambda_i(\mathbf{W})$ is the i-th eigenvalue associated to $\mathbf{W}$. The value of α that minimizes (3.30) is given by:

$$\alpha^* = \frac{2}{\lambda_1(\mathbf{L}) + \lambda_{N-1}(\mathbf{L})} \tag{3.31}$$

If the weight matrix is chosen in this way, it will be sure be double stochastic symmetric.

3.6 Conclusive Remarks

In this chapter, it is presented a list of the main important approaches in the distributed scenario. In particular, by analyzing the previously cited works, it is possible to say that most of the works are just a parallel version of the original centralized techniques. Accordingly, they are not actually distributed since they just split the computation and successively send the results to a central node that makes a decision. In others applications, bridge sensors, or a loop through all the network's node is necessary to reach a common agreement on the final result.

In the rest of the thesis, a work on the fully distributed scenario is carried out. Focusing on both the supervised and unsupervised learning scenarios, different techniques where nodes are allowed to exchange information only through the neighbors are presented. There is no need of central authority to find agreement among local results.

Part II

Proposed Distributed Approaches

Chapter 4

A decentralized approach for distributed ensemble clustering

This chapter introduces a heuristic approach for distributed learning. It is based on the ensemble clustering technique, which is widespread in the centralized case, due to its ability to improve the quality of individual data clustering. At the same time, its use on distributed learning has been relatively limited, which is the main motivation for this chapter. In Section 4.2.1, the ensemble cluster procedure is extended in order to make it suitable to a fully distributed scenario. Next, the approach is evaluated on multiple real-world contexts.

4.1 Ensemble Clustering

Cluster ensemble has proved to be a good alternative when facing cluster analysis problems. As is illustrated in Fig. 4.1, the idea behind this approach is very simple: to obtain the final result, a set of partitions is extracted from the same dataset for successively be combined into the final clustering. In this way, the quality of individual results is improved. These techniques are particularly suited for very large dataset since they are able to obtain a common structure on partitions [17, 50, 51].

So, formalizing these concepts, if a set of n patterns $\{x_1, x_2, \dots, x_n\}$ is considered, different partitions m of the same dataset can be performed $\mathbb{P} = \{P_1, P_2, \dots P_m\}$, where each $P_i = \left\{C_1^i, C_2^i, \dots, C_{k_i}^i\right\}$ is a partition of the set X with k_i clusters.

Definition 4.1. The *ensemble clustering* technique aims at finding out the consensus partition $P^* \in \mathbb{P}$ which better represents the properties of each solution.

It generally works by applying two different steps [17]:

- Generation phase: is the process by which the different partitions are obtained. Generally, it influences the final result, so an appropriate clustering algorithm must be chosen to obtain good performances.

- Consensus phase: is the process by which the different partitions are combined and weighted each others to find out the global one that better fit, with respect to a defined metric, the existing clustering.

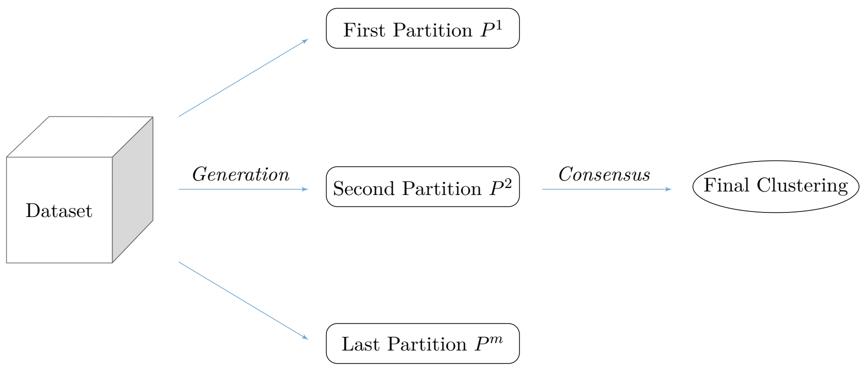


Figure 4.1. Diagram of the general process of cluster ensemble

Leveraging the consensus across multiple clustering results, provides more accurate and stable solutions when compared to the traditional single clustering techniques. For this reason, the consensus step is considered the core of these approaches and a lot of techniques have been developed to find agreement among the results. However, the commonly used approaches are generally focused on two main consensus functions: the object co-occurrences and the median partition. In the first one, the idea is to determine the label associated with each object in the partition. This is performed by analyzing how many times an object belongs to one cluster or how many times two objects belong together to the same cluster. In effect, the label association is a typical problem of the unsupervised learning approach. Since the label associated with each object in a partition is symbolic, there is no relation between the set of labels given by a clustering algorithm and the one given by another one. Hence, at the end of the generation phase, a relabeling procedure is mandatory for find out an agreement. Among the relabeling based methods, Plurality Voting (PV), Voting for fuzzy clusterings, Voting Active Clusters (VAC), Cumulative Voting (CV) and many others can be found.

The second consensus approach, which is the median partition, can be formulated as an optimization problem where the aim is to find out the partition that maximizes the similarity among all the partitions of the ensemble:

$$P* = arg \max_{P \in \mathbb{P}} \sum_{j=1}^{m} \Gamma(P, P_j) \tag{4.1}$$

where Γ is the similarity measure between partitions. Generally, the approaches differ on the local measurement criteria. Below, there are summarized the most important ones:

- Counting pairs measures: are able to count the pairs of objects on which two partitions agree or disagree. (Jaccard coefficient, Rand Index, Fowlkes-Mallows Index and so on).
- Set matching measures: are based on set cardinality comparisons (F measure).
- Information base theory: gives an indicator of the information shared between two partitions (class entropy, normalized mutual information).

- Kernel measures: are specifically designed for the median partition and are proven to be positive semidefinite kernels (graph Kernel based measure).

Recently, an additional criterion has been presented in [52], where to judge the correctness of a partition, a local similarity criterion δ is created. In particular, the index can be used to evaluate the similarity between two results and their quality. It can be expressed as follows:

$$\gamma^{i,j} = \frac{1}{2}\Big(p_s\Big(\frac{1}{n_i}\sum_{k=1}^{n_i}\omega_k^{i,j} + \frac{1}{n_j}\sum_{k=1}^{n_j}\omega_k^{j,i}\Big) + p_q(\delta^i + \delta^j)\Big) \tag{4.2}$$

where

$$\omega^{i,j} = S(C_i^k, CC(C_i^k, R_j)) \tag{4.3}$$

and p_s, p_q are two arbitrary parameters given by the user ($p_s + p_q = 1$). In 4.3, S is the similarity between the k-th cluster of the i-th result (C_i^k) and CC is the corresponding cluster of C_i^k found in the j-th result. After the resolution of the local conflicts, a global agreement coefficient Γ must be evaluated for the management of the global results.

$$\Gamma = \frac{1}{m}\sum_{i=1}^{m}\Gamma^i \tag{4.4}$$

where

$$\Gamma^i = \frac{1}{m-1}\sum_{j=1, j\neq i}^{m}\gamma^{i,j} \tag{4.5}$$

However, the steps required to evaluate this measure are complex and onerous in terms of time and resources. Additionally, the ensemble clustering criteria focus only on the consensus' agreement without allowing a change or a creation of a new partition. Consequently, they consider the initially provided partitions as the only necessary ones. In the next section, these limits have been overcomed by extending the cluster ensemble technique in a distributed context. Hence, differently from the centralized approach where the different partitions are obtained by considering the same initial dataset in a central location and varying only the clustering algorithms, in the proposed approach is added the possibility of treating data spread logically in a network of agents. Specifically, in this scenario, partitions could be obtained by using different starting dataset, (accordingly to what each agent is able to initially acquire) and/or varying the particular clustering approach. Furthermore, the global common structure is reached through cluster validation indexes able to reduce the total amount of exchanged information by sending only the local representatives among nodes. Finally, to be completely independent from the original partitions, the construction of new solutions will also be allowed.

4.2 Distributed ensemble clustering

In this section, the clustering ensemble techniques presented in [52] has been extended in a distributed setting. The novelty of the approach presented in this chapter, is the capability of working with a fully distributed scenario where there is a network of agents linked together and that acquires data autonomously. To reach the final consensus, the agents are not forced to have the same local dataset, while they are allowed to start with a different partition constructed by considering theirs own data only. In this way, it is ensured the proper functioning in the worst distributed case, testing what happens when the number of the initial clusters at each node is different too. Even when two nodes have both access to the same cluster they still work with separate data points. For instance, if we think to a sensor network that has to measure the temperature or the level of air pollution in a physical environment using sensors spread all over the land, it is not realistic to assume that each node starts with the same temperature or even the same level of pollution. Additionally, regardless what is done by the usual centralized ensemble clustering approaches where all of the data are necessary to reach an agreement, in the proposed approach only the local representatives are communicated among the neighbors. In this way, the computational cost and the information exchanged are substantially reduced.

Before introduce the proposed algorithm, the following condition has to been assumed:

Assumption 4.2. Supposing that the connectivity is known a priori and fixed, then the network will be:

- fully connected (every node can be reached from any other node);
- undirected (the adjacency matrix is symmetric).

The algorithm is based on a simple idea: the collaboration is carried on by exchanging information between agents and by using a consensus voting algorithm. Computing a consensual result from clustering having a different number of clusters is generally a difficult task, due to the lack of trivial correspondence between the resulting partitions. In this work, the collaborative process consists of an automatic and mutual refinement of the results, until the different partitions become statistically similar with a good internal validity index. In particular, the indexes will check the disagreement among the partitions, allowing the agents to detect the main conflicts and solve them by merging or splitting clusters. Additionally, since these operations are not always necessary the control with a cluster validation index will help us to check the improvement or the worsening of the adjustments. One of the novelties introduced is precisely related to the management of the local changes of a clustering result. Avoiding the use of onerous metrics as (4.2), three validity indexes, further detailed in Appendix A.1 have been considered to evaluate the local changes based on the global information:

- *Davies-Bouldin Index* ([53]);
- *Dunn Index* ([54]);
- *DW-DB* ([55]).

Whenever a modification takes place in a local result, the agent computes the index taking into account the other results. If the modified result obtains better indexes the changes are accepted and the centroids re-computed, otherwise the conflict is eliminated and the algorithm iterates. This check is necessary to avoid to fall into a local minimum or maximum, in fact, the multiple iterations of merge and split without any check could produce the situation in which there is one overall cluster for all the objects or many clusters as the objects are. There are differences in performance depending on which index is used. The DB index performs well in highly differentiated datasets but is too inaccurate when the clusters are not well separated. The Dunn index is useful when there are a lot of attributes but it is harder to compute and tends to fall in a situation where only one global cluster is found. The DW-DB index performs better than the others in both separable or non-separable classes and different datasets.

This first strategy introduced in this thesis for learning in a fully decentralized way is simple, yet it results in a highly efficient training algorithm. The Validated Distributed Ensemble Clustering (V-DEC) is composed by four different steps:

4.2.1 Initial Clustering

In the first step, each node has a partial vision of the entire environment. Autonomously the agents try to find out the local partition of their own data by applying a clustering algorithm. In particular, the different local results should be obtained employing several algorithms (K-means, Fuzzy C-means, Spectral clustering, EM) or varying theirs parameters (using different dissimilarity measures, number of the cluster or random initial centers) or just re-sampling or reweighing the set of objects (different bootstrap sample for given data). There are no constraints about how the partitions could be obtained. The use of heterogeneous clustering algorithms balances the performance.

The novelty is the capacity to work with different datasets without require each node to start with the same local dataset. Even different object representations, different subsets of objects or projections of the objects on different subspaces could be used. Hence, completely different local datasets are used to ensure the proper functioning in the worst distributed case, testing what happens when also the number of the initial cluster at each node is different. For the sake of simplicity, it has been decided to work mainly with the K-means algorithm (and the EM for the comparison), assuming to initialize each agent with the same global number of clusters; in this way, each node is forced to search more clusters compared to those it really sees. It is important to underline that the algorithm works the same also if agents do not have access to all clusters.

Fig. 4.2 introduces a toy problem used to better explain the steps of the algorithm. In the initial clustering, there are several errors: some clusters are split and others one are classified as one.

4.2.2 Collaboration Phase

The collaboration phase is composed of two steps: the conflict detection and the conflict resolution.

Conflict detection

The collaboration phase stems from the fact that each node has to reach a solution that is similar to the centralized one, where all of the patterns are sent and analyzed in a single central location. Thus, the first step consists in discovering the main

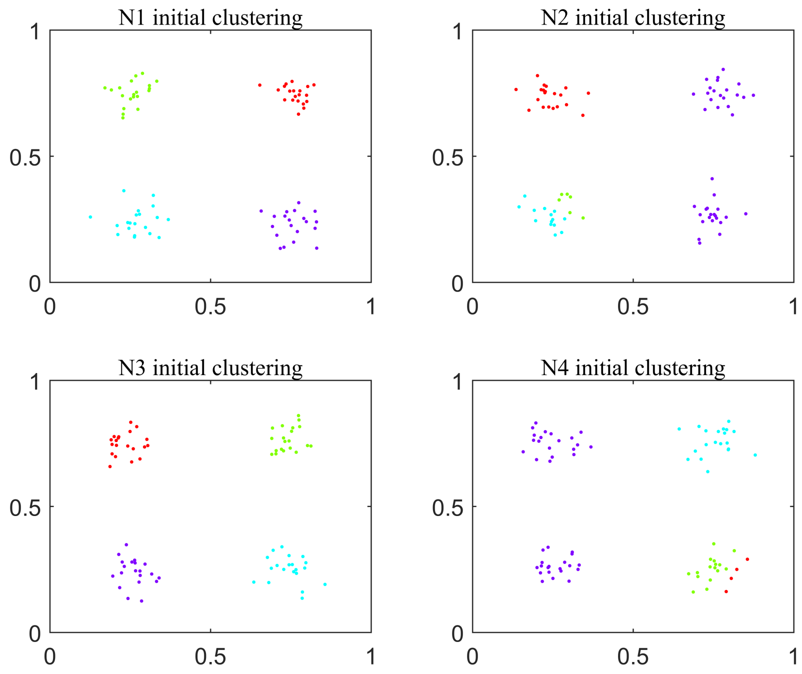


Figure 4.2. Toy problem initial clustering

conflicts among the local partitions. Each node exchange the centroids' information, or the local representative of the particular algorithm, to construct a global similarity matrix *Omega (Ω)*. The similarity S(C_i^k ,C_i^k) between two clusters is evaluated by observing the relationship between the size of their intersection and the size of the cluster itself. In this way, a measure on the distance between the clusters of a node and those founded by the other agents is obtained.

Additionally, a corresponding function between objects from each pair of the dataset is defined in order to observe which cluster from a result is the most similar to a cluster of an another one.

Definition 4.3. The Corresponding Cluster (CC) of C_i^k from the result R_i is founded as the most similar C_i^k of the result R_j, with $i \neq j$ as the following equation formalizes:

$$CC(C_i^k, R_j) = C_j^l \tag{4.6}$$

with

$$S(C_i^k, C_j^l) = max\{S(C_i^k, C_j^m), \forall m \subseteq [1, n_j]\} \tag{4.7}$$

where n_j is the number of clusters found in the result R_j by agent j.

Finally, the conflict importance has to be evaluated.

Definition 4.4. The conflicts' importance K is evaluated performing the inverse of the similarity value between a cluster and its CC (making it ranged in the interval 0-1):

$$\mathcal{K}_k^{i,j} = 1 - S(C_i^k, CC(C_i^k, R_j)) \tag{4.8}$$

It could happen that no conflict occurs between C_i^k and its CC in the j-th result: in that case, a NaN is inserted in the matrix.

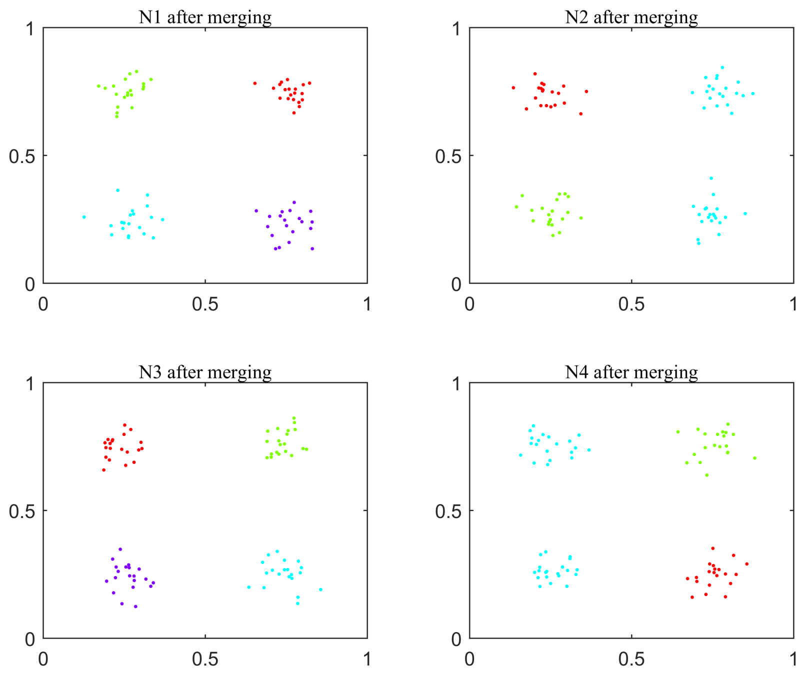


Figure 4.3. Toy problem after merging

It has to be noted that, in the cluster ensemble technique [52], a conflict occurs only when the similarity between a cluster and its CC is less than one. Whereas, in an actual distributed environment, a conflict may occur also when different clusters of the same result have the same CC. This occurrence is thus considered as a conflict in this version of the algorithm.

Conflict Resolution

Once the conflicts are detected, the most important ones, according on their value of K, are selected to be solved. Generally, the local resolution consists in applying merging and splitting operators on each involved conflicts in order to improve the similarity among partitions. The choice is based on the conflict's importance (value of K): if it is under a certain threshold the clusters are merged together, otherwise, the clusters will be split. This situation corresponds to the case in which one cluster has two or more corresponding cluster in another node, or otherwise when in a cluster there are more classes than those that really exist. In the traditional cluster ensemble approaches [52], the normally operations involved in the resolution of a conflict are three: merging, splitting or reclustering. In particular, the merging and the splitting are applied together (when a cluster is merged the other involved in the conflict is split) while the recluster is used if the initial clustering result is under a certain goodness threshold. In the V-DEC procedure, this version is revised by making it more suited for a purely distributed environment, changing the operations involved. In the proposed algorithm, the merging operation is simply done by reassigning the labels coherently after selecting and storing the elements which generate conflicts. It can be seen in Fig. 4.3 when two clusters are merged

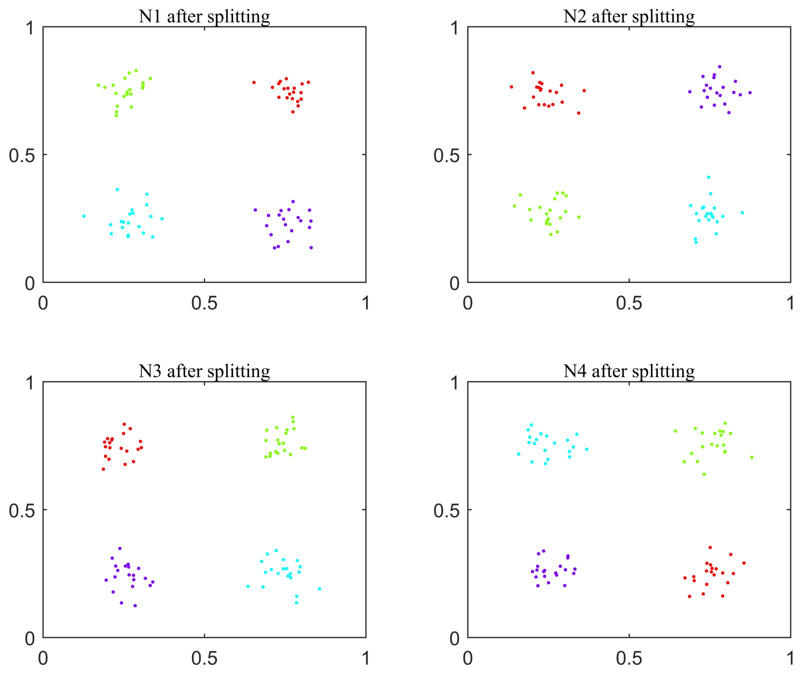


Figure 4.4. Toy problem after splitting

together (in the toy problem example).

Instead, the split operation requires a bit more attention. The K-Means algorithm is applied again to the conflict's data point, initializing the number of clusters at two; hence each cluster involved in the conflict is split into two different clusters coherently. It has to be noted that, in some cases, the algorithm could apply the split operator also when it is not required, that's why a validation index is inserted to check if the new clustering result is effectively better than the previous one. In this way, splitting results are carried on only if the validity index of the new cluster is better than the old one, otherwise, the changes on labels are discarded. After the resolution, the detection of the conflicts must be reiterated and the cluster labels must be normalized to ensure that each result has a complete set of labels (if there are N clusters in a result they should be labeled from 1 to N). The centroids are also recomputed after each modification of the results. As it can be seen from Fig. 4.4 the split operators is able to divide all of the clusters erroneously created by the initial clustering.

4.2.3 Additional refinement steps

All of the previous steps are sufficient to converge to the centralized solution when the complete dataset is given to the agents. Otherwise, an additional step could become necessary. In effect, in very few cases, it could happen that at the end of the procedure the similarity matrix Omega is still not diagonal. This means that there are some additional conflicts to be solved. Hence, the clusters in the columns of Omega are firstly merged by producing a big cluster that necessitate an accurate splitting (this type of conflict is not taken into account in the traditional clustering ensemble techniques):

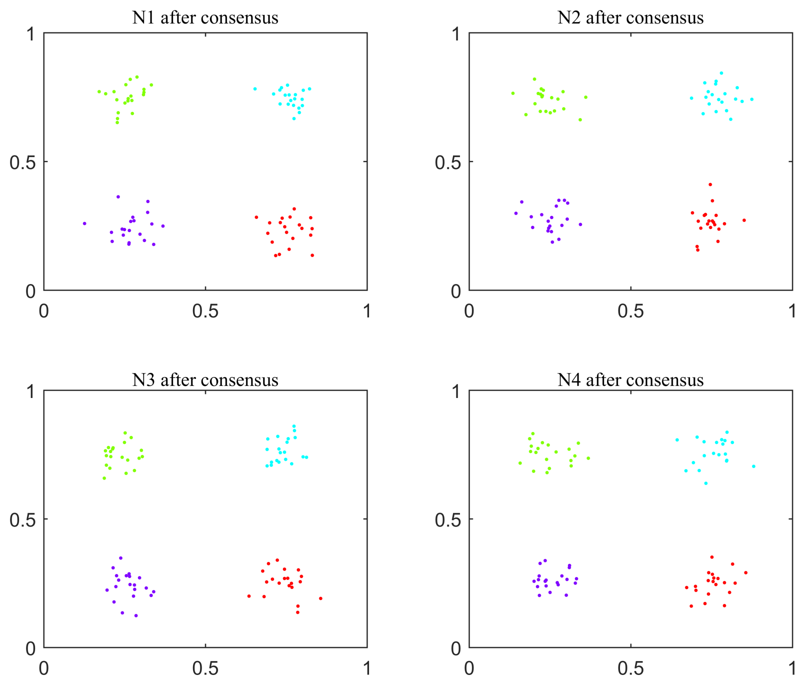


Figure 4.5. Toy problem after consensus

Assumption 4.5. A cluster (k) in the i-th result must be split when its CC in the j-th result (i $\neq j$) is the same of another cluster (m) of the i-th result found in the j-th result, and this can be formalized as follows:

$$CC(C_i^k, R_j) = CC(C_i^m, R_j) \tag{4.9}$$

with $i \neq m$ and $m \neq k$.

In these additional steps, the performances of the DB and DW-DB indexes are very similar, while the Dunn index is inappropriate and thus ignored because does not allow any splitting of the clusters.

4.2.4 Consensus computation

In the unsupervised learning environment the number of classes and the corresponding labels are unknown. Hence, at the end of the procedure all of the conflicts have been solved and different partitions, which only represent a permutation of the original ones, have been produced. Since all the partitions differ only in cluster labeling, they can be considered identical in terms of clustering accuracy, thus just a relabeling operation is necessary to find a common agreement. In the state of the art clustering ensemble techniques, a consensus operation is in place and works exchanging all the datapoints between every learner to find an agreement on the label. In the V-DEC approach, each agent has already evaluated all the centroids, thus there is no need to exchange other information. The agreement is achieved by applying the initial clustering algorithm (e.g. K-means) to the centroids. After this step, the labels are reassigned accordingly.

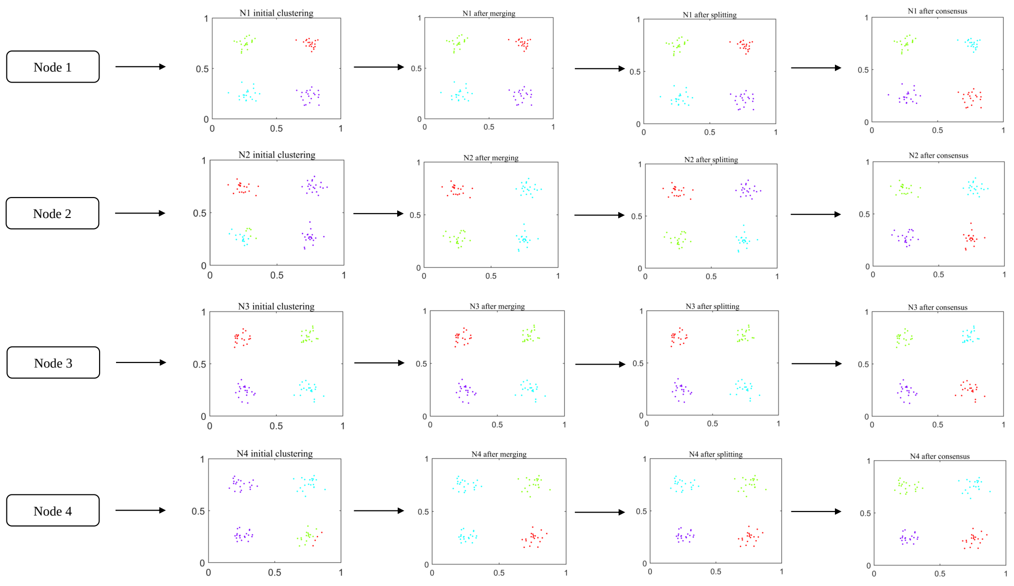


Figure 4.6. Syntethic description of the V-DEC steps

To better converge to a unified result, for results evaluation purpose only, the relabeling is done incorporating the real labels of the clusters inside the analysis. This is further explained in the results section. A visual representation should be seen in Fig. 4.5 where all the clusters are correctly labeled. The figure shows how the procedure is suitable for a purely distributed environment where each agent has to evaluate the labels locally.

A summary representation of the results of each step is given in fig. 4.6 where, starting from the initial clustering results, the V-DEC operators are able to gradually improve the solution by achieving, at the end of the procedure, the same global solution that is obtained by a centralized approach.

Despite its simplicity, the V-DEC results in an interesting algorithm. It is easy to implement, and it achieves a very low error. The overall scheme of the algorithm is summarized in the flow chart 1.

4.3 Validation procedure

This section presents some experimental results to show the performance of the V-DEC algorithm and to evaluate its behavior when is compared with a fully centralized approach. Four different available datasets of the UCI repository are used to test it in different distributed scenarios.

A schematic description of them can be found in Table 4.1, while some additional details are provided as follows:

- *Iris Data Set* [56] is a classification dataset, composed by 150 instances for each of the three classes. The input is given by 4 features.

- *Wine Data Set* [57] where the task is to identify the correct type of wine among 3 classes thanks to 178 instances composed by 13 features.

Algorithm 1 V-DEC Algorithm

```
1: **Initial Clustering**
2:    Let $R = \{R^{ik}\}_{1 \leq i \leq m}$ as the initial set of clustering.
3: **Collaboration phase**
4:    Conflict Detection
5:       $k$ = conflicts(R) with $K_k^{i,j} = 1 - S(C_i^k, CC(C_i^k, R_j))$
6:       where $S(C_i^k, C_j^l) = \max\{S(C_i^k, C_j^m), \forall m \subseteq [1, n_j]$
7:    Conflict Resolution
8:       $K_k^{i,j} = \text{argmax}_{K_l^{r,s}}\ Cl(K_l^{r,s})$ and $CC(C_i^k, R_j) = C_j^l$
9:    Local resolution of conflict
10:      **If** $k <$ threshold **then** $R^{'} = R^{'} \setminus k \cup$ merge(k,$R^j$)
11:      **else** $R^{'} = R^{'}\{C_k^j \cup$ split($C_k^j$, $|k|$ )
12:        **If** {Validity index($R^{'}$) > Validity index($R$)} **then:** $R = R^{'}$
13:        **End if**
14:      **End if**
15:   **Additional refinement steps**
16:   **Consensus computation**.
```

- *Ionosphere Data Set* [58] consists of 16 high-frequency antennas and 17 pulse number for the Goose Bay system. Instances are described by 2 attributes per pulse number. It is a binary classification dataset, where each learner gets 87 datapoints with 10 attributes each.

In all cases, input variables are normalized between 0 and 1, and missing values are replaced with the average computed over the rest of the dataset. As explained before, the datasets have been partitioned to test the algorithm in a purely distributed scenario, differentiating from [52] where all the dataset is given to each node. The number of attributes is different from dataset to dataset, but is always greater than three. The performances are tested on three different techniques where the same set of runs and parameters are used:

- **Centralized Algorithm**;
- **Samarah Algorithm** that is part of the Cluster Ensemble techniques presented in [52];
- **V-DEC**: Validated Distributed Ensemble Clustering.

Before starting with the trials, some preliminary assumptions have to been performed. In the first step of the V-DEC algorithm the K-means, with a random initialization, is used as initial clustering approach. The EM algorithm is also used to compare the results and to see how much the final result is affected by the use of different local clustering algorithms.

The threshold for the merge and split criterion is fixed to 0.5, this means that two clusters are merged only if more than half of the elements are in conflict. This is

Dataset	Features	Instances	Classes	Task
Iris Data Set	4	150	3	Classification
Wine Data Set	13	178	3	Classification
Ionosphere Data Set	34	351	2	Classification

Table 4.1. Description of the datasets

the safest value because it avoids accidental merging of clusters that become hard to be split in the sequent phase. For each test, 100 runs are performed and the average value is inserted in the results.

As stated in the Consensus section, to be able to evaluate the quality of the final clustering results, it is necessary that the labels of each learner are all coherent. To achieve that, a data-driven clustering is applied to all the centroids. The results' centroids are gathered in a matrix, clustered via K-Means and, taking into account the real clusters' centroids, relabeled. In this way, it will be possible to compute quality indexes to evaluate the performance. To this end, four quality indexes could be used:

- Rand Index ([59])
- Fowlkes–Mallows index ([60])
- F-measure ([60])
- K-Index ([61])

An accurate description of each one is obtained in the Appendix A.2

4.3.1 Results analysis

In Tab.4.2 the V-DEC approach is compared with the cluster ensemble techniques and the centralized approach in terms of quality indexes. Best results for each dataset are reported in bold. As it can be seen, the performance of the proposed approach is slightly less good when is compared with that obtained by the ensemble clustering techniques. This is coherent with this analysis, because they use the entire dataset at each node, while the proposed algorithm forces each agent in having a partial and different vision for each dataset to work in a purely distributed scenario. It is important to underline that, despite these constraints, the performances of the V-DEC procedure are comparable with the centralized approach. Additionally, to have a more simple representation of the results, a graphical visualization of the indexes can be obtained in Figures. 4.7a, 4.7c, 4.7b.

Once ascertained that the presented approach is comparable to a fully centralized one, it has been tested how happens when the topology of the network changes, or the used clustering algorithms changes. To this end, table 4.3 compares the three external validity indexes when the number of agents in the network is increased, as well as when the initial classification algorithm is changed. Best results for each initial configuration are highlighted in bold. As it can be seen, results are quite

Dataset	Algorithm	Rand Index	F-Measure	F-M Index
Iris	C	0.73 (± 0.03)	0.64 (± 0.02)	0.60 (± 0.01)
	S	**0.85 (± 0.00)**	**0.78 (± 0.00)**	**0.78 (± 0.01)**
	V-DEC	0.76 (± 0.05)	0.65 (± 0.07)	0.59 (± 0.02)
Wine	C	0.68 (± 0.09)	0.66 (± 0.03)	0.62 (± 0.04)
	S	**0.88 (± 0.04)**	**0.83 (± 0.06)**	**0.83 (± 0.06)**
	V-DEC	0.75 (± 0.06)	0.71 (± 0.08)	0.70 (± 0.05)
Ionosphere	C	0.56 (± 0.01)	0.53 (± 0.03)	0.50 (± 0.04)
	S	0.59 (± 0.03)	0.53 (±0.07)	**0.50 (± 0.09)**
	V-DEC	**0.71 (± 0.02)**	**0.58 (± 0.04)**	0.49 (± 0.05)

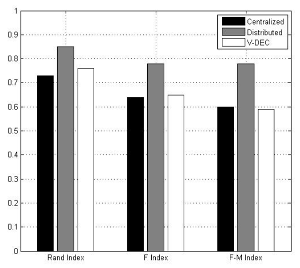
Table 4.2. Cluster quality Indexes for the Centralized, the ensemble clustering approach and the V-DEC algorithm.

insensitive to the increasing of the complexity of the network, while the initial algorithm has more impact in the final values.

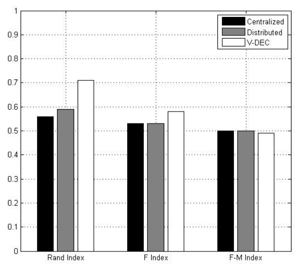
4.3.2 Discussion

This chapter presents an extension of the traditional centralized cluster ensemble technique to the case of a fully distributed scenario. The data is partitioned and collected over a network of agents. As it can be seen from the previous results, the V-DEC procedure is slightly less good in respect to the traditional cluster ensemble technique, (this is coherent with this analysis, because they use the entire dataset at each node, while the V-DEC procedure forces each agent in having a partial and different vision for each dataset) reaching similar performance with a fully centralized implementation.

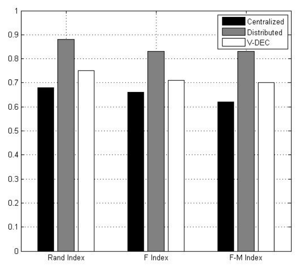
On the other hand, only the local representatives are exchanged on the network, increasing the efficiency and the speed of the procedure.



(a) Iris Dataset



(b) Ionosphere Dataset



(c) Wine Dataset

Figure 4.7. Quality Indexes comparison for each dataset

Dataset	Algorithm	N learners	FM	RI	KI
Iris	K-Means	1	0.786 ± 0.065	0.845 ± 0.066	**0.737 ± 0.196**
	EM	1	**0.827 ± 0.110**	**0.859 ± 0.101**	0.711 ± 0.260
	K-Means	4	**0.761 ± 0.055**	**0.828 ± 0.062**	**0.661 ± 0.020**
	EM	4	0.676 ± 0.069	0.758 ± 0.050	0.585 ± 0.121
Wine	K-Means	1	0.429 ± 0.007	0.594 ± 0.001	0.226 ± 0.003
	EM	1	**0.534 ± 0.062**	**0.629 ± 0.073**	**0.352 ± 0.119**
	K-Means	4	**0.453 ± 0.037**	**0.570 ± 0.050**	0.231 ± 0.062
	EM	4	0.441 ± 0.036	0.570 ± 0.033	**0.236 ± 0.063**
Vehicle	K-Means	1	0.367 ± 0.018	0.637 ± 0.021	**0.262 ± 0.012**
	EM	1	**0.377 ± 0.025**	**0.660 ± 0.020**	0.245 ± 0.040
	K-Means	4	**0.369 ± 0.025**	**0.616 ± 0.039**	**0.234 ± 0.033**
	EM	4	0.356 ± 0.027	0.615 ± 0.0302	0.199 ± 0.032
	K-Means	10	**0.366 ± 0.025**	0.617 ± 0.031	**0.226 ± 0.027**
	EM	10	0.342 ± 0.024	**0.624 ± 0.022**	0.193 ± 0.028
Ionosphere	EM	1	**0.685 ± 0.059**	**0.613 ± 0.090**	0.319 ± 0.305
	K-Means	1	0.605 ± 0.000	0.589 ± 0.000	**0.411 ± 0.000**
	EM	4	**0.620 ± 0.067**	0.533 ± 0.021	0.089 ± 0.114
	K-Means	4	0.599 ± 0.211	**0.581 ± 0.020**	**0.389 ± 0.470**
	EM	10	**0.612 ± 0.060**	0.532 ± 0.018	0.107 ± 0.093
	K-Means	10	0.596 ± 0.296	**0.573 ± 0.024**	**0.342 ± 0.091**

Table 4.3. Cluster quality indexes for K-Means and EM initialization over different initial network configurations.

Chapter 5

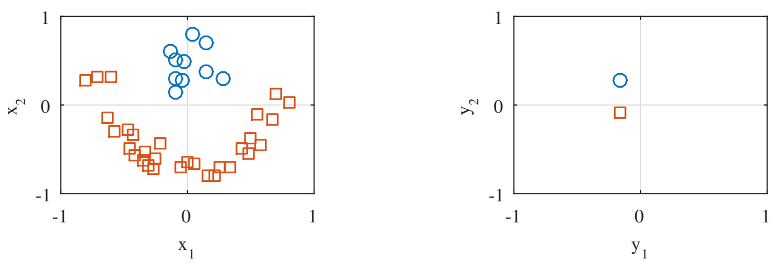
Distributed spectral clustering

This chapter starts from the observation that in data mining application, like the one introduced in the previous chapter, the information is encoded in the similarity matrix between points. Thus in this chapter, an algorithm able to reconstruct (in a decentralized fashion) the matrix of Euclidean distances (EDM) among all points is introduced. In the following section, the procedure will be applied on the particularly spectral clustering technique, but obviously it can be extended on all of the approaches that make use of a similarity matrix. First of all, a basic introduction to the spectral clustering technique is provided. Next, it is extended in a distributed context by the use of an approach based on the gradient procedure.

5.1 Introduction

Performing inference on data which is partitioned over geographically distinct locations is now considered as a fundamental problem in many scientific endeavors. Over the last years, different authors have proposed distributed variants for any of the standard supervised and unsupervised algorithms available in the machine learning community. The algorithm proposed in this chapter starts from the observation that in most of the data mining methods the information is encoded in the matrix D of pairwise distances between patterns. For example, the kernel method or the regularized terms are essentially based on the information related to the similarity between points. In this chapter, the focus is on the spectral clustering, which is also based on this matrix, since it works by performing clustering in a suitably transformed space, whose mapping is constructed starting from a similarity graph corresponding to the data. Different ways of constructing the graph (and extracting its eigenstructure) are possible, but essentially the problem still remains the same: the procedure depends on the pairwise distances among all points, and to the best of our knowledge no distributed protocol for its solution has been proposed. Most of the research has focused in parallelizing its computation with a single coordinating agent [62, 63].

In a distributed setting, each agent is able to obtain the matrix by using its own training data, while the information about the distance between points belonging to different agents is unknown. To obtain this information, in this chapter it is introduced the task of reconstruction (in a decentralized fashion) the matrix of



(a) Original data **(b)** After transformation

Figure 5.1. An example of eigen-transformation on a 2D dataset. (a) 40 points from the original data. (b) First 2 eigenvectors of Laplacian matrix computed from a similarity graph with Euclidean distance on edges. All patterns belonging to a cluster are mapped to the same point.

Euclidean distances (EDM) among all points. Indeed, perfect knowledge of this matrix would allow each agent to solve independently the original spectral clustering problem, for a wide range of different choices of the underlying data graph [64]. Recasting the problem in this way, however, allows us to leverage over a large number of works on matrix completion [65, 66] and EDM completion [67], especially in the distributed setting [68, 69]. Particularly, a distributed gradient-descent algorithm, originally proposed for semi-supervised learning over networks [69] is considered. This is an iterative procedure, where at each step every agent performs a single gradient descent step on its own estimate, followed by an averaging step with respect to its neighbors' estimates. This kind of techniques have a long history in the optimization field [70], and they have recently gained a wide popularity under the name of 'diffusion' strategies [71, 72]. The proposed algorithm reduces the computational complexity by exploiting the specific structure of the EDM matrices, which allows it to operate on a suitable factorization of the original matrix in order to reduce the number of parameters to be estimated.

As previously said, the proposed algorithm is the first truly decentralized procedure for performing spectral clustering, which respects the constraints put forward before, particularly:

- it only considers local communication among neighbors;
- update steps can be performed easily even on low-power devices;
- no coordinating entity is required, such that every node has the same importance in the overall network.

5.2 Spectral clustering

Before analyzing the distributed environment, the basic concepts pertaining to the spectral clustering theory are briefly introduced [73, 64].

Suppose it is given a set of N points $S = \{\mathbf{x}_i\}_{i=1}^{N}$, where each $\mathbf{x}_i \in \mathbb{R}^d$. The aim is to partition the set S into k disjoint clusters, where k is given beforehand, such that a predefined measure of quality is optimized (e.g. the ones described in Appendix

A). The first step of any spectral clustering algorithm consists in constructing a *data-adjacency* graph $\mathcal{G}^D = \left(\mathcal{V}^D, \mathcal{E}^D\right)$,[1] where each vertex in $\mathcal{V}^D$ corresponds to a point in S, and each edge in $\mathcal{E}^D$ weights the similarity among two points. An adjacency matrix $\mathbf{W}^D \in \mathbb{R}^{N\times N}$ is also defined, where w_{ij}^D is the similarity among points $\mathbf{x}_i$ and $\mathbf{x}_j$. There are many ways of constructing $\mathbf{W}^D$, corresponding to different variants of frameworks and part of them is summarized as follows:

- **The $\epsilon-$ neighborhood graph**, where all of the points with a distance smaller than ϵ are connected. At the end of the approach, all of the points obtain a distance of the same scale (at most ϵ) so, the weighting of the edges will not incorporate more information and the graph will be usually considered unweighted.
- **The fully connected graph** all of the points with positive similarity are connected each other. This model is useful if the similarity function itself models local neighborhoods. As an example, a simple choice is a fully connected graph based on a Gaussian weighting with bandwidth $\gamma > 0$:

$$w_{ij}^D = \exp\left\{-\gamma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2^2\right\} . \tag{5.1}$$

  The parameter γ plays a similar role as the parameter ϵ in case of the ϵ-neighborhood graph.

Although there are multiple choices for $\mathbf{W}^D$, it is important to stress here that anyone of them practically depends on the Euclidean distance between points, an aspect which is fundamental in the proposed distributed strategy's design.

Once the weighting matrix $\mathbf{W}^D$ is specified, the second step of the spectral clustering consists in computing the Laplacian adjacency matrix, which is defined as:

$$\mathbf{L}^D = \mathbf{D}^D - \mathbf{W}^D , \tag{5.2}$$

where $\mathbf{D}$ is the degree matrix. Similarly to what happens with $\mathbf{W}^D$, there are many variants for constructing $\mathbf{L}^D$, such as normalizing it by the degree matrix or using an iterated version [64].

The third step of the spectral clustering procedure extracts the leading k eigenvectors (where k is the number of desired clusters) from $\mathbf{L}^D$. This operation could be implemented efficiently, particularly if the elements of $\mathbf{W}^D$ are symmetric [74].

Suppose that the eigenvectors are stacked column-wise in a matrix $\mathbf{U}$. The ith row of $\mathbf{U}$ can be interpreted as the transformation of $\mathbf{x}_i$ in the space induced by the eigenstructure of $\mathbf{L}^D$. Based on this consideration, the last step of SC algorithms is to cluster in k groups the rows of $\mathbf{L}^D$, typically by using the standard k-means procedure. A schematic description of the spectral clustering procedure can be given in the Algorithm: 2.

The rationale for this is that clustering in this new space is generally simpler since it exploits *a priori* the similarity information contained in the adjacency graph (similar ideas can be found in other areas of machine learning, such as semi-supervised

[1]A superscript D is used to differentiate it from the agents' graph introduced in the next section.

Algorithm 2 Unnormalized spectral clustering

```
1: Input : Let $K$ as the number of clusters and $S \in \mathbb{R}^{nxn}$
2: Construct a similarity graph by one of the way described in the previous section
   and let $W$ be its weighted adjacency matrix.
3: Compute the unnormalized Laplacian $L$.
4: Compute the first $k$ eigenvectors $u_1, \dots, u_k$ as columns.
5: for  $i = 1$ to $n$ do
6:     Let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i$-th row of $U$.
7: end for
8:     Cluster the points $(y_i)_{i=1,\dots,n \text{ in } \mathbb{R}^k}$ with the $k$-means algorithm into clusters
   $C_1, \dots, C_k$
9: Output: Clusters $A_1, \dots, A_k$ with $A_i = \{j | y_j \in C_i\}$
```

learning with manifold regularization [75]). As an extreme case, consider the toy problem presented in Fig. 5.1. The original 2-dimensional points in Fig. 5.1a are organized in two optimal clusters, one of which is non-trivial to capture with a standard k-means, which assumes hypersphericity of the clusters. However, in the new domain induced by $\mathbf{U}$ (see Fig. 5.1b), all the patterns belonging to a cluster are mapped to a single point.

5.3 Distributed Spectral Clustering over Networks

5.3.1 Formulation of the problem

For the rest of this work, it is assumed that the dataset S to be clustered is not available in a centralized location and that the following assumptions are needed:

Assumption 5.1. The dataset is partitioned over L agents, such that the kth agent has access to a dataset S_k and the union of each local dataset gives us the original one:

$$\bigcup_{k=1}^{L} S_k = S \tag{5.3}$$

Assumption 5.2. The connectivity among agents can be represented as a second graph, that in this work is called the *agents*' graph, $\mathcal{G}^A = \left(\mathcal{V}^A, \mathcal{E}^A\right)$, such that:

- each vertex $v \in \mathcal{V}^A$ is an agent;
- two agents i and j can communicate (i.e. exchange information) between them only if the edge (i, j) is in the set $\mathcal{E}^A$.

Assumption 5.3. The agents' graph is assumed to be:

- time-invariant;
- connected (i.e., two agents are always reachable by traversing a finite amount of edges);

- undirected, such that if $(i,j) \in \mathcal{E}^A$, then $(j,i) \in \mathcal{E}^A$.

Additionally, let us supposed that a mechanism of synchronization for performing iterative computations is available for each node. This is a standard assumption in most of the literature on distributed learning [76, 77, 78].

From what said in the previous section, it is clear that the distributed spectral clustering problem is equivalent to the distributed computation of the matrix $\mathbf{L}^D$, which in turn is equivalent to the computation of the EDM $\mathbf{E}$. In the following section, an efficient distributed gradient algorithm is proposed to this end.

5.3.2 Distributed computation of the EDM

In the proposed algorithm is presented the problem of EDM completion [79]. Generally, the matrix completion problem could be defined as the problem of recovering the missing entries of a matrix only from a set of known entries [65]. It has many practical applications, as well as sensors localization, covariance estimation, and customer recommendations.

Generally, the problem is the one of recovering the missing entries of a distance matrix when the dimension of the data unknown is generally smaller compared to the number of considered data points.

Let $\mathbf{E}_k$ be the EDM computed only from the patterns in S_k; we are interested in estimating, in a totally decentralized fashion, the global EDM with respect to all the N patterns. To begin with, we note that with a proper rearrangement of patterns, the global EDM $\mathbf{E}$ can always be expressed as:

$$\mathbf{E} = \begin{bmatrix} \mathbf{E}_1 & ? & ? \\ ? & \ddots & ? \\ ? & ? & \mathbf{E}_L \end{bmatrix}, \tag{5.4}$$

This particular structure implies that the sampling set is not random, and makes non-trivial the problem of completing $\mathbf{E}$ solely from the knowledge of the local matrices. At the opposite, the idea of exchanging the entire local datasets between nodes is unfeasible because of the amount of data which would need to be shared.

Based on these considerations, it is proposed a framework for the distributed estimation of $\mathbf{E}_k$, which consists of four steps:

1. **Patterns exchange**: every agent exchanges a fraction p of the available S_k with its neighbors.

   This step is necessary so that the agents can increase the number of known entries in their local matrices. In order to maximize the diffusion of the data within the network, this step is iterated $n_{\max}^{(1)}$ times; at every iteration, an increasing percentage of shared data is constituted by pattern received by the neighbors in previous iterations. A simple strategy to do this consists, at the iteration n, to choose $\frac{n_{\max}^{(1)}-n+1}{n_{\max}^{(1)}}p$ patterns from the local dataset, and $\frac{n-1}{n_{\max}^{(1)}}p$ patterns received in the previous $n-1$ iterations. In order to preserve privacy, this step can include one of the privacy-preserving strategies known in the literature [80] (it is further detailed this in Sec.8.4).

2. **Local EDM computation**: each agent computes, using its original dataset and the data received from its neighbors, an incomplete approximation $\hat{\mathbf{E}}_k \in \mathbb{R}^{N\times N}$ of the real EDM matrix $\mathbf{E}$.

3. **Entries exchange**: the agents exchange a sample of their local EDMs $\hat{\mathbf{E}}_k$ with their neighbors. Again, this step is iterated $n^{(2)}_{\max}$ times using the rule of step 1. In this case, $n^{(2)}_{\max}$ denotes the maximum number of iterations.

4. **Distributed EDM completion**: the agents complete the estimate $\tilde{\mathbf{E}}$ of the global EDM using the strategy detailed next.

5.3.3 Diffusion gradient descent

The last step of the algorithm is concerning with the distributed EDM completion. The idea is to complete the square matrix $\tilde{\mathbf{E}}$ containing the pairwise distances among the training patterns:

$$\tilde{E}_{ij} = \left\|\mathbf{x_i} - \mathbf{x_j}\right\|_2^2, \forall i,j = 1, \dots, N \tag{5.5}$$

Definition 5.4. Generally the Euclidean distance matrix satisfies the following conditions:

- the matrix is symmetric;
- $\tilde{E}_{ij} = 0$ for all the elements on the main diagonal;
- it entries are not-negative and satisfy the triangle inequality;
- it is rank deficient. It is possible to show that the rank of $\tilde{\mathbf{E}}$ is upper bounded by $d+2$ (and the rank is generally $d+2$), so in all practical applications it is very small when is compared to the number of data points ($d \ll N$).

By exploiting this property, it can be concluded that the full Euclidean distance matrix from a restrictive set of given distances can be evaluated since the low-rank property assures that there is redundancy between the available data.

Assumption 5.5. In the following, let us suppose to have observed only a subset of entries of $\tilde{\mathbf{E}}$, in the form of $\hat{\mathbf{E}}$. More formally, this can be formalized this in the following way:

$$\tilde{\mathbf{E}} = \begin{cases} \hat{E}_{ij} = \tilde{E}_{ij} & \text{if } \Omega_{ij} = 1 \\ \hat{E}_{ij} = 0 & \text{otherwise} \end{cases}. \tag{5.6}$$

where the local matrix $\mathbf{\Omega}_k$ is:

$$\mathbf{\Omega}_k = \begin{cases} 1 & \text{if } \hat{E}_{ij} \neq 0 \\ 0 & \text{otherwise} \end{cases}. \tag{5.7}$$

Starting from this formulation the aim is to recover the original matrix $\tilde{\mathbf{E}}$ from $\hat{\mathbf{E}}$ by minimizing an optimization problem. To this end, an associating with each agent k to a twice-differentiable cost function $J_k(\tilde{\mathbf{E}})$ is need to seek the unique minimizer of the aggregate cost function $J^{glob}(\tilde{\mathbf{E}})$ for finding a matrix $\tilde{\mathbf{E}}$.

Definition 5.6. The specified (joint) cost function to be minimized can be expressed as:

$$\min_{\tilde{\mathbf{E}} \in \mathrm{EDM}(N)} \sum_{k=1}^{L} J_k(\tilde{\mathbf{E}}) = \sum_{k=1}^{L} \left\| \mathbf{\Omega}_k \circ \left(\hat{\mathbf{E}}_k - \tilde{\mathbf{E}} \right) \right\|_F^2 , \tag{5.8}$$

where EDM(N) is the set of EDMs of size $N \times N$, $|\mathbf{A}\|_{\mathcal{F}}$ is the Frobenius norm of matrix $\mathbf{A}$ and $\circ$ denotes the Hadamard product.[2]

The problem in (5.8) is known to be NP-hard in general, but convex relaxations can be realized in order to render the problem tractable. A convenient alternative formulation is to cast it into an optimization one on the set of positive semidefinite matrices. In particular, it is exploit the algorithm introduced in [69], which in turn derives from the framework of diffusion adaptation (DA) for optimization [77] and on previous works on EDM completion [67]. In this way, the objective function can be formulated only in terms of the low-rank factor $\mathbf{V}$, strongly reducing the computational cost of the realized algorithm:

Definition 5.7. The problem in Eq. (5.8) can be reformulated in the following way:

$$J_k(\mathbf{V}) = \left\| \mathbf{\Omega}_k \circ \left[\hat{\mathbf{E}}_k - \kappa\left(\mathbf{V}\mathbf{V}^{\mathrm{T}} \right) \right] \right\|_{\mathrm{F}}^2 \quad k = 1, \ldots, L , \tag{5.10}$$

where $\kappa(\cdot)$ is the Schoenberg mapping, which maps every positive semidefinite (PSD) matrix to an EDM, given by:

$$\kappa(\mathbf{E}) = \mathrm{diag}(\mathbf{E})\mathbf{1}^{\mathrm{T}} + \mathbf{1}\mathrm{diag}(\mathbf{E})^{\mathrm{T}} - 2\mathbf{E} , \tag{5.11}$$

such that diag($\mathbf{E}$) extracts the main diagonal of $\mathbf{E}$ as a column vector, and it is also exploited the known fact that any PSD matrix $\mathbf{D}$ with rank r admits a factorization $\{\mathbf{D} = \mathbf{V}\mathbf{V}^{\mathrm{T}}\}$, where $\mathbf{V} \in \mathbb{R}_*^{N \times r} = \{\mathbf{V} \in \mathbb{R}^{N \times r} : \det(\mathbf{V}^{\mathrm{T}}\mathbf{V}) \neq 0\}$.

A practical advantage of this formulation is that the rank of V identifies the dimension of the embedding space. In this way, the problem becomes to find out a formulation that can obtain the low-rank factor $\mathbf{V}$ in a decentralized and online manner. There are several possible ways by which the distributed strategies can be used to seek the minimizer of Eq. (5.8). In this approach, it will be supposed that at every iteration, are executed two operations. The first is an updating, where agent k combine the existing iterates from its neighbors to obtain an intermediate iterate. All of the other agents are simultaneously performing a similar step. In particular,

[2] In particular, let us suppose to have two matrices $\mathbf{A}, \mathbf{B}$ of the same dimension nxp, then the Hadamard product $\mathbf{A} \circ \mathbf{B}$ is a matrix of the same dimension as the operands, evaluated in the following way:

$$(\mathbf{A} \circ \mathbf{B})_{i,j} = (A)_{i,j}(B)_{i,j} \tag{5.9}$$

For matrices of different dimensions (nxp ans mxq, where $n \neq m$ and $p \neq q$) the Hadamard product is undefined.

the intermediate estimate is evaluated by using a gradient descent algorithm on the smooth cost function.

The second operation is a diffusion step, where agent k approximates the vector $\mathbf{V}$ and uses it to update its intermediate iterate. Again, all of the other agents in the network are simultaneously performing a similar information exchange step. The

Algorithm 3 Pseudocode of the proposed distributed spectral clustering algorithm, at the kth agent.

```
1:  Input : Local dataset $S_k$, number of nodes $L$ (global), maximum number of iterations $T$.
2:  for $n = 1$ to $n_{\max}^{(1)}$ do
3:      Select a set of input patterns and share them with the neighbors $\mathcal{N}_k$.
4:      Receive patterns from the neighbors.
5:  end for
6:  Compute the incomplete EDM matrix $\hat{\mathbf{E}}_k$ $n = 1$ to $n_{\max}^{(2)}$
7:  for  $n = 1$ to $n_{\max}^{(2)}$ do
8:      Select a set of entries from $\hat{\mathbf{E}}_k$ and share them with the neighbors.
9:      Receive entries from the neighbors.
10:     Update $\hat{\mathbf{E}}_k$ with the entries received.
11: end for
12: Initialize $\mathbf{V}_k[0]$.
13: for  $n = 1$ to $T$ do
14:     Compute $\mathbf{V}_k[n]$ using Eq. (5.12).
15:     Diffuse local information using Eq. (5.14). end for
16: Compute the Laplacian matrix $\tilde{\mathbf{L}}$ from $\tilde{\mathbf{E}}$.
17: Perform spectral clustering using $\tilde{\mathbf{L}}$.
```

distributed completion of the EDM is thus obtained by combining the following steps, and it is generally defined as an alternation of updating and diffusion equations, in the form of [69]:

1. **Initialization**: All the agents initialize the local matrices $\mathbf{V}_k$ as random $N \times r$ matrices.

2. **Updating of V**: At time n, the kth agent updates the local matrix $\mathbf{V}_k$ using a gradient descent step with respect to its local cost function:

$$\tilde{\mathbf{V}}_k\left[n+1\right] = \mathbf{V}_k[n] - \eta_k[n]\nabla_{\mathbf{V}_k} J_k(\mathbf{V})\,. \tag{5.12}$$

where $\eta_k\left[n\right]$ is a positive step-size. It is straightforward to show that the gradient of the cost function is given by:

$$\nabla_{\mathbf{V}_k} J_k(\mathbf{V}) = \kappa^*\Big\{\boldsymbol{\Omega}_k \circ \\ \circ \Big(\kappa\Big(\mathbf{V}_k\left[n\right]\mathbf{V}_k^{\mathrm{T}}\left[n\right]\Big) - \hat{\mathbf{E}}_k\Big)\Big\}\mathbf{V}_k\left[n\right]\,, \tag{5.13}$$

where $\kappa^*(\mathbf{A}) = 2\left[\operatorname{diag}(\mathbf{A}1\right] - \mathbf{A})$ is the adjoint operator of $\kappa(\cdot)$.

3. **Diffusion**: In order to propagate information over the network, the updated matrices are combined according to some mixing weights $\mathbf{C} \in \mathbb{R}^{L\times L}$:

$$\mathbf{V}_k\left[n+1\right] = \sum_{i=1}^{L} C_{ki}\tilde{\mathbf{V}}_i\left[n+1\right]. \qquad (5.14)$$

where $C_{ki} > 0$ if and only if agents k and i are connected, in order to send information only through neighbors. The mixing weights are generally chosen to provide a convex combination at every agent. A simple choice, which is used throughout the experimental results, are the so-called 'max-degree' weights (3.27).

This approach has two main advantages. First, it is able to take into account naturally the properties of EDM matrices. Secondly, at every step, each node has a complete estimate of the overall matrix, instead of a single column-wise block. Thus, there is no need of gathering the overall matrix at the end of the optimization process. For a rationale of this approach and an analysis of its convergence behavior in the case of convex cost functions, it is possible refer to any introductory publication on DA [72, 77]. Convergence of a similar family of algorithms in the case of non-convex cost functions is instead derived in [45]. The proposed algorithm for distributed spectral clustering over networks is summarized in Algorithm 3.

5.4 Experimental Validity

5.4.1 Experimental Setup

This section evaluates the performance of the proposed algorithm when is compared with a fully centralized approach, where local data is sent beforehand to a (virtual) centralized processor. Six different public datasets available on the UCI repository [81] are considered; a schematic description of which is given in Table 5.1. In all

Dataset	Features	Instances	Classes	Original Task
Australian credit approval	14	690	2	Classification
Hill-Valley	100	606	2	Classification
Ionosphere	34	351	2	Classification
LSVT Voice Rehabilitation	309	126	2	Classification
Twomoons	2	400	2	Clustering
Vehicle Silhouettes	18	752	4	Classification

Table 5.1. Detailed description of each Dataset. Additional information on them is provided in Section 5.4.1

cases, the optimal clustering is known beforehand for testing purposes, either in the case of classification datasets (where clusters correspond to classes) or because the

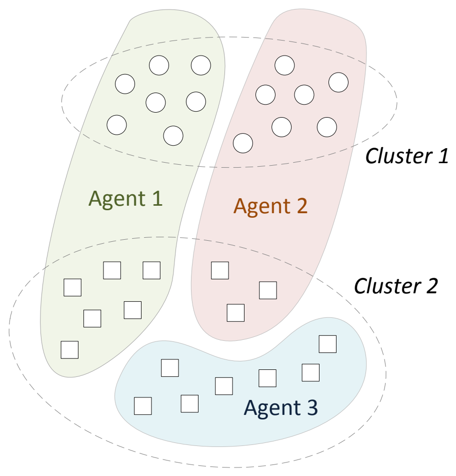


Figure 5.2. Example of distributed clustering over a network, involving two clusters (denoted by circles and squares, respectively), and three agents. Every agent has a varying number of points, and it may not have representatives from each of the clusters.

dataset is artificially generated. In the following, some additional information on each of them is presented.

- *Australian credit approval* was already introduced in Sec. 7.5.2.
- *Hill-Valley* dataset has the task to identify whenever the plotted points will create a Hill (a “bump” in the terrain) or a Valley (a “dip” in the terrain).
- *Ionosphere*: which was already discussed in Section 4.3.1.
- *LSVT Voice Rehabilitation* [82] dataset where the aim is to assess whether voice rehabilitation treatment leads to phonations considered ’acceptable’ or ’unacceptable’. Each feature corresponds to the application of a speech signal processing algorithm which has to characterize objectively the signal.
- *Twomoons* dataset contains two clusters, whose shape is similar to a waxing and waning crescent moon.
- *Vehicle Silhouettes* [83]: it contains four classes of vehicles to be classified with a set of features extracted from their silhouette. The vehicles may be viewed from one of many different angles.

In all cases, input features were normalized between -1 and 1 before the experiments. All experiments are then averaged over 5 different runs of simulations. In each run it is considered a random topology of 7 nodes according to the so-called “Erdos-Rènyi model” [84] such that every pair of nodes is connected with a fixed probability $p = 0.5$. The only global requirement is that the overall topology is connected. At each round the overall dataset is randomly partitioned among the agents (as in Fig. 5.2).The following two algorithms are compared:

- *Distributed Spectral Clustering*: this is the previously described approach where the Laplacian Matrix is evaluated in a distributed fashion using the distributed

EDM completion algorithm presented in Section 5.3.2. For the initial exchange phase, a small value of $n_{\max}^{(1)} = n_{\max}^{(2)} = 150$ is set. The EDM estimation algorithm is run for $T = 1000$ iterations. For the step-size, a fixed step-size strategy is used. In particular, the optimal values of η is chosen searching it in the exponential interval 10^j with $j \in \{-10, -9, ..., 2, 3\}$.

- *Centralized approach*: this is equivalent in having a centralized agent, where the traditional spectral clustering algorithm is performed on the global dataset. It is used as an optimal benchmark to evaluate the proposed approach.

It is used a standard spectral clustering procedure, where the Laplacian matrix is constructed using Eq. (5.1) with $\gamma = 1$, which was found to work relatively well on all datasets.

5.4.2 Results

Dataset	Algorithm	F-Measure	Rand-Index	K-Index	F-M Index
Australian credit approval	C	0.445 ± 0.198	0.499 ± 0.004	0.028 ± 0.014	**0.564 ± 0.041**
	D	**0.473 ± 0.000**	**0.500 ± 0.000**	**0.032 ± 0.000**	0.504 ± 0.000
Hill-Valley	C	0.488 ± 0.178	**0.501 ± 0.001**	**0.042 ± 0.019**	**0.596 ±0.059**
	D	**0.506 ± 0.024**	0.500 ± 0.000	0.029 ± 0.000	0.499 ± 0.000
Ionosphere	C	**0.944 ± 0.000**	**0.811 ± 0.000**	0.046 ± 0.000	**0.655 ± 0.000**
	D	0.641 ± 0.035	0.504 ± 0.000	**0.119 ± 0.000**	0.638 ± 0.000
LSVT Voice Rehabilitation	C	0.418 ± 0.000	**0.509 ± 0.000**	0.091 ± 0.000	**0.542 ± 0.000**
	D	**0.434 ± 0.113**	0.499 ± 0.000	**0.139 ± 0.000**	0.532 ± 0.000
Twomoons	C	0.723 ± 0.361	**0.813 ± 0.000**	**0.792 ± 0.000**	**0.817 ± 0.000**
	D	**0.788 ± 0.254**	0.751 ± 0.000	0.711 ± 0.000	0.752 ± 0.000
Vehicle Silhouettes	C	**0.222 ± 0.139**	0.448 ± 0.036	0.026 ± 0.007	**0.386 ± 0.023**
	D	0.170 ± 0.080	**0.650 ± 0.000**	**0.187 ± 0.000**	0.312 ± 0.000

Table 5.2. Experimental results on the different dataset. The average and the standard deviation of the F-Index, Rand Index, Kappa Index, FM-Index are shown for both the centralized (C) and the distributed (D) approaches. Best results for each algorithm are highlighted in bold.

The goal of the experiment was to evaluate how the proposed distributed EDM algorithm can influence the performance of the external quality indexes introduced in Appendix A.2 when is compared with a centralized algorithm:

- Rand Index [59]
- Falks-Mallows [60]
- F-measure [60]
- K-Index [85]

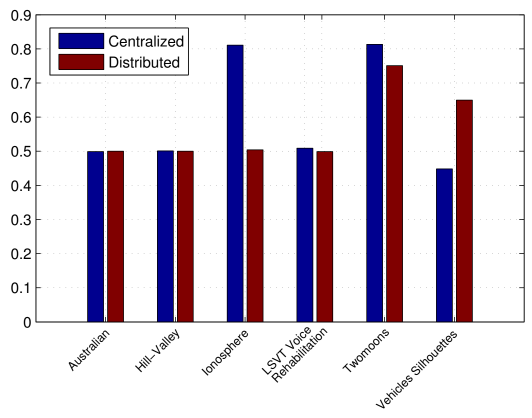


Figure 5.3. The Rand Index for all of the described datasets for both the Centralized Spectral Clustering algorithm and the Distributed approach.

The results are summarized in Table 5.2 where they have been averaged over 10 k-means evaluations and over the different agents in the distributed case. For each dataset, the mean and the standard deviation of each quality index is computed. The best result for each dataset and for each index is highlighted in bold.

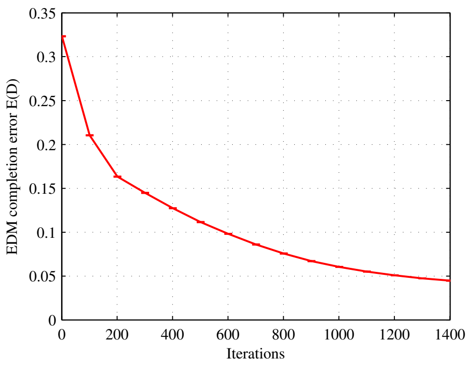
As it can be seen, the results of the two approaches are reasonably aligned. Whereas the centralized approach can significantly boost performance for almost all the quality indexes in the Ionosphere dataset, in the other examples results are more comparable. In particular, in the Australian credit approval, Hill-Valley, LSVT Voice Rehabilitation, and Twomoons datasets the values of the quality indexes are very similar, while in the Vehicle Silhouettes dataset the proposed approach outperforms the centralized one with respect to the F-measure and the F-M Index. The most important result suggested from Table 5.2 is that the distributed EDM computation can indeed be an effective approach to be applied in a distributed scenario since it is able to match very closely the performance of the centralized spectral clustering algorithm.

To further strengthen the results, a visual representation of the Rand-Index is given in Fig 5.3. As it has been noted the two approaches present a similar behavior in the majority of datasets. A slightly worse trend is reported for the Ionosphere dataset, but it is offset by the behavior of the Vehicle Silhouettes dataset that presents an increase of nearly 20% of the Rand Index. To reinforce the conclusion, it is reported a completion error during the iterations of the distributed EDM estimation algorithm. The error is evaluated at each iteration in the following way:

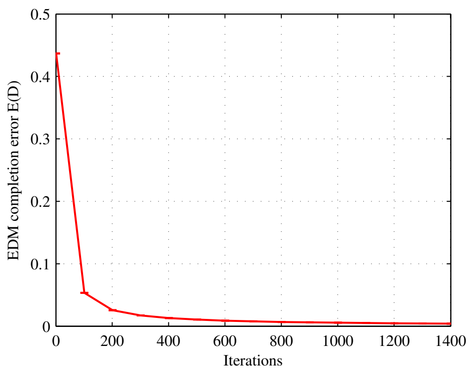
$$\text{Err} = \frac{1}{L}\sum_{k=1}^{L}\frac{\left\|\mathbf{E}-\tilde{\mathbf{E}}_k\right\|}{\|\mathbf{E}\|}\,. \tag{5.15}$$

The convergence of the proposed approach, averaged over the different runs, is summarized in Fig. 5.4, where on the x-axis is reported the iterations number and on the y-axis the EDM completion error.

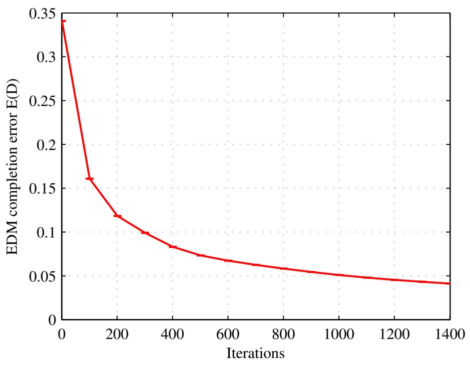
The reported trend proves that the proposed approach is able to rapidly converge to the real EDM. It can be said that, with the sole exception of the Twomoons, the



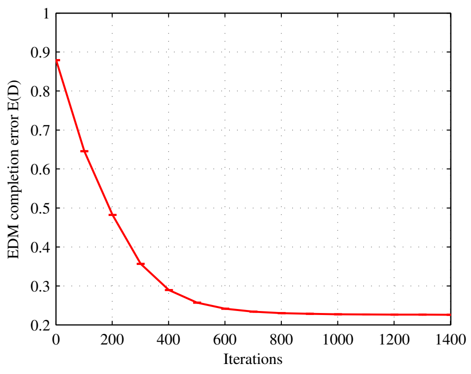
(a) Australian credit approval



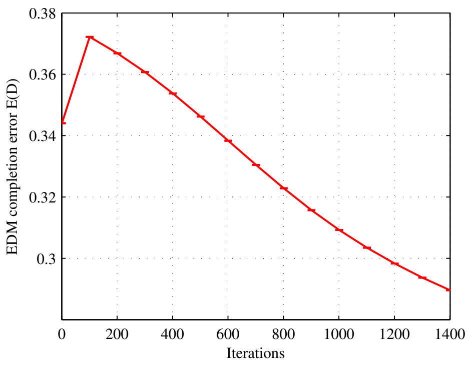
(b) Hill-Valley



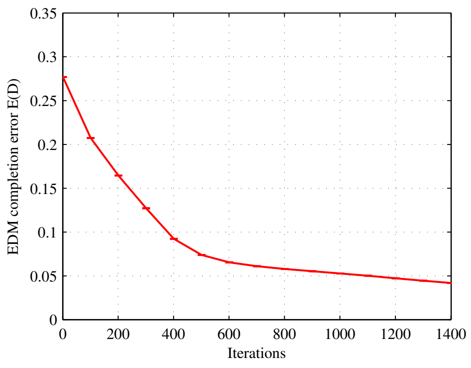
(c) Ionosphere



(d) LSVT Voice Rehabilitation



(e) Twomoons



(f) Vehicle Silhouettes

Figure 5.4. Average EDM completion error of the distributed gradient procedure, at each iteration. The vertical bars represent the standard deviation from the error.

algorithm is able to obtain very similar results in all of the datasets, where in a few number of iterations it obtains very low errors. The error is independent of the size of the dataset, although a low number of steps are necessary to reach convergence to a reasonable accuracy.

5.5 Discussion

This chapter has introduced a fully distributed procedure for performing spectral clustering over networks of computing agents. The proposed approach is able to efficiently match a fully centralized implementation, without however requiring the presence of a coordinating node, and using only in-network communication.

Chapter 6

Distributed Learning for RWFNN

This chapter continues the investigation on distributed training algorithms from a different point of view, adding the possibility to the data to arrive one-by-one or chunk-by-chunk. Specifically, it is introduced an approach for random weight fuzzy neural network, which is based on the Adaptive Neuro-Fuzzy Inference System (ANFIS). The use of these models is widespread in the centralized case, due to their good trade-off of algorithmic simplicity and non linear modeling capabilities. At the same time, their use in the distributed learning setting has not been investigated in depth and this is the main motivation for this chapter. After introducing the Random-Weight Fuzzy Neural Networks, their application in the distributed approach has been described.

6.1 Fundamentals

This section introduces some concepts and results that will be used in the rest of this chapter. First of all, some preliminary information on the Adaptive Neuro Fuzzy Inference System are given, then an explanation on how it can be extended to realize the Random Weight Fuzzy Neural Network (RWFNN) is detailed. Finally, an on-line training procedure is introduced.

6.1.1 Adaptive Neuro Fuzzy Inference System

A little description of the Adaptive Neuro Fuzzy Inference System (ANFIS), introduced by Jang in 1993 [86] is presented in this section. In this technique, the features of Artificial Neural Network (ANN) and the ones of a Fuzzy Inference System (FIS) are linked together to realize a methodology able to model the imprecision and uncertainty of a system by using the adaptability of the learning procedure typical of a neural network. The capability to deal with incomplete, uncertain and hypothetical information, makes it suitable tool for the solution of regression, density estimation, classification and pattern recognition problems [87, 88]. The idea underlying the operating model is very simple. Firstly, the initial fuzzy model is derived by considering the input-output data. In particular, the Takagi Sugeno fuzzy inference system is used for learning the parameters of the C different fuzzy rules, where the kth rule, can be defined in the following way:

Definition 6.1. The kth rule, $k = 1 \dots C$, of a TS system is:

$$\begin{aligned}&\text{If } x_1 \text{ is } B_1^{(k)},\ x_2 \text{ is } B_2^{(k)} \dots \text{ and } x_D \text{ is } B_D^{(k)} \text{ then}\\ &y^{(k)} = h\left(\mathbf{x}; \omega^{(k)}\right)\end{aligned} \tag{6.1}$$

.

where $\mathbf{x} = [x_1\ x_2\ \dots\ x_D]$ is a row vector (or pattern) in the D-dimensional (embedded) input space and $y^{(k)}$ is the scalar output associated with the rule. The latter is characterized by the membership functions (MFs) $\mu_{B_j^{(k)}}(x_j)$ of the fuzzy input variables $B_j^{(k)}$, $j = 1 \dots D$, and the set of parameters $\omega^{(k)}$ of the related crisp output functions in the consequent parts. Several alternatives are possible for the fuzzification of crisp inputs, the composition of input MFs, and the way rule outputs are combined [89].

Once obtained the parameters of the fuzzy inference system, in a successively step, the artificial neural networks are used to tune the rules and produce the final ANFIS model. A general scheme of the ANN is given in Fig. 6.1, where the architecture is composed of one fixed hidden layer and a linear output one. In this particular case, the connections from the hidden layer and the output are fixed, while the ones from the input and the hidden are trained by the network. To produce the output value,

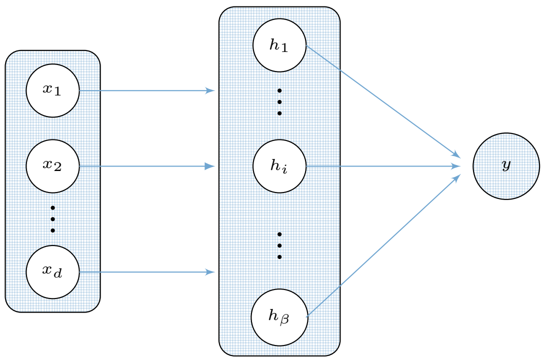


Figure 6.1. Architecture of an ANN

the input parameters must be opportunely combined.

Assumption 6.2. The mathematical model able to link the relationship between the input and the output can be described in the following:

$$f(x) = \sum_{i=1}^{B} \beta_i h_i(\mathbf{x}) = \beta^T \mathbf{h}(\mathbf{x}) \tag{6.2}$$

where $\beta \in \mathbb{R}^B$ and $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}_1), \dots, h_B(\mathbf{x}_B)]^T$.

This is equivalent to a linear model over the transformed vector $h(\mathbf{x})$, and several methods exist to solve this kind of formulation. These models are widespread in different contexts: kernel methods, radial basis functions, wavelet expansions but in this chapter, theirs application on the Random Weight Fuzzy Neural Network are deepen investigated.

6.1.2 Random Weight Fuzzy Neural Networks

This section, the RWFNN [90] is considered, by reformulating the classical approach represented by Adaptive Neuro-Fuzzy Inference System (ANFIS). The RWFNN is an inference system where the fuzzy rule parameters of antecedents (i.e. membership functions) are randomly generated and the ones of consequent are estimated using a regularized least square algorithm.

The structure of an RWFNN consists of 5 feed forward layers and a visual representation can be analyzed in Fig.6.2. Each layer is constituted by a set of nodes and each node is associated with a fuzzy rule. Each node performs a particular operation on the signals coming from the previous layer and sends the result of the calculation to the nodes in the next layer.

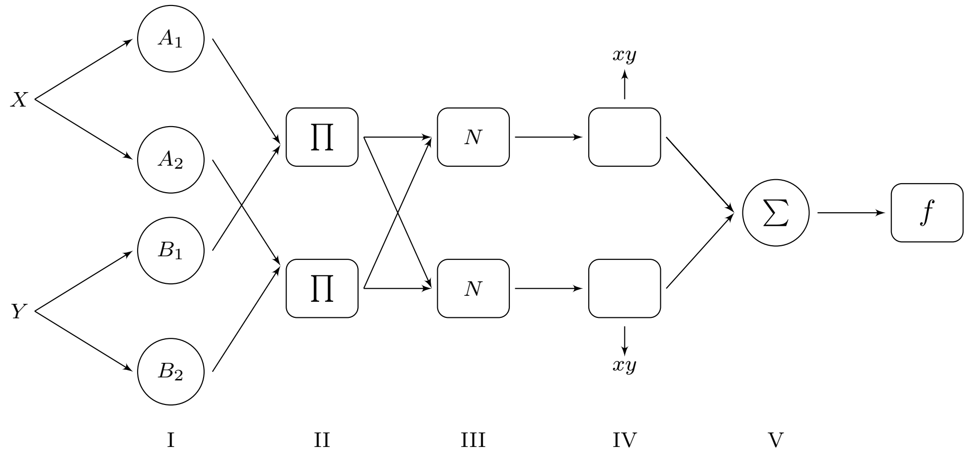


Figure 6.2. A typical ANFIS architecture for a two-input Sugeno model

The aim is estimate a scalar output $y \in \mathbb{R}$ from a d-dimensional input $\mathbf{x} = [x_1, \ldots, x_d]^{\mathrm{T}}$. Each feature of the input is fuzzified and mapped onto a set of membership functions (MFs). Let m be the number of rules of the RWFNN network, then the structure of the fuzzy inference system can be summarized as follows:

- Layer 1. Every node i in this layer, $i = 1 \ldots m$, is associated with an input MF $\mu_{(i,j)}(x_j, \boldsymbol{\alpha}_{(i,j)})$ operating on the jth dimension of the input vector $\mathbf{x}$ for the ith rule. The number and the range of the antecedents' parameters $\boldsymbol{\alpha}$ depend on the type of the MFs. Their values are chosen at the beginning of the learning process from a fixed probability distribution, which is independent of the training data.
- Layer 2. Every node in the second layer corresponds to an *if-then* rule of the FIS. If the adopted operator for the logical AND is the algebraic product, then

the output of the ith node is:

$$w_i(\mathbf{x}) = \prod_{j=1}^{d} \mu_{(i,j)}(x_j),\ i = 1 \dots m. \tag{6.3}$$

- Layer 3. Normalization:

$$\overline{w}_i(\mathbf{x}) = \frac{w_i(\mathbf{x})}{\sum_{h=1}^{m} w_h(\mathbf{x})},\ i = 1 \dots m. \tag{6.4}$$

- Layer 4. Local output of the ith rule:

$$\tilde{f}_i(\mathbf{x}) = \overline{w}_i(\mathbf{x})(\boldsymbol{\beta}_i^{\mathrm{T}}\mathbf{x}^{+}),\ i = 1 \dots m, \tag{6.5}$$

  where $\boldsymbol{\beta}_i$ is the $(d+1)$-dimensional vector given by $\boldsymbol{\beta}_i = [\beta_{(i,0)}, \dots, \beta_{(i,d)}]^{\mathrm{T}}$ and $\mathbf{x}^{+} = [1, \mathbf{x}]^{\mathrm{T}}$.
- Layer 5. This layer is constituted by a single node that computes the overall output $\hat{y}$ as the sum of all the normalized firing strengths:

$$\hat{y} = \sum_{i=1}^{m} \tilde{f}_i\,. \tag{6.6}$$

In first instance, the case when the entire training set is available before the learning procedure is considered.

Definition 6.3. Let $\mathcal{T} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ be the set of data available for the training phase, and denoting with $x_{(r,j)}$ the jth component of the rth input vector, then the hidden matrix $\mathbf{H} = [\mathbf{H}^1, \dots, \mathbf{H}^m]$ can be defined as:

$$\mathbf{H}^i = \begin{bmatrix} \overline{w}_i & \overline{w}_i x_{(1,1)} & \cdots & \overline{w}_i x_{(1,d)} \\ \overline{w}_i & \overline{w}_i x_{(2,1)} & \cdots & \overline{w}_i x_{(2,d)} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{w}_i & \overline{w}_i x_{(n,1)} & \cdots & \overline{w}_i x_{(n,d)} \end{bmatrix},\ i = 1 \dots m, \tag{6.7}$$

. Rearranging the parameters $\boldsymbol{\beta}$ in the form:

$$\boldsymbol{\beta} = [\boldsymbol{\beta}_{(1,0)} \cdots \boldsymbol{\beta}_{(1,d)} \boldsymbol{\beta}_{(2,0)} \cdots \boldsymbol{\beta}_{(2,d)} \cdots \boldsymbol{\beta}_{(m,0)} \cdots \boldsymbol{\beta}_{(m,d)}]^{\mathrm{T}}, \tag{6.8}$$

the optimization problem for training an RWFNN can be reformulated as a Least-Squares (LS) problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{y} - \mathbf{H}\boldsymbol{\beta}\|_2^2 + \frac{\lambda}{2} \|\boldsymbol{\beta}\|_2^2 \tag{6.9}$$

where $p = m(d+1)$ and $\lambda > 0$ is the *regularization factor*.

The problem (6.9) has a unique solution, given by:

$$\boldsymbol{\beta}^* = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\mathbf{I}\right)^{-1} \mathbf{H}^{\mathrm{T}}\mathbf{y}\,. \tag{6.10}$$

The learning procedure can be easily adapted to the classification problem. In the case of binary classification, where $y \in \{-1, 1\}$, the estimated class of a pattern $\mathbf{x}$ is given by:

$$\hat{y} = \operatorname{sgn} f(\mathbf{x}) . \tag{6.11}$$

In a multi-class classification problem, the output y can assume values in set of M labels, each one corresponding to a class. A possible way to represent this situation is to represent a generic output $\mathbf{y}_i$ as a vector of M elements $y_{i,j}$, $y_{i,j} = 1$ if the correspondent $\mathbf{x}$ is of class j, and $y_{i,j} = 0$ otherwise. In this case $\boldsymbol{\beta}$ becomes a $p \times M$ matrix and $\mathbf{y}$ a $N \times M$ matrix, and the norm in (6.9) is replaced with a matrix norm (e.g. the Euclidean norm).

6.1.3 Online learning of RWFNNs

In the classical training of ANFIS networks, a typical problem arises in the estimation of numerical parameters of antecedents and consequents for each fuzzy rule. The methods used to update the parameters can be broadly divided into batch learning schemes and sequential learning schemes.

Definition 6.4. Batch learning schemes assume that the entire training data is available before starting the learning process.

They can be used exclusively for the training offline, keeping the system parameters constant while computing the error associated with each sample in the input. In particular, the learning algorithm updates its parameters after consuming the whole batch. In [86, 91, 90] a batch version of the ANFIS technique is proposed, where the algorithm requires a cycle on the complete dataset over a number of epochs.

On the other hand, there is the sequential approach:

Definition 6.5. Sequential learning schemes assume that the entire training data is not available before beginning the training procedure, but data arrives one-by-one or chunk-by-chunk.

The sequential learning should be used for both the off line and on-line training. The former considers a static dataset, while in the latter some additional constraints have to be inserted. In both cases, data used for learning become available in a sequential order before being used to update the best predictor at each step. These methods are particularly suited for those contexts where it is computationally infeasible to train over the entire dataset for memory and computationally requirements, or for those ones where the dynamical adaption of the algorithm to new patterns in the data is mandatory. For their capability to work with non stationary dataset, the on-line learning approaches started to be extensively studied. In [92] an approach based on an on-line learning of Takagi-Sugeno (TS) type is proposed. Furthermore, a Simplified eTS (Simpl_eTS) [93], a dynamic evolving neurofuzzy inference system (DENFIS) [94], and a sequential adaptive fuzzy inference system (SAFIS) [95] represent only few part of the literature involved in this field. However, all these approaches are able to handle data on a one-by-one basis, causing problems when data arrive on a chunk-by-chunk basis. For that, in the following section is presented an algorithm able to work in contexts where data arrives chunk by chunk. Additionally, to make the problem more realistic, the application is allowed to be extended on a distributed context where data are acquired in different local areas.

Thus, for the rest of these sections, an on-line learning will be considered:

Assumption 6.6. Let us assume that the training set is not entirely available before the learning procedure but it is provided to the model in form of a sequence of mini-batches $\mathcal{T}_1, \ldots, \mathcal{T}_N$, such that:

$$\mathcal{T} = \bigcup_{k=1}^{\mathrm{N}} \mathcal{T}_k \,. \tag{6.12}$$

This is a typical scenario in systems where data is acquired and processed continuously over time (e.g. sensors in industrial plants, trade data in financial markets), or when the size of the training set is extremely large to make the matrix inversion in (6.9) computationally infeasible. In this case, the RWFNN can be trained using the BRLS algorithm, which consists of a sequence of two-step updates. Let $\boldsymbol{\beta}[0] = \mathbf{0}$ and $\mathbf{P}[0] = \lambda^{-1}\mathbf{I}$ be the initial estimates of the consequents parameters and of the state matrix respectively. At the $(k+1)$th iteration, a new chunk of data $\mathcal{T}_{k+1}$ is presented to the model. Denoting with $\mathbf{H}_{k+1}$ and $\mathbf{y}_{k+1}$ the hidden matrix and the output vector computed on the new data, the estimates of the state matrix and of the consequents parameters are updated according to:

$$\mathbf{P}[k+1] = \mathbf{P}[k] - \mathbf{P}[k]\,\mathbf{H}_{k+1}^{\mathrm{T}}\mathbf{K}_{k+1}^{-1}\mathbf{H}_{k+1}\mathbf{P}[k] \tag{6.13}$$

$$\boldsymbol{\beta}[k+1] = \boldsymbol{\beta}[k] + \mathbf{P}[k+1]\,\mathbf{H}_{k+1}^{\mathrm{T}}\left(\mathbf{y}_{k+1} - \mathbf{H}_{k+1}\boldsymbol{\beta}[k]\right) \,, \tag{6.14}$$

where

$$\mathbf{K}_{k+1} = \mathbf{I} + \mathbf{H}_{k+1}\mathbf{P}[k]\,\mathbf{H}_{k+1}^{\mathrm{T}} \,. \tag{6.15}$$

The recursive formulation is based essentially on the use of the QR decomposition and, at each iteration, the new matrix is appended to the matrices originated by the QR decomposition of the preceding iteration. The 'old' and the 'new' information are weighted by a forgetting factor λ. This procedure is characterized by very good numerical stability, the new parameters are evaluated by using only raw data and without forming any correlation matrix. Working directly on data, and using a stable linear technique like the QR, the robustness of the algorithm is increased and the numerical stability is enhanced. Details on the convergence of the algorithm can be found in [96] and references therein.

6.1.4 Distributed Average Consensus

Sec.3.5 has already introduced the distributed average consensus protocol, while this chapter applies this theory to the distributed learning of a RWFNN model.

Theoretically, different strategies can be employed in the distributed learning of ANFIS networks. That is, by the application of rules' synthesis based on distributed fuzzy clustering and distributed least-squares estimation. A preliminary work can be found in [97], where an OS-Fuzzy-ELM is presented. All of the antecedent parameters of membership functions are randomly assigned first, and then, the corresponding consequent parameters are determined analytically. However, this topic has not been exploited sufficiently in literature.

Exploiting the fact that when dealing with the RWFNNs the parameters of the membership functions are randomly selected instead of being estimated during the learning process, in the following section this approach is combined with the DAC protocol, in a distributed scenario.

Differently from what is generally performed by the centralized online learning algorithm, in the proposed distributed approach each agent is able to receive and work with a stream of one-by-one data or chunk by chunk. The concept of epoch disappears, allowing the algorithm to avoid retraining whenever new data is received. In particular, when a new chunk of data is received each agent updates the estimate of the consequent parameters by using the Block Recursive Least-Squares (BRLS) algorithm. Then the approach is able to converge to a single model for all agents, by using the DAC protocol. The learning algorithm is designed such that it does not require the presence a coordination authority, nor the exchange of any possibly sensitive data through the network.

For the rest of the work a network of L interconnected agents, whose connectivity is described by a directed graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ is considered, where $\mathcal{V}=\{1,\ldots,L\}$ is the set of the agents and each edge $e\in\mathcal{E}$ represents a connection. The existence of a connection implies a communication between the two agents. An agent i is said 'neighbor' of an agent j if $e_{i,j}\in\mathcal{E}$.

Let us resume briefly the basic concept of DAC. Each agent i has a vector of measurements $\boldsymbol{\theta}_i$, which represents its estimate of the quantity $\boldsymbol{\theta}$. The goal of the protocol is for all the agents to converge to the global average:

$$\hat{\boldsymbol{\theta}} = \frac{1}{L}\sum_{l=1}^{L}\boldsymbol{\theta}_l\,. \tag{6.16}$$

For discrete-time distributed systems the $(n+1)$th iteration of the protocol is defined by a set of linear updating equations:

$$\boldsymbol{\theta}_l\left[n+1\right] = \sum_{l=1}^{L}\boldsymbol{\theta}_l\left[n\right]\,, \tag{6.17}$$

which can be represented in form of a linear system:

$$\boldsymbol{\theta}\left[n+1\right] = \mathbf{W}\boldsymbol{\theta}\left[n\right]\,. \tag{6.18}$$

If the weights $\mathbf{W}$ are chosen appropriately, the iterations defined in (6.18) converge locally to the average (6.16).

6.2 Distributed On-Line Learning of RWFNN

This section puts together all of the previous notations by formally introducing the distributed on-line procedure for the RWFNN. A training set $\mathcal{T}$ distributed over a network of agents is considered.

Assumption 6.7. Each agent k receives a stream of data in form of a sequence of mini-batches $\mathcal{T}_{k,1}, \ldots, \mathcal{T}_{k,N}$, such that:

$$\begin{aligned}\bigcup_{i=1}^{N} \mathcal{T}_{k,i} &= \mathcal{T}_k \\ \bigcup_{k=1}^{L} \mathcal{T}_k &= \mathcal{T}\end{aligned} \tag{6.19}$$

The goal is to agree on a single model for each agent, with performance comparable to a centralized RWFNN trained using the entire data set available up until then. In order to make the algorithm suitable to be implemented in a variety of architectures and applications with stringent requirements in terms of connectivity and privacy, a set of constraints have to been introduced.

Assumption 6.8. The algorithm must satisfy the following items:

- no agent is allowed to coordinate the training process;
- agents are not allowed to exchange any data pattern;
- only local communication between connected agents is possible.

By combining the problem of distributed learning with the RWFNN criterion, it can be said that:

Definition 6.9. The global optimization problem of the distribute RWFNN can be stated as:

$$\beta^* = \min_{\beta \in \mathbb{R}^p} \frac{1}{2}(\sum_{k=1}^{L} \|\mathbf{y}_k - \mathbf{H_k}\beta\|_2^2) + \frac{\lambda}{2}\|\beta\|_2^2 \tag{6.20}$$

where $\mathbf{H}_k$ and $\mathbf{y}_k$ are the hidden matrix and output vector computed over the local dataset S_k.

Remember from Section 6.1.2 the optimal weight vector, in this case, it can be expressed as:

$$\begin{aligned}\boldsymbol{\beta}^* &= (\sum_{k=1}^{L}(\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k) + \lambda\mathbf{I})^{-1}\sum_{k=1}^{L}(\mathbf{H}_k^{\mathrm{T}}\mathbf{y}_k) \\ &= \left(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\mathbf{I}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{y}\,.\end{aligned} \tag{6.21}$$

The proposed algorithm finds the optimal solution of (6.21) in a distributed fashion, using an alternation of local updating steps based on the BRLS algorithm introduced in Sect. 6.1.3 and global averages steps computed using the DAC protocol illustrated in sec. 6.1.4. Practically, the algorithm consists in the following steps:

1. **Initialization**: all the agents agree on the parameters of the antecedents. This step can be achieved with a single run of the DAC protocol. Additionally, each agent i initializes its state matrix $P_i[0] = \lambda^{-1}\mathbf{I}$ and its consequents parameters $\boldsymbol{\beta}_i[0] = \mathbf{0}$.

2. At the $(k+1)$th iteration, each agent i receives a new chunk $\mathcal{T}_{i,k+1}$. Then the iteration consists of two phases:

   (a) **Local update**: the agents update their local estimates of the parameters of the consequents $\boldsymbol{\beta}_i[k+1]$ and of the state matrix $\mathbf{P}_i[k+1]$ according to Eqs. (6.13)-(6.14), using the local data $\mathcal{T}_{i,k+1}$.

   (b) **Global average**: all the agents agree on a single vector for the consequents parameters through the use of the DAC protocol. The final value for the parameters of the consequents at the iteration $k+1$ is given by:

$$\boldsymbol{\beta}[k+1] = \frac{1}{L}\sum_{i=1}^{L}\boldsymbol{\beta}_i[k+1]\,. \tag{6.22}$$

The overall algorithm at the ith agent is summarized in Alg. 4.

6.3 Experimental validity

6.3.1 Description of the datasets

To validate the proposal, the algorithm has been tested on three datasets available from the UCI Machine Learning repository[1], whose characteristics are summarized in Table 6.1. The datasets have been chosen in such a way that they could represent a variety of applicative domains and problems. Below, more information about each of them is provided:

- *Concrete* - This is a civil engineering dataset [98] representing a regression problem, where the attributes represent some chemical properties of the concrete, and the goal is to predict the concrete compressive strength.

- *Transfusion* - This is a binary classification dataset, consisting in an extraction of records from a donor database [99]. The features represent the statistics about the donor's past blood donations. The goal is to predict whether the donor donated blood in March 2007.

- *CCPP* - This dataset represents the problem of predicting the electrical energy output of a power plant from some ambient variables [100, 101]. Particularly interesting in the context of the proposal is that the measurements have been collected from a network of sensors located around the plant during a period of 6 years.

All the datasets have been preprocessed before the training procedure by scaling their input features between -1 and 1. Each input feature is mapped onto two linguistic labels.

[1]http://archive.ics.uci.edu/ml

Algorithm 4 Pseudocode of The On-line Distributed RWFNN Algorithm at The ith Agent

```
1: **Input** : Let $L$ (global) as the number of agents, $N$ as the number of mini-batches, $\mathcal{T}_i$ as the training set, $\lambda$ as the regularization parameter and $\epsilon$ as the tolerance threshold.
2: **Output** to be estimated is $\boldsymbol{\beta}^*$.
3: Generate the parameters of the membership functions, in accordance with the other $L-1$ agents.
4: Initialize $\boldsymbol{\beta}_i^*[0] = 0$, $\mathbf{P}_i[0] = \lambda^{-1}\mathbf{I}$.
5: **for** $k = 0$ to $N-1$ **do**
6:     Update $\boldsymbol{\beta}_i[k+1]$ and $\mathbf{P}_i[k+1]$ according to:
           $\mathbf{P}[k+1] = \mathbf{P}[k] - \mathbf{P}[k]\mathbf{H}_{k+1}^{\mathrm{T}}\mathbf{K}_{k+1}^{-1}\mathbf{H}_{k+1}\mathbf{P}[k]$
           $\boldsymbol{\beta}[k+1] = \boldsymbol{\beta}[k] + \mathbf{P}[k+1]\mathbf{H}_{k+1}^{\mathrm{T}}(\mathbf{y}_{k+1} - \mathbf{H}_{k+1}\boldsymbol{\beta}[k])$ ,
       where
           $\mathbf{K}_{k+1} = \mathbf{I} + \mathbf{H}_{k+1}\mathbf{P}[k]\mathbf{H}_{k+1}^{\mathrm{T}}$ .
7:     Compute $\boldsymbol{\beta}_i^*[k+1]$ using the DAC protocol:
           $\boldsymbol{\beta}[k+1] = \frac{1}{L}\sum_{i=1}^{L}\boldsymbol{\beta}_i[k+1]$
8: **end for**
9: **return** $\boldsymbol{\beta}_i^*[k+1]$
```

Assumption 6.10. For each linguistic label a triangular Membership Function is used:

$$\mu(x;a,b,c) = \begin{cases} 0 & x < a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ \dfrac{c-x}{c-b} & b \le x \le c \\ 0 & x > c \end{cases} , \tag{6.23}$$

where a, b and c are sampled from the uniform distribution in $[-1, 1]$, with the constraint that $a < b < c$.

Since in this work the aim is not the one of achieving the lowest possible error for the datasets but rather to demonstrate the effectiveness of the proposal, a thorough search of the best fuzzy set for each variable has not be conducted.

Dataset	Features	Instances	Task	Desired Output
Concrete	8	1030	R	Concrete Strength
Transfusion	4	748	BC	Blood Donation
CCPP	4	9568	R	Electrical Energy

Table 6.1. Description of The Datasets.

6.3.2 Experimental setup

An open-source MATLAB toolbox [2] has be used to implement the proposal. During the implementation, the attention is not concerned on the effect of the communication deriving by a distributed implementation, but more about the convergence behavior of the algorithm. For this reason, it is provided a serial version of the algorithm where the network is simulated. The following algorithms are compared:

- **Cons-RWFNN**: this is a RWFNN trained according to the DAC-based procedure illustrated in Sect. 6.2. For all the experiments the maximum number of consensus iterations is set to 300 and $\epsilon = 10^{-4}$.
- **OLS-RWFNN**: this is a RWFNN trained using centralized BRLS-based algorithm illustrated in Sect. 6.1.3, where all the training set is collected and made available at a single node. This model serves as a lower bound for the accuracy of the distributed approach.
- **L-RWFNN**: in this model, each agent receives a stream of data and updates its local estimate of the consequents parameters, but no agreement takes place in the network.
- **ADMM-RWFNN**: this is the model proposed in [90], where the training data is distributed through the network and the RWFNN is trained in batch on the entire dataset using the well-known Alternating Direction Method of Multipliers (ADMM).

The results of the experiments are computed over a 10-fold cross validation on each dataset, and each experiment is repeated 10 times to average out and mitigate possible randomness effects due to the random initialization of the parameters of the antecedents. The optimal values for the regularization factor λ are obtained by running a 5-fold cross validation using OLS-RWFNN on the training data, and then sharing them with all the other algorithms. The regularization parameter λ is searched in the discrete interval 10^j, $j \in \{-3, -2, \ldots, 2\}$, and the optimal values resulting from the search are reported in Table 6.2.

For all the experiments, a 12-agent network is constructed according to the so-called 'Erdős−Rényi model'[84], where every pair of agents has a 35% of probability to be connected, and the network is forced to be strongly connected. The edge weights $\mathbf{W}$ are chosen using the 'max degree' strategy (3.27) described in Section 3.5

[2]https://bitbucket.org/robertofierimonte/code-distributed-online-rwfnn

Dataset name	λ
Concrete	10^{-3}
Transfusion	10^{-2}
CCPP	10^{-2}

Table 6.2. Optimal Values for The Regularization Factor λ

6.3.3 Results and discussion

In the experiments, an analysis on how the results change when new patterns are presented to the systems is performed in terms of time of convergence. For all the datasets, the models are initialized using a small block of 100 patterns, and each subsequent mini-batch consists in 20 patterns. In case of regression datasets - Concrete and CCPP - the performance of the models are evaluated using the Normalized Root Mean-Squared Error (NRMSE):

$$\text{NRMSE} = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N\hat{\sigma}_y}} \tag{6.24}$$

where $\hat{\sigma}_y$ is the estimated variance of $\{y_i, \ldots, y_N\}$.

For the Transfusion dataset, the average misclassification error is used. The final measurement results are summarized in Table 6.3. As it can be expected, the

Dataset	Algorithm	Test Error	Training Time [s]
Concrete	OLS-RWFNN	0.256 ± 0.147	18.410 ± 1.860
	Cons-RWFNN	$\mathbf{0.262 \pm 0.013}$	$\mathbf{17.870 \pm 1.638}$
	L-RWFNN	0.325 ± 0.022	17.780 ± 1.635
	ADMM-RWFNN	0.258 ± 0.157	2.180 ± 0.174
Transfusion	OLS-RWFNN	0.223 ± 0.014	0.294 ± 0.028
	Cons-RWFNN	$\mathbf{0.229 \pm 0.010}$	$\mathbf{0.291 \pm 0.009}$
	L-RWFNN	0.237 ± 0.007	0.269 ± 0.009
	ADMM-RWFNN	0.224 ± 0.014	0.211 ± 0.135
CCPP	OLS-RWFNN	0.072 ± 0.012	1.616 ± 0.325
	Cons-RWFNN	$\mathbf{0.087 \pm 0.006}$	$\mathbf{1.630 \pm 0.399}$
	L-RWFNN	0.154 ± 0.016	1.485 ± 0.316
	ADMM-RWFNN	0.072 ± 0.012	0.370 ± 0.059

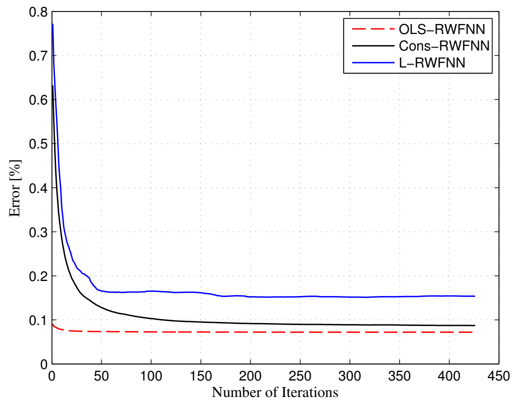
Table 6.3. Final Values of Test Error and Computational Time (Average and Standard Deviation)

centralized approach is able to obtain the best results in terms of test error, for all the considered datasets. This happens because the global vision of the entire dataset reduces the possibility of error during the training phase, although increasing the amount of data to be treated in a single central processor. Coherently with the expectations, L-RWFNN obtains the worst results in terms of test error since no collaboration among the agents is allowed during the training procedure. Additionally, it can be noted that the proposed algorithm is able to track efficiently the accuracy of OLS-RWFNN and ADMM-RWFNN, since it achieves only a slightly higher error, ranging from an additional 0.01 for Concrete and CCPP to approximately the same values for Transfusion. Additionally, the training procedure for Cons-RWFNN is generally faster when compared to the centralized approach, since the dataset is partitioned among the agents, making the process faster and easier.

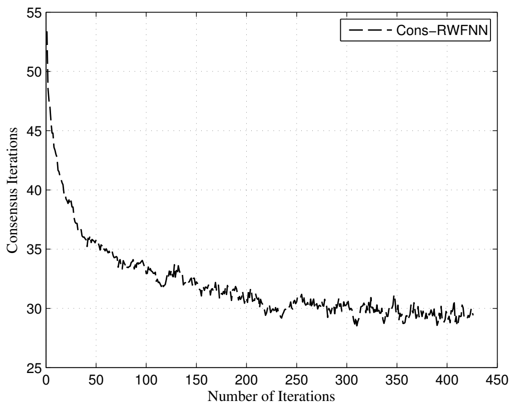
For a visual representation the evolution of the error as new data arrives at the network, is reported on the left panel of Fig. 6.3. It should be observed that in all cases, L-RWFNN fails in tracking the performance of OLS-RWFNN, especially for Concrete. Additionally, it suffers from high variance, due to the different results achieved by the agents. On the opposite, Cons-RWFNN is able to track the behavior of OLS-RWFNN, achieving very similar performance, in particular for CCPP and Concrete. For Transfusion dataset, it can be noted an imperceptible increase in the error of Cons-RWFNN after 20 iterations caused by the procedure used for distribute the data. Finally, on the panel on the right of Fig. 6.3 is shown the evolution of the number of DAC iterations. As it can be expected, the number of iterations required to reach consensus decreases as the numbers of available patterns increases, showing that at the end all the agents converge to a common structure.

6.3.4 Observations

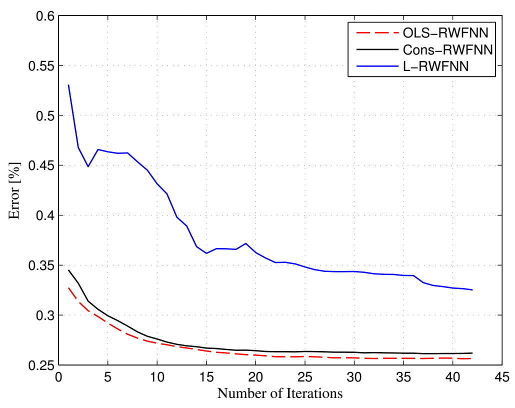
In this chapter an innovative on-line training method for RWFNN is introduced. The parameters of the antecedent are randomly fixed before the training process and the case is the one in which the training data are distributed across a network of interconnected agents. Experimental results obtained on benchmark datasets prove the effectiveness of our proposal and its ability to efficiently match a fully centralized implementation, without however requiring the presence of a coordinating agent. A natural extension of this problem might concern the use of different and more complex FNN architecture.



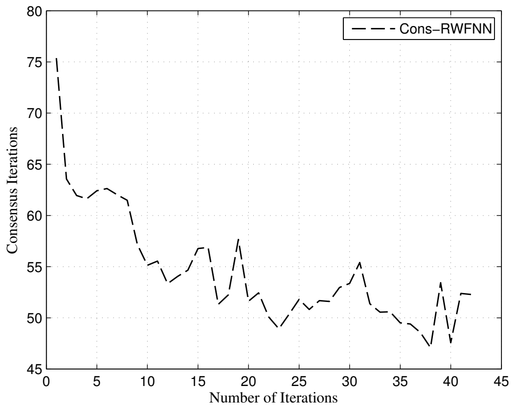
(a) Training Error (Dataset CCPP)



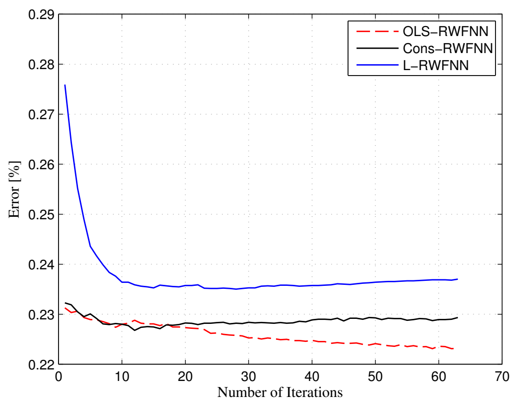
(b) Consensus Iterations (Dataset CCPP)



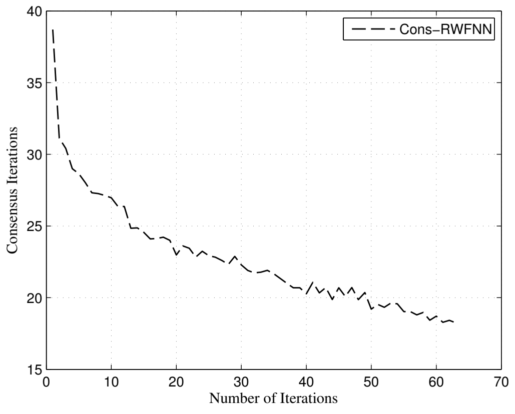
(c) Training error (Dataset concrete)



(d) Consensus Iterations (Dataset concrete)



(e) Training error (Dataset Transfusion)



(f) Consensus Iterations (Dataset Transfusion)

Figure 6.3. Performance of Cons-RWFNN and L-RWFNN, compared to OLS-RWFNN, as the agents obtain new data. The panels on the left show the evolution of the error, while the panels on the right show the iterations to reach consensus.

Chapter 7

Distributed Expectation-Maximization

This section follows the lines of the previous chapters, where the problem of performing inference on data distributed across a network of agents, with limited connectivity, is treated. A non-convex distributed optimization in multi-agent networks with time-varying (not symmetric) connectivity, is applied to the well-known Expectation Maximization (EM) approach. The local solutions of the EM problem are found by exploiting a convexification-decomposition technique, through which a sequence of strongly convex, decoupled, optimization subproblems is created. The agreement is achieved by a consensus-based update.

7.1 Introduction

As widely discussed in the previous chapters, distributed optimization techniques have witnessed a surge of interest for multi-agent systems fields. As previously said, many such problems can be formulated as the cooperative minimization of the agents' sum-utility J:

$$\begin{aligned} &\min_{\mathbf{w}} \sum_{k=1}^{L} J_k(\mathbf{w}) \\ &\text{s.t. } \ \mathbf{w} \in \mathcal{W} \end{aligned} \tag{7.1}$$

where each $J_k(\mathbf{w}) : \mathbb{R}^m \to \mathbb{R}$ is the smooth cost function of agent $i \in \{1, \dots, L\}$. In some cases, to promote some extra structure in the solution, a nonsmooth term is added to the overall objective function:

$$\begin{aligned} &\min_{\mathbf{w}} \sum_{k=1}^{L} J_k(\mathbf{w}) + G(\mathbf{x}) \\ &\text{s.t. } \ \mathbf{w} \in \mathcal{W} \end{aligned} \tag{7.2}$$

In this way the solution could be regularized; for instance, the structures $G(\mathbf{x}) = c\,\|\mathbf{x}\|_1$ or $G(\mathbf{x}) = c\sum_{i=1}^{N} \|\mathbf{x}_i\|_2$ are widely used to impose sparsity on it.

Optimization problems in the form of (7.2) are interesting in many scientific endeavors, especially for those ones where the optimization could not be performed in a centralized fashion. Motivated by these observations, a solution method, with provable convergence, has been developed in the next section for the general class of density-based clustering algorithms. Among them, the Gaussian Mixture Model is certainly the most famous one [102], with batch and incremental versions rely on data that is assumed to be available at a central processing unit [103, 104].

Recently, parallel and partially decentralized version of them starts to be proposed, with the still presence of a master-slave architecture to achieve performances comparable to the centralized scheme. However, they are not applicable for the general formulation of (7.2), and/or in a fully distributed scenario as the one introduced in chapter 3. For instance, some of them require the knowledge of the whole J from all the agents; others call for the presence of a fusion center collecting at each iteration data from all the agents; others are implementable only on specific network topologies.

This chapter introduces a new proposal to this issue, where a distributed framework with provable convergence for the nonconvex multi-agent optimization is presented. It is based on the Gaussian Mixture Model for solving an Expectation Maximization approach. The agreement among the local results is achieved by a step of consensus learning. A convexification-decomposition technique [105] is used to find the local solution of the EM problem of each agent, by means of a sequence of strongly convex, decoupled, optimization subproblems. In each subproblem, the agents compute theirs estimate locally, independently from the others. Successively, a consensus step is performed to force an agreement on the different solutions. A stopping criterion is inserted to break the procedure when an ad-hoc convergence criterion will be satisfied.

In the following sections, the centralized Expectation-Maximization approach is first introduced. Successively, is detailed the proposal to its distributed version.

7.2 Problem Statement

For the rest of the chapter, a distributed setting satisfying the below assumptions will always considered.

Assumption 7.1. (On the problem)
The network is composed of I agents such that:

A1 each agent collects M_i measurements $y_{i1}, \dots y_{iM_i}$, with $y_{im} \in \mathbb{R}^D$ for all $i, \dots m$;

A2 the measurements are assumed to be independent and identically distributed (i.i.d);

A3 the environment is stationary and unchanging during the course of the measurement process thanks to the i.i.d. assumption.

Assumption 7.2. (On Data Distribution)
Let $\mathcal{N}(y|\mu, \Sigma)$ denotes the evaluation of a Gaussian density with mean μ and covariance Σ at point y, then:

- the precision matrix is defined as $P = \Sigma^{-1}$;
- the measurements are assumed to obey a Gaussian mixture distribution with K classes of the form:

$$y_{im} \sim \sum_{k=1}^{K} \pi_{ik} \mathcal{N}(y_{im}|\mu_k, \Sigma_k), \tag{7.3}$$

  $m = 1, \ldots, M_i$, $i = 1, \ldots, I$;
- π_{ik} is the prior probability of the agent i associated with the k^{th} Gaussian component, and such that $\sum_{k=1}^{K} \pi_{ik} = 1$.

All parameters are unknown, thus the goal consists in obtaining a distributed algorithm for estimate such parameters from the underlying data $y = \{y_{im}\}_{i,m}$. Following a maximum likelihood (ML) approach, the proposal proceeds by maximizing the log-likelihood function:

$$\mathcal{L}(y|\pi, \mu, \Sigma) = \sum_{i=1}^{I} \sum_{m=1}^{M_i} \log \left(\sum_{k=1}^{K} \pi_{ik} \mathcal{N}(y_{im}|\mu_k, \Sigma_k) \right) \tag{7.4}$$

where the set of unknown parameters is given by the tuple $\theta = \{\pi, \mu, \Sigma\}$, where $\pi = \{\pi_i\}_{i=1}^{I}$ with $\pi_i = \{\pi_{ik}\}_{k=1}^{K}$, $\mu = \{\mu_i\}_{i=1}^{I}$, and $\Sigma = \{\Sigma_i\}_{i=1}^{I}$.

The EM algorithm is a well known numerical method used to compute the ML estimates in the presence of incomplete observations [106].

In particular, a set of latent variables $z \in \mathbb{R}^{I\overline{M}}$, with $\overline{M} = \sum_{i=1}^{I} M_i$ is generally introduced. They are not directly observed but are rather inferred from the observations. They represent an indicator vector such that, the lth entry z_l is equal to k if the lth observation belongs to the kth class.

Hence, the incomplete observations are combined with the complete ones to give the overall set of variables to the model. Since z is unknown, it will be treat as a random variable. In the sequel, for completeness of exposition, a recall on the centralized EM algorithm is given.

7.3 Centralized EM Algorithm

In a centralized approach, the entire vector of observations y is sent to a single computing unit, which is in charge of performing the maximization of (7.4), which could be rewrite avoiding the dependence on the number of agents:

$$\mathcal{L}(y|\pi, \mu, \Sigma) = \sum_{m=1}^{M} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(y_m|\mu_k, \Sigma_k) \right) \tag{7.5}$$

Assumption 7.3. Data Distribution

A1 Let $\beta^t = \{[\mu_k^t]_{k=1}^K, [\text{vec}(\Sigma_k^t)]_{k=1}^K\}$ be the set of estimates of means and covariances at the tth iteration of the EM algorithm, where $\text{vec}(\cdot)$ denotes the vectorization operator.

A2 for each node, let $\pi^t = \{\pi_i^t\}_{i=1}^I$ be the estimates of the mixing probabilities at the t-th iteration over all the network.

A3 let $\theta^t = \{\pi^t, \beta^t\}$.

Then, the EM iteration alternates between performing an expectation (E) step, and a maximization one. In the former, it is created a function for the expectation of the log-likelihood by using the current estimate of the parameters. While in the latter, the parameters are evaluated by maximizing the expected log-likelihood obtained in the previous step.

Expectation Step: at each iteration t, the algorithm replaces (7.4) with a concave lower bound evaluated around the current estimate θ^t, which is given by the conditional expectation:

$$Q(\theta, \theta^t) = \mathbb{E}_z \left\{ \mathcal{L}(y, z|\theta)|y, \theta^t \right\} = \sum_{i=1}^{I} \widetilde{g}_i(\theta; \theta^t) \tag{7.6}$$

where

$$\widetilde{g}_i(\theta; \theta^t) = \sum_{m=1}^{M_i} \sum_{k=1}^{K} \alpha_{ikm}^t \left[\ln \pi_{ik} + \ln \mathcal{N}(y_{im}|\beta_k) \right] \tag{7.7}$$

$$\alpha_{ikm}^t = \frac{\pi_{ik}^t \mathcal{N}(y_{im}|\beta_k^t)}{\sum_{k=1}^K \pi_{ik}^t \mathcal{N}(y_{im}|\beta_k^t)}, \quad m = 1 \ldots M_i, \quad k = 1 \ldots K, \; i = 1 \ldots I. \tag{7.8}$$

It is easy to prove that (7.6) has the same gradient of (7.4) when is evaluated at θ^t. Thus, (7.6) and (7.4) share the same optimality conditions at θ^t. Everything in the E step is known before the step is taken except for the α_{ikm}, which is computed according to (7.8) at the beginning of the E step section.

Maximization Step: the new estimate θ^{t+1} is obtained by solving the optimization problem:

$$\theta^{t+1} = \arg\max_{\theta \in \mathcal{C}} Q(\theta; \theta^t), \tag{7.9}$$

where $\mathcal{C}$ is a feasibility set that constrains the values of θ. Typically, $\mathcal{C}$ requires that π_i is a positive vector whose components sum up to one, for all i, and that each matrix $\Sigma_k \in \mathbb{S}_{++}^Q$ (or $P_k \in \mathbb{S}_{++}^Q$), for all k, where $\mathbb{S}_{++}^Q$ is the cone of positive definite matrices.

Problem (7.9) admits the closed form solution:

$$\pi_{ik}^{t+1} = \frac{1}{M_i} \sum_{m=1}^{M_i} \alpha_{ikm}^t, \quad \text{for all } i, k, \tag{7.10}$$

$$\mu_k^{t+1} = \frac{\sum_{i=1}^{I}\sum_{m=1}^{M_i} y_{im}\alpha_{ikm}^t}{\sum_{i=1}^{I}\sum_{m=1}^{M_i}\alpha_{ikm}^t}, \quad \text{for all } k, \tag{7.11}$$

$$\Sigma_k^{t+1} = \frac{\sum_{i=1}^{I}\sum_{m=1}^{M_i}(y_{im}-\mu_m^t)(y_{im}-\mu_m^t)^T\alpha_{ikm}^t}{\sum_{i=1}^{I}\sum_{m=1}^{M_i}\alpha_{ikm}^t}, \quad \text{for all } k. \tag{7.12}$$

The EM algorithm is guaranteed to convergence to a local maximum of the likelihood function [107], but it is sensitive to the initialization of the parameters. Therefore, to ensure convergence to a good local minimum, a suitable initialization is needed. It is important to notice that while the estimate of prior probabilities in (7.10) requires knowledge of local information only, the estimates of the other parameters in (7.11), (7.12), require global information.

In the next section, it is illustrated a way of distributing the optimization in (7.9) via local cooperation among agents embedded into a sparse communication network.

7.4 Distributed EM Algorithm over networks

Let us now consider a distributed environment, where agents are connected to each other via a sparse undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the set of vertices (nodes) and $\mathcal{E}$ is the set of edges. The neighborhood of agent i (including node i) is defined as $\mathcal{N}_i = \{j|(j,i) \in \mathcal{E}\} \cup \{i\}$; it sets the communication pattern between single-hop neighbors: agents $j \neq i$ in $\mathcal{N}_i$ can communicate with node i. Associated with graph $\mathcal{G}$, it is associated a set of positive weights $W = \{w_{ij}\} \in \mathbb{R}^{I\times I}$ matching the topology of $\mathcal{G}$, i.e., such that $w_{ij} > 0$ if $j \in \mathcal{N}_i$, and 0 otherwise. The following assumption on the network connectivity, and on the matrix W has also be performed:

Assumption 7.4. On the network topology/connectivity:

(A1) The graph $\mathcal{G}$ is connected, i.e., there exists a undirected path connecting any pair of nodes in the network;

(A2) The weight matrix W is doubly stochastic, i.e., it satisfies

$$W\,1 = 1 \quad \text{and} \quad 1^T W = 1^T. \tag{7.13}$$

Several choices able to satisfy Assumption A2, such as the Laplacian weights, the maximum degree weight, the Metropolis-Hasting and so on have already been introduced in Sec 3.5 .

Leveraging on the previous assumptions, the sequel will introduce the proposed distributed learning approach. Devising a distributed algorithm for maximizing (7.4) faces two main challenges, namely: the nonconvexity of the $\mathcal{L}$ in and the lack of global information at each agent side. To cope with these issues, a general framework for distributed nonconvex optimization over networks recently proposed in [105] has been exploit. It combines SCA techniques with dynamic consensus mechanisms, as described next.

Local SCA optimization: each agent i maintains a local estimate of the global optimization variables in (7.4) in the structure $\theta_i^t = \{\pi_i^t, x_i^t\}$. Specifically, π_i^t are the local prior probability vectors iteratively updated; while x_i is equal to $\{[\mu_{ik}^t]_{k=1}^K, [\mathrm{vec}(P_{ik}^t)]_{k=1}^K\}$ [with $P_{ik} = \Sigma_{ik}^{-1}$ being the local estimate of the k-th precision matrix].

Following the approach proposed in [105], at each iteration t, every agent builds a local (strongly concave) approximation ($\widetilde{f}$) of the objective function (7.4) at the current point θ_i^t. As for the centralized implementation of the EM algorithm, a natural choice for such approximation exploits the surrogate function in (7.6)-(7.7), in order to permit each agent to compute *locally* and *efficiently* the new iteration. In particular, each agent i solves the following optimization problem:

$$\max_{(\pi_i, x_i)\in\mathcal{C}} \quad \widetilde{f}_i(\pi_i, x_i; \pi_i^t, x_i^t) + \sum_{j\neq i} h_j^t(x_i) \tag{7.14}$$

where

$$\widetilde{f}_i(\pi_i, x_i; \pi_i^t, x_i^t) = \widetilde{g}_i(\pi_i, x_i; \pi_i^t, x_i^t) - \frac{\tau}{2}\|x_i - x_i^t\|^2, \quad \text{for all } i, \tag{7.15}$$

$$h_j^t(x_i) = \sum_{m=1}^{M_j}\sum_{k=1}^{K} \alpha_{jkm}^t \ln \mathcal{N}(y_{jm}|x_{ik}), \quad \text{for all } j \neq i, \tag{7.16}$$

with $\tau > 0$. The solution of problem (7.14) cannot be computed by node i in a fully distributed manner yet, because the second term in the objective function depends on information (e.g., data) that is not locally available. To cope with this issue, the second term in (7.14) is linearized around point x_i^t, thus obtaining:

$$\max_{(\pi_i, x_i)\in\mathcal{C}} \quad \widetilde{f}_i(\pi_i, x_i; \pi_i^t, x_i^t) + q_i(x_i^t)^T(x_i - x_i^t) \tag{7.17}$$

where $q_i(x_i^t) = \sum_{j\neq i} \nabla_{x_i} h_j^t(x_i^t)$. Now, all the missing information needed by agent i to perform the computation in (7.17) is represented by the (unknown) gradient vector $q_i(x_i^t)$. Thus, as proposed in [105], the $q_i(x_i^t)$ in (7.17) is replaced with a *local* estimate, say $\widetilde{q}_i^t$, asymptotically converging to $q_i(x_i^t)$, and solve instead:

$$\left(\hat{\pi}_i^t, \hat{x}_i^t\right) \;=\; \operatorname*{arg\,max}_{(\pi_i, x_i)\in\mathcal{C}} \quad \widetilde{f}_i\left(\pi_i, x_i; \pi_i^t, x_i^t\right) + \widetilde{q}_i^{t\,T}(x_i - x_i^t) \tag{7.18}$$

In the sequel, it has been shown how to update the local estimate $\widetilde{q}_i^t$ in (7.18) in a totally distributed manner.

Dynamic consensus over the network: Rewriting $q_i(x_i^t)$ in (7.17) as

$$q_i(x_i^t) = I \cdot \underbrace{\left(\frac{1}{I}\sum_{j=1}^{I} \nabla_{x_i} h_j^t(x_i^t)\right)}_{\triangleq \overline{\nabla h}(x_i^t)} - \nabla_{x_i} h_i^t(x_i^t) \tag{7.19}$$

the updating of $\widetilde{q}_i^t$ is obtained by mimicking (7.19):

$$\widetilde{q}_i^t = I \cdot s_i^t - \nabla_{x_i} h_i^t(x_i^t), \tag{7.20}$$

where s_i^t is a local auxiliary variable (controlled by user i) that aims to asymptotically track $\overline{\nabla h}(x_i^t)$. Leveraging *dynamic* average consensus methods [108], this can be done updating s_i^t according to the following recursion:

$$s_i^{t+1} = \sum_{j \in \mathcal{N}_i} w_{ij} s_j^t + \Big(\nabla_{x_i} h_i^t(x_i^{t+1}) - \nabla_{x_i} h_i^t(x_i^t) \Big) \tag{7.21}$$

with $s_i^0 \triangleq \nabla_{x_i} h_i^t(x_i^0)$, and where $\{w_{ij}\}_{ij}$ is any set of weights satisfying Assumption A2. Since the weights are constrained by the network topology, agent i updates its estimate s_i^t by using messages received only from agents in its neighborhood $\mathcal{N}_i$. Furthermore, letting

$$\nabla_{x_i} h_i^t(x_i^t) = \{[\nabla_{\mu_{ik}} h_i^t(x_i^t)]_{k=1}^K, [\text{vec}(\nabla_{P_{ik}} h_i^t(x_i^t))]_{k=1}^K\},$$

it could be obtained:

$$\nabla_{\mu_{ik}} h_i^t(x_i^t) = P_{ik}^t \sum_{m=1}^{M_i} \alpha_{ikm}^t (y_{im} - \mu_{ik}^t) \tag{7.22}$$

$$\nabla_{P_{ik}} h_i^t(x_i^t) = \frac{1}{2} \sum_{m=1}^{M_i} \alpha_{ikm}^t \Big((P_{ik}^t)^{-1} - (y_{im} - \mu_{ik}^t)(y_{im} - \mu_{ik}^t)^T \Big) \tag{7.23}$$

for $k = 1, \ldots, K$, $i = 1, \ldots, I$.

7.4.1 Distributed Maximization Step

Problem (7.18) can be recast as:

$$\begin{aligned} \max_{(\pi_i, x_i) \in \mathcal{C}} \; & \sum_{m=1}^{M_i} \sum_{k=1}^{K} \alpha_{ikm}^t \ln(\pi_{ik}) + \sum_{m=1}^{M_i} \sum_{k=1}^{K} \alpha_{ikm}^t \ln \mathcal{N}(y_{i,m} | x_{ik}) + \widetilde{q}_i^{t\,T} (x_i - x_i^t) \\ & - \frac{\tau}{2} \| x_i - x_i^t \|^2 \end{aligned} \tag{7.24}$$

From (7.24), the prior probability vector π_i is obtained by solving the following local sub-problem:

$$\begin{aligned} \max_{\pi_i} \quad & \sum_{m=1}^{M_i} \sum_{k=1}^{K} \alpha_{ikm}^t \ln(\pi_{ik}) \\ \text{subject to} \quad & \sum_{k=1}^{K} \pi_{i,k} = 1, \quad \pi_{i,k} \in [0,1], \end{aligned} \tag{7.25}$$

which admits the closed form solution:

$$\hat{\pi}_{i,k}^t = \frac{1}{M_i} \sum_{m=1}^{M_i} \alpha_{ikm}^t, \qquad k = 1, \ldots, K. \tag{7.26}$$

Also, let $\widetilde{q}_i^t = \{\widetilde{q}_{\mu_i}^t, \widetilde{q}_{P_i}^t\}$, with $\widetilde{q}_{\mu_i}^t = [\widetilde{q}_{\mu_{ik}}^t]_{k=1}^K$ and $\widetilde{q}_{P_i}^t = [\widetilde{q}_{P_{ik}}^t]_{k=1}^K$. Thus, from (7.24), the mean vectors μ_{ik}'s are obtained by solving the following subproblem:

$$\begin{aligned} \max_{\{\mu_{ik}\}_{k=1}^K} \; & -\frac{1}{2} \sum_{m=1}^{M_i} \sum_{k=1}^{K} \alpha_{ikm}^t (y_{im} - \mu_{ik})^T P_{ik}^t (y_{im} - \mu_{ik}) + \sum_{k=1}^{K} \widetilde{q}_{\mu_{ik}}^{t\;T} (\mu_{ik} - \mu_{ik}^t) \\ & - \frac{\tau}{2} \sum_{k=1}^{K} \| \mu_{ik} - \mu_{ik}^t \|^2. \end{aligned} \tag{7.27}$$

Problem (7.27) is additive in the variables $\{\mu_{ik}\}_{k=1}^{K}$, and thus it can be split into K subproblems, each one having closed form solution given by:

$$\hat{\mu}_{i,k}^{t} = \left(P_{ik}\sum_{m=1}^{M_i}\alpha_{ikm}^{t} + \tau I\right)^{-1}\left(P_{ik}\sum_{m=1}^{M_i}\alpha_{ikm}^{t} y_{im} + \widetilde{q}_{\mu_{ik}}^{t} + \tau\mu_{ik}^{t}\right), \tag{7.28}$$

$k = 1, \dots K$.

Finally, from (7.24), the precision matrices P_{ik}'s are obtained by solving the following subproblem:

$$\max_{\{P_{ik}\}_{k=1}^{K}\in\mathbb{S}_{++}^{Q}} \frac{1}{2}\sum_{m=1}^{M_i}\sum_{k=1}^{K}\alpha_{ikm}^{t}\left(\ln|P_{ik}| - \text{Tr}\left((y_{im}-\mu_{ik}^{t})(y_{im}-\mu_{ik}^{t})^{T}P_{ik}\right)\right)$$
$$+\sum_{k=1}^{K}\widetilde{q}_{P_{ik}}^{t\,T}\text{vec}(P_{ik}-P_{ik}^{t}) - \frac{\tau}{2}\sum_{k=1}^{K}\|P_{ik}-P_{ik}^{t}\|_F^2, \tag{7.29}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. Problem (7.29) does not admit a closed form solution. However, (7.29) is a (strongly) convex problem, which is additive in the variables $\{P_{ik}\}_{k=1}^{K}$, and whose global optimal solution can be found using very efficient numerical tools [109].

7.4.2 The EMC algorithm

This section is now in the position to formally introduce the proposed method, which is termed expectation-maximization-consensus (EMC) algorithm. The method builds on the iterates (7.26), (7.28), (7.29), (7.20), (7.21), introduced in the previous sections.

The algorithm consists of three major steps, namely:

i The expectation step, where each agent i computes locally the posterior probabilities $\{\alpha_{ikm}^{t}\}_{i,m,k}$ as in (7.8). This is the same step of the centralized approach, since it can be performed locally by each agent, by using only its local information;

ii The maximization step, where the optimal solutions $(\hat{\pi}_i^t, \hat{x}_i^t)$ are computed using (7.26), (7.28), (7.29). Then, the local estimates π_i^{t+1} and z_i^{t+1} are updated via a convex combination of the computed solutions $(\hat{\pi}_i^t, \hat{x}_i^t)$ and the current iterate (π_i^t, x_i^t) throughout the step-size sequence γ^t;

iii The consensus step aimed at forcing global agreement on variables x_i^t [cf. (7.32)], and to update the local variables $\widetilde{q}_i^t$ and s_i^t [cf. (7.20)-(7.21)].

The convergence properties of Algorithm 5 are illustrated in the following Proposition.

Proposition 6.1: Let $\{\theta_i^t = (\pi_i^t, x_i^t)\}_{i=1}^{N}$ be the sequence generated by Algorithm 1. Suppose that:

i Assumptions A1 and A2 hold;

Algorithm 5 Pseudocode of the EMC algorithm at node i

```
1: **Input** : Local dataset $\{y_{im}\}_{m \in M_i}$, number of nodes $I$, maximum number of iterations $T$, number of classes $K$, step-size sequence $\{\gamma^t\}_t$.
2: **Inizialization:** $\pi_i^0$, $\mu_i^0$, and $P_i^0$; $s_i^0 = \nabla_{x_i} h_i^0(x_i^0)$, $\widetilde{q}_i^0 = (I-1)s_i^0$;
3: **for** $t = 1$ to $T$ **do**;
4:     **Expectation Step**: Compute $\alpha_{ikm}^t$ as in (7.8), for all $k, m$;
5:     **Maximization Step**: Evaluate $\hat{\pi}_i^t$ using (7.26), $\hat{\mu}_i^t = \{\hat{\mu}_{ik}^t\}_{k=1}^K$ as in (7.28), and $\hat{P}_i^t = \left\{\hat{P}_{ik}^t\right\}_{k=1}^K$ solving (7.29). Let $\hat{x}_i^t = \left\{\hat{\mu}_i^t, \text{vec}\left(\hat{P}_i^t\right)\right\}$, and compute
           $\pi_i^{t+1} = \pi_i^t + \gamma^t \left(\hat{\pi}_i^t - \pi_i^t\right)$     (7.30)
           $z_i^t = x_i^t + \gamma^t \left(\hat{x}_i^t - x_i^t\right)$     (7.31)
6:     **Consensus Step**: Agent $i$ collects data $\{z_j^t\}_{j \in \mathcal{N}_i}$ and $\{s_j^t\}_{j \in \mathcal{N}_i}$ from its neighbors and updates $x_i^t$, $s_i^t$, and $\widetilde{q}_i^t$ as:
           $x_i^{t+1} = \sum_{j \in \mathcal{N}_i} w_{ij}\, z_j^t$     (7.32)
           $s_i^{t+1} = \sum_{j \in \mathcal{N}_i} w_{ij} s_j^t + \left(\nabla_{x_i} h_i^{t+1}(x_i^{t+1}) - \nabla_{x_i} h_i^t(x_i^t)\right)$     (7.33)
           $\widetilde{q}_i^{t+1} = I \cdot s_i^{t+1} - \nabla_{x_i} h_i^{t+1}(x_i^{t+1})$     (7.34)
7: **end for**
```

ii the step-size sequence $\{\gamma^t\}_t$ is chosen so that $\{gamma^t \in (0,1]$ for all t, $\sum_{t=0}^{\infty} \gamma^t = \infty$, and $\sum_{t=0}^{\infty} (\gamma^t)^2 < \infty$;

iii the sequence $\{\theta_i^t = (\pi_i^t, x_i^t)\}_{i=1}^N$ is bounded.

Then:

a all the limit points of the sequence $\{\theta_i^t\}_{i=1}^N$ are stationary solutions of (7.4);

b all the sequences $\{\theta_i^t\}_t$ asymptotically agree, i.e., $\|\theta_i^t - \theta_j^t\|_2 \underset{t \to \infty}{\longrightarrow} 0$, for all i, j.

Proof. Algorithm 1 is a special case of the NEXT framework proposed in [105]. Then, under the above assumptions on the network among agents, and the algorithm's parameters, all conditions of Theorem 3 in [110] are satisfied, and the convergence result follows. □

On the choice of γ^t: The conditions on the step-size sequence $\{\gamma^t\}_t$ given in Proposition 1 ensure that the step-size decays to zero, but not too fast. There are many diminishing step-size rules in the literature satisfying such conditions. For instance, the following two rules have been founded very effective in the experiments:

$$\texttt{Rule 1:}\quad \gamma^t = \frac{\gamma^0}{(t+1)^\delta},\ \ \gamma^0 > 0,\ \ 0.5 < \delta \leq 1, \tag{7.35}$$

$$\texttt{Rule 2:}\quad \gamma^t = \gamma^{t-1}(1 - \mu\gamma^{t-1}),\quad t \geq 1, \tag{7.36}$$

with $\gamma^0 \in (0,1]$ and $\mu \in (0,1)$.

7.5 Experimental Results

In this section, the approach is validated by using real-world datasets when the EMC algorithm is compared with the centralized EM and others distributed protocols. Additionally, several toy problems have been inserted to give a graphical visualization of the convergence properties of the procedure.

As the data features have a different physical nature, patterns are normalized column-wise between -1 and 1. A time-invariant, connected and undirected graph is used in 10 different runs of each simulation. In each run, it is used a predefined set of agents, $I = 10$ and $I = 20$, and a random topology graph. In particular, every pair of nodes has a fixed probability $p = 0.4$ to be presented in the graph, according to the so-called "Metropolis" weight strategy introduced in sec. 3.5 . Each node has a local dataset, obtained by partitioning the global one in a random fashion.

All the experiments are carried out using MATLAB R2013b on a machine with Intel Core i5 processor with a CPU @ 3.00 GHz and 16 GB of RAM.

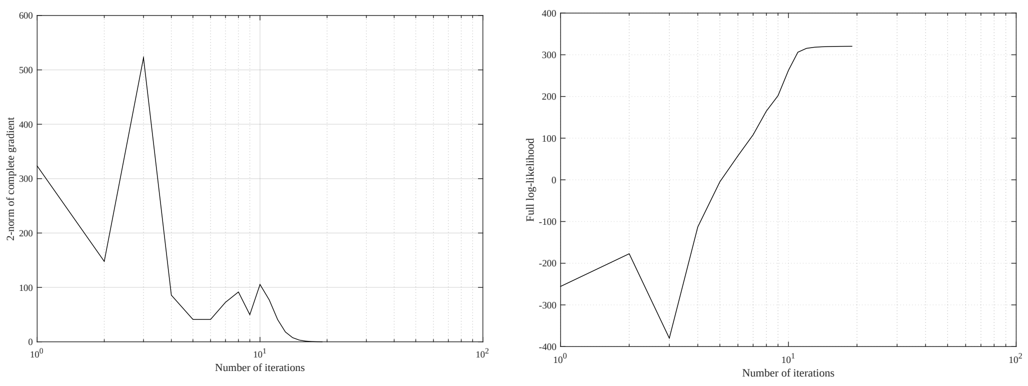
7.5.1 Toy Problems

In this section a graphical visualization of the results, in order to make more intuitive how the procedure works, is given for the toy problems described below:

- *Gaussian Mixture:* in this dataset, the observations are sampled from four Gaussian mixtures with a variance parameter enough small to avoid the classes' overlapping. As it can be seen from Fig.7.1, a stationary point is reached. This is confirmed in Fig.7.2, where it can be seen that each node is able to accurately identify the different clusters.

- *Crescent Moon:* the synthetic dataset is composed of two intuitively separable clusters. The first one is a sphere that could be associated with a full moon, while the other one is a structure identified as a crescent moon. Even if there are only two structures, the number of clusters is set to four in order to capture the underlying moon-shaped non-convex structure of the data. By the way, the estimation of the optimal number of clusters, wither centralized or distributed, is not the scope of this work. As it can be seen from Fig.7.3 and Fig.7.4, the algorithm is able to converge to a stationary point.

- Corner: the dataset contains four linearly separable clusters located at the four corners of a quadratic grid as the one presented in Fig.7.5. The number of clusters is firstly set to four, but as it can be seen from Fig.7.6, the algorithm is not able to accurately model the real structure of the data.



(a) Gradient **(b)** Log-Likelihood

Figure 7.1. Gradient and Log-Likelihood for the Gaussian Problem

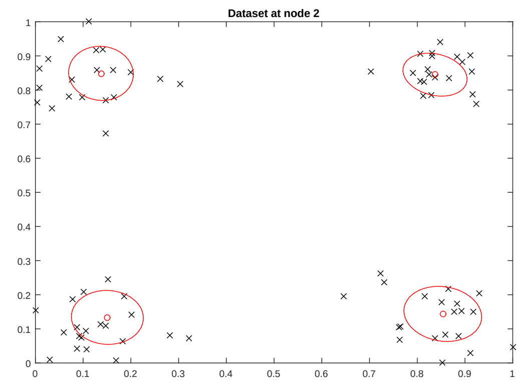
For these reasons, the number of classes has been increased to eight, allowing the algorithm to accurately identify all of the given clusters as it can be shown in Fig. 7.7 and 7.8.

7.5.2 Real-World Datasets

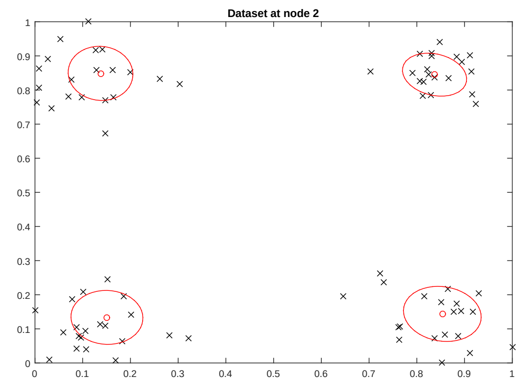
In this section, the performance of the proposed algorithm are evaluated when the following algorithms are used for the comparison:

- *EMC*: the parameters are evaluated in a distributed fashion using the distributed EM algorithm presented in Section 7.4. The algorithm runs for $T = 100$ iterations;
- *Expectation-Maximization algorithm*:[107] this is equivalent to having a centralized agent, where the traditional EM algorithm is performed on the global dataset by using a Gaussian Mixture model. It is used as an optimal benchmark to evaluate the proposed approach;
- *GU*: is a distributed expectation-maximization (EM) approach where a consensus filter is applied to diffuse the local information over the network [111];
- *DEM*: it recasts the centralized problem in a set of smaller local clustering problems with consensus constraints on the cluster parameters. The schemes presented in [112] does not exchange local data among nodes but relies only on single-hop communications.

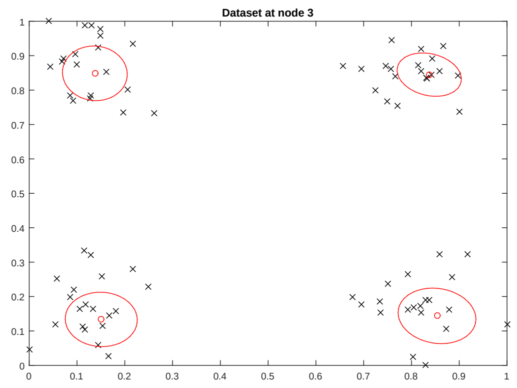
A inner fold cross validation is performed to compute the optimal parameters for all of the used approaches. In particular, for the EMC procedure, a grid-search procedure is executed on the set $\{0.2; 0.4; 0.6; 0.8; 1\}$ for the γ parameter, and on the set $\{0.001; 0.002; 0.005; 0.01; 0.02; 0.05; 0.1; 0.2; 0.5\}$ for λ. For the GU approach, the set $\{0.001; 0.002; 0.005; 0.01; 0.02; 0.05; 0.1; 0.2; 0.5\}$ is used for searching the learning rate τ, and $\{0.001; 0.002; 0.005; 0.01; 0.02; 0.05; 0.1; 0.2; 0.5\}$ for the regularizing factor λ. Finally, for the DEM approach, the learning rate η_μ is searched



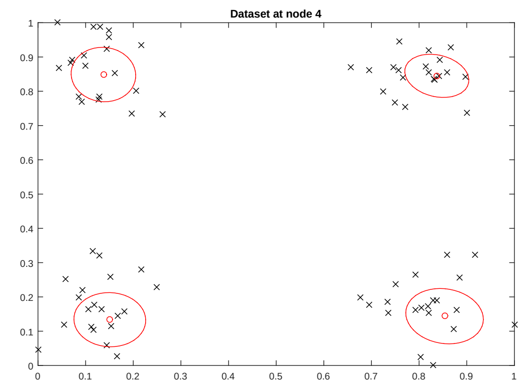
(a) Node 1



(b) Node 2



(c) Node 3



(d) Node 4

Figure 7.2. Mixtures for the gaussian problem

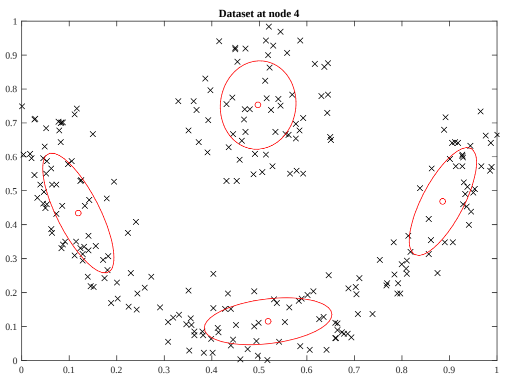
in the set $\{0.001; 0.002; 0.005; 0.01; 0.02; 0.05; 0.1; 0.2; 0.5\}$ set, while η_σ in this one: $\{1; 5; 10; 50; 100\}$. [1]

Three different public datasets available on the UCI repository[2]are considered for the trials. A schematic description of them is given in Table 7.1. In all cases, the optimal clustering is known beforehand for testing purposes, either in the case of classification datasets (where clusters correspond to classes) or because the dataset is artificially generated. Below, some additional information on each one of them are presented:

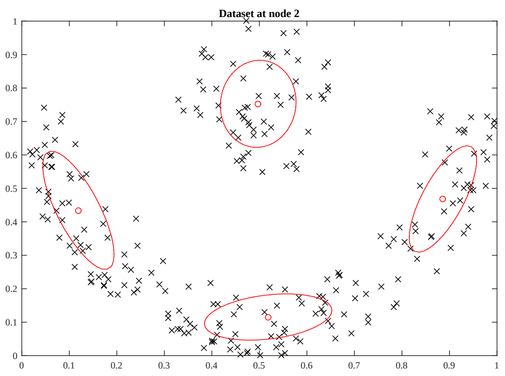
- *Australian credit approval*: [113] dataset is a binary classification dataset, which concerns credit card applications. It is interesting because it contains a good mixture of attributes, both continuous and nominal.
- *Pima Indians Diabetes Data Set*: was already introduced in Sec.8.4.2;
- *Iris Dataset* : whose description is already given in Sec.4.3.

[1]A threshold is added on the covariance matrix's diagonal which, although positive definite in theory, may become numerically singular in practice.

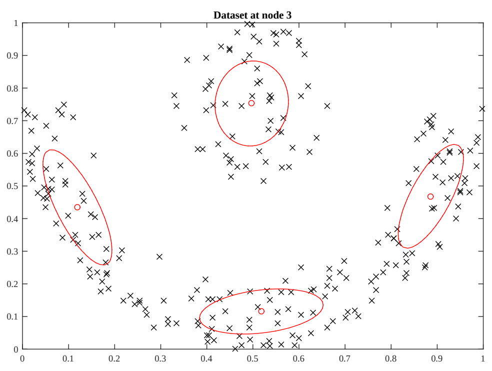
[2]`https://archive.ics.uci.edu/ml/datasets.html`



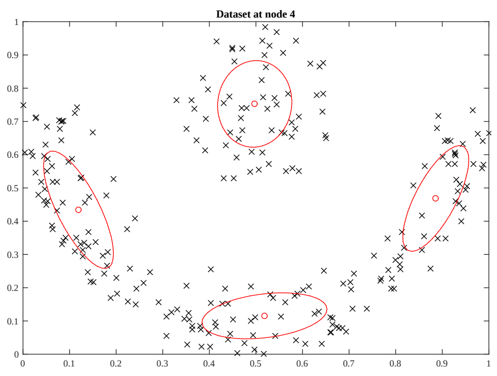
(a) Node 1



(b) Node 2

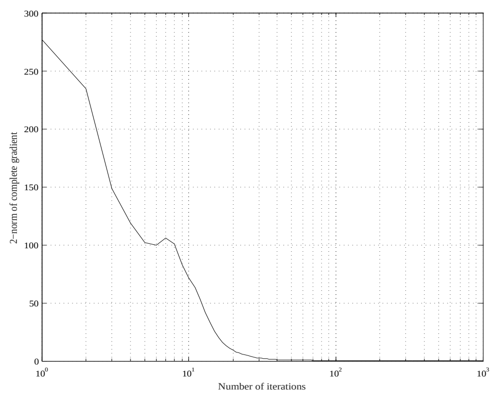


(c) Node 3

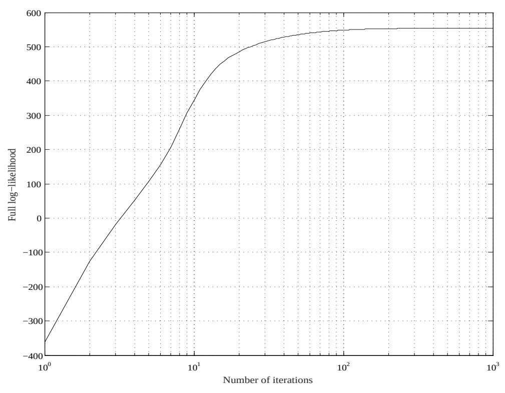


(d) Node 4

Figure 7.3. Mixtures for the Crescent Moon example

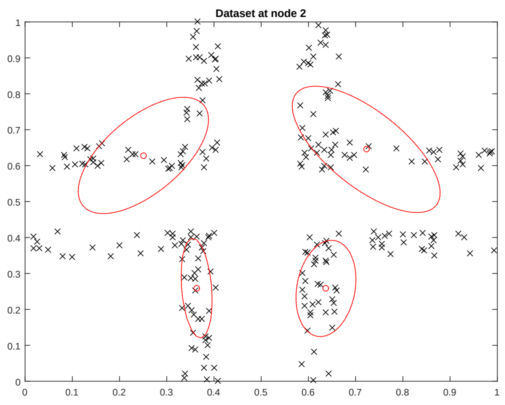


(a) Gradient

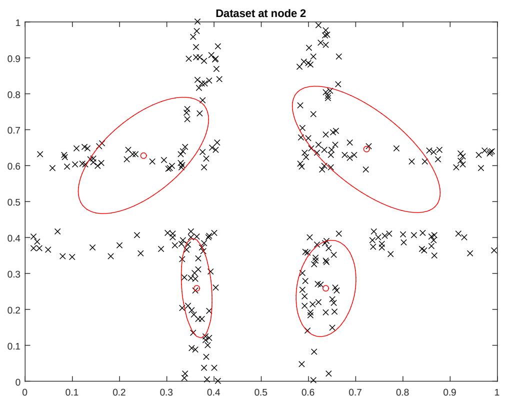


(b) Log-likelihood

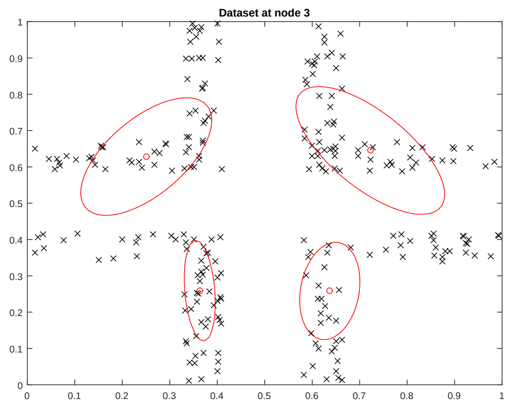
Figure 7.4. Gradient and Log-Likelihood for the Crescent Moon example



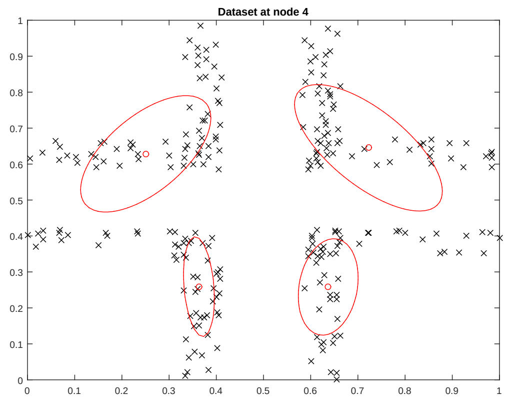
(a) Node 1



(b) Node 2

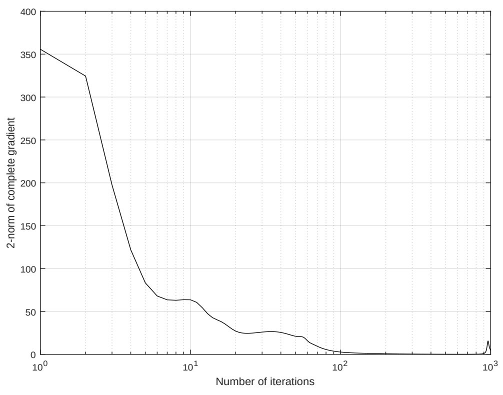


(c) Node 3

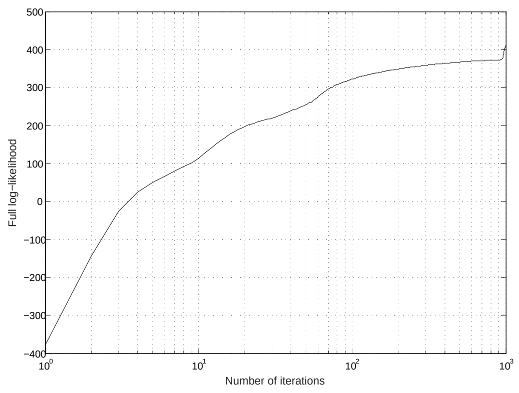


(d) Node 4

Figure 7.5. Mixtures for the Corner example with 4 clusters

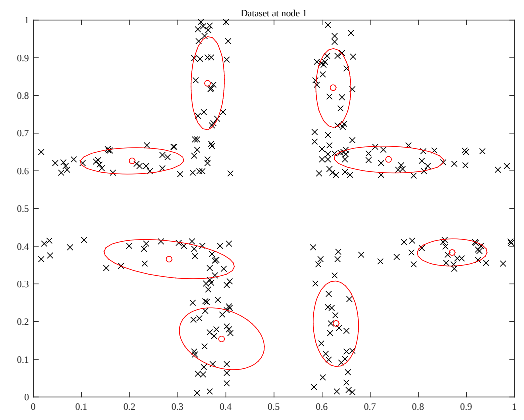


(a) Gradient

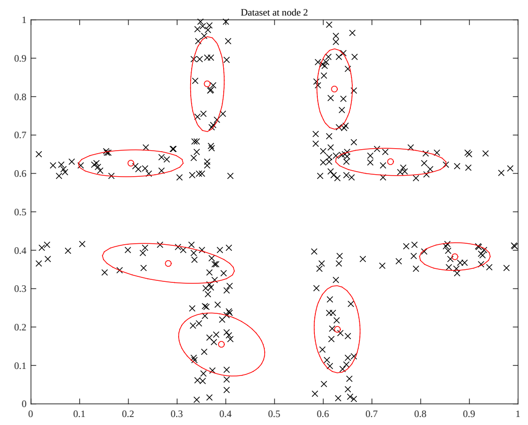


(b) Log-Likelihood

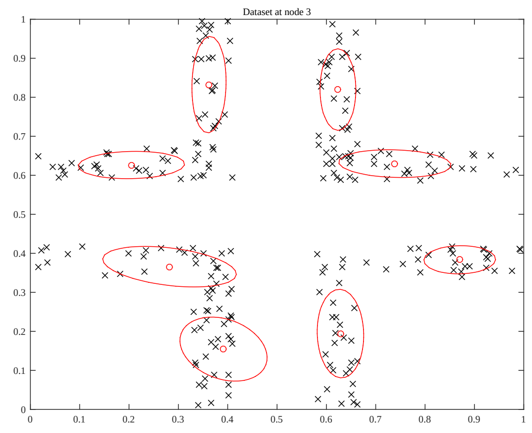
Figure 7.6. Gradient and Log-Likelihood for the Corner example with 4 clusters



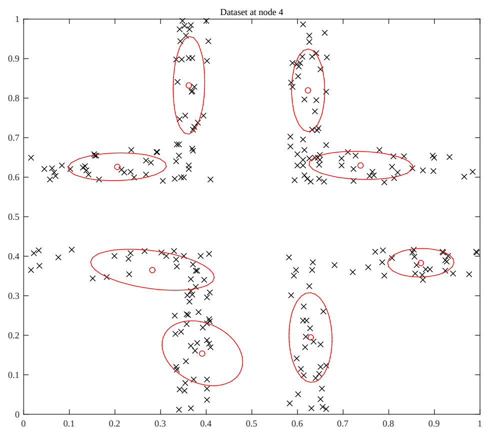
(a) Node 1



(b) Node 2

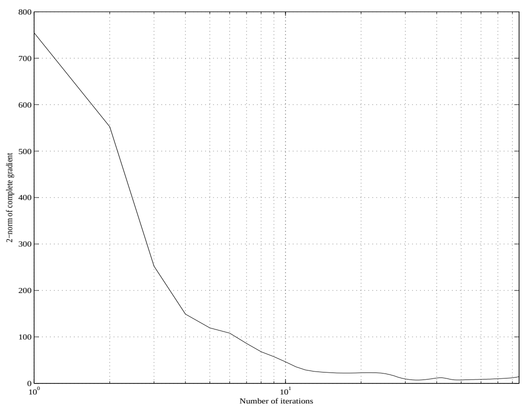


(c) Node 3

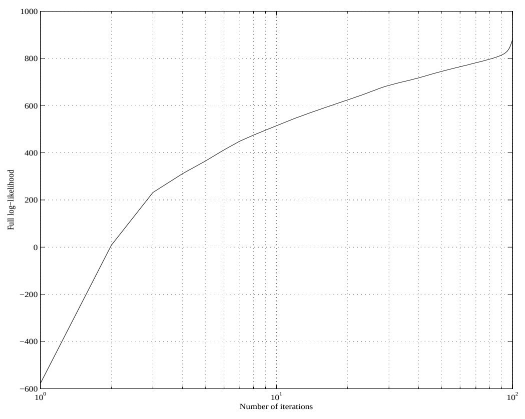


(d) Node 4

Figure 7.7. Mixtures for the Corner example with 8 clusters



(a) Gradient



(b) Log-Likelihood

Figure 7.8. Gradient and Log-Likelihood for the Corner example with 8 clusters

Dataset	Features	Instances	Classes
Australian	14	690	2
Pima	8	768	2
Iris	9	100	2

Table 7.1. Detailed Description of each Dataset

The proposed approach is numerically validated by the use of several quality indexes, detailed in Appendix A.2 and listed below:

- Rand Index [59]
- Falks-Mallows [60]
- F-measure [60]
- K-Index [85]

Precisely all of the indexes range in $[0, 1]$, with 1 indicating a perfect correlation between the true label of the cluster and the output of the clustering algorithm, and 0 the perfect negative correlation. In Table 7.2 and Table 7.3 there are summarized the results averaged over 10 evaluations, and over the different agents, in a network of 10 and 20 nodes respectively. The mean and the standard deviation of each dataset is evaluated, highlighting in bold the best result for each dataset and for each index. It should be noted that the results of the approach are reasonably aligned with the other algorithms. In particular, especially when the network's complexity is increased, EMC can significantly boost performance for almost all the quality indexes. To further strengthen the results, a visual representation of the indexes is given in Fig. 7.9 for the Australian Dataset. As it can be noted, the EMC approach is always better than the others algorithms, while also reaching similar performance with the centralized one.

Finally, to reinforce the conclusion, it is also report the likelihood of the different approaches for all of the dataset, showing what happens when the network complexity changes. The function is evaluated at each iteration by using eq:(7.4), and in Fig. 7.11, the reported trends prove the robustness of the proposed approach and its ability to scale with the size of the network. The same cannot be said for the GU approach; in effect, if in a network of 10 agents it obtains a certain performance when the number of nodes is increased, the trends are always worst. The overall worst performances are obtained by the DEM approach, in both the configuration networks. Conversely, the centralized EM outperforms the other approaches, but this is obvious since it has a complete vision of the dataset.

7.5.3 Time of convergence

This section provides an analysis on the convergence behavior for both the distributed and the centralized approaches. In the former, the convergence time for each node is reported, while in the latter the overall time is considered. The results are analyzed

Dataset	Algorithm	Rand-Index	F-M Index	F-Measure	K-Index
Pima	EM	0.524 ± 0.019	0.594 ± 0.144	0.373 ± 0.125	0.151 ± 0.125
	EMC	0.528 ± 0.020	0.588 ± 0.054	**0.440 ± 0.156**	0.153 ± 0.151
	GU	**0.542 ± 0.025**	0.592 ± 0.044	0.394 ± 0.101	**0.182 ± 0.121**
	DEM	0.528 ± 0.023	**0.646 ± 0.098**	0.378 ± 0.188	0.017 ± 0.044
Australian	EM	**0.615 ± 0.019**	0.648 ± 0.125	0.349 ± 0.125	**0.452 ± 0.010**
	EMC	0.532 ± 0.078	**0.704 ± 0.038**	**0.454 ± 0.250**	0.289 ± 0.222
	GU	0.569 ± 0.053	0.648 ± 0.049	0.390 ± 0.167	0.275 ± 0.206
	DEM	0.603 ± 0.095	0.627 ± 0.084	0.299 ± 0.258	0.402 ± 0.245
Iris	EM	0.786 ± 0.017	0.708 ± 0.025	**0.689 ± 0.057**	**0.605 ± 0.087**
	EMC	**0.802 ± 0.047**	**0.781 ± 0.033**	0.633 ± 0.184	0.589 ± 0.150
	GU	0.664 ± 0.114	0.609 ± 0.154	0.547 ± 0.355	0.300 ± 0.167
	DEM	0.782 ± 0.016	0.735 ± 0.020	0.299 ± 0.258	0.586 ± 0.069

Table 7.2. Experimental results on the different dataset with a network of 10 nodes. The average and the standard deviation of the F-Measures, Rand Index, Kappa Index, FM-Index are used for the EM, EMC, GU, DEM approach. Best results for each algorithm are highlighted in bold.

Dataset	Algorithm	Rand-Index	F-M Index	F-Measure	K-Index
Pima	EM	0.524 ± 0.019	0.594 ± 0.144	0.373 ± 0.125	0.151 ± 0.125
	EMC	**0.551 ± 0.016**	0.627 ± 0.067	**0.424 ± 0.195**	**0.232 ± 0.135**
	GU	0.549 ± 0.003	**0.695 ± 0.007**	0.306 ± 0.150	0.068 ± 0.012
	DEM	0.535 ± 0.018	0.673 ± 0.072	0.361 ± 0.186	0.004 ± 0.035
Australian	EM	**0.615 ± 0.019**	0.648 ± 0.125	0.349 ± 0.125	**0.452 ± 0.010**
	EMC	0.5594 ± 0.097	**0.674 ± 0.066**	**0.393 ± 0.270**	0.346 ± 0.279
	GU	0.558 ± 0.060	0.585 ± 0.053	0.382 ± 0.119	0.285 ± 0.171
	DEM	0.558 ± 0.077	0.607 ± 0.065	0.357 ± 0.195	0.274 ± 0.218
Iris	EM	**0.786 ± 0.017**	0.708 ± 0.025	**0.689 ± 0.057**	**0.605 ± 0.087**
	EMC	0.630 ± 0.199	0.709 ± 0.086	0.540 ± 0.095	0.373 ± 0.258
	GU	0.633 ± 0.063	0.513 ± 0.089	0.529 ± 0.043	0.317 ± 0.114
	DEM	0.743 ± 0.035	**0.781 ± 0.126**	0.620 ± 0.087	0.521 ± 0.083

Table 7.3. Experimental results on the different dataset with a network of 20 nodes. The average and the standard deviation of the F-Measures, Rand Index, Kappa Index, FM-Index are used for the EM, EMC, GU, DEM approach. Best results for each algorithm are highlighted in bold.

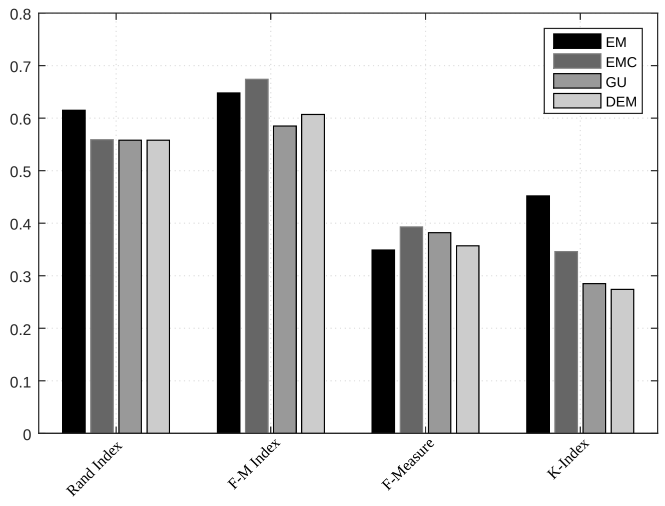
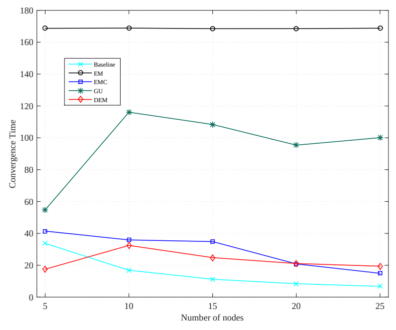
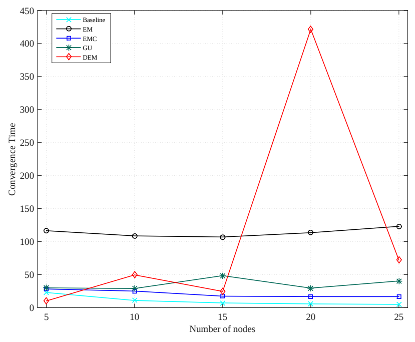


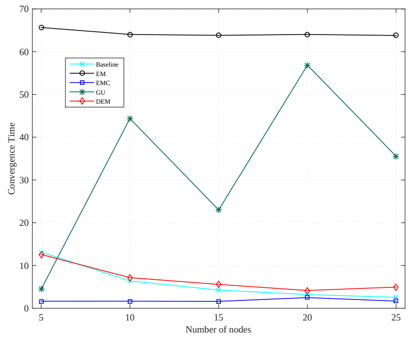
Figure 7.9. The four indexes for all the described algorithm in the Australian Dataset



(a) Australian dataset

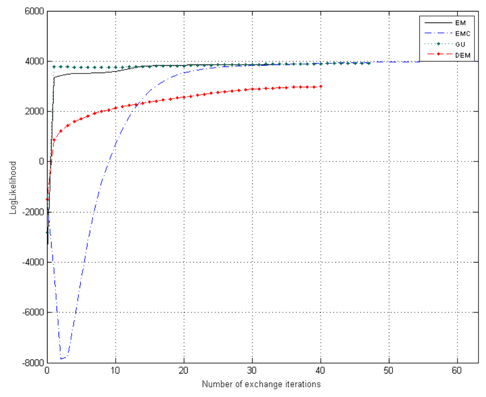


(b) Pima Dataset

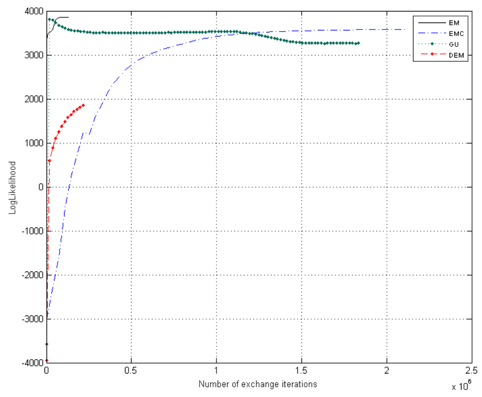


(c) Iris dataset

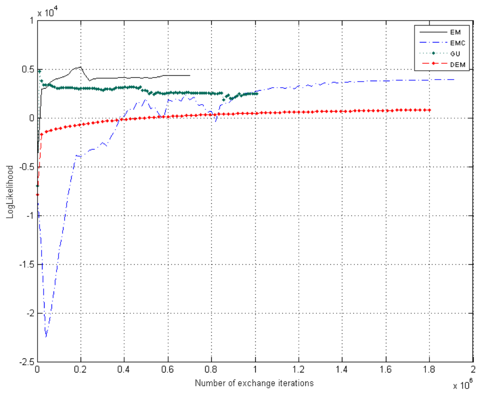
Figure 7.10. Average convergence time for EMC, DEM and GU on a single node. In the centralized procedure, the behavior is not represented as a straight line cause the different initialization that impact on the time of convergence. A theoretical baseline obtained by diving the centralized time by the number of nodes is added.



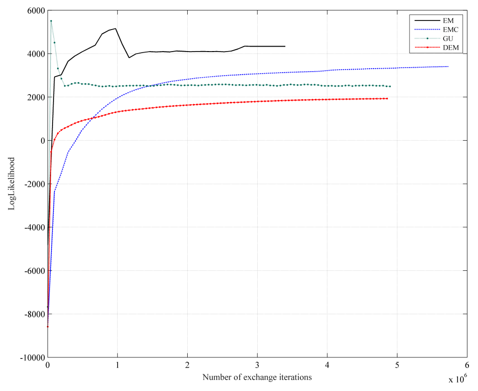
(a) Pima 10



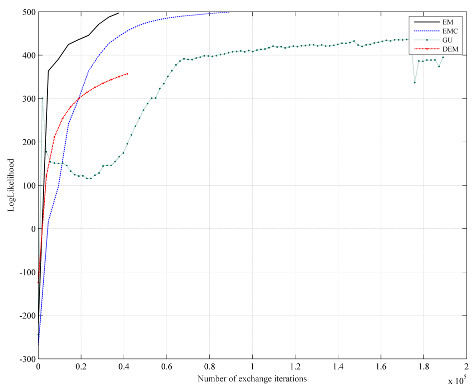
(b) Pima 20



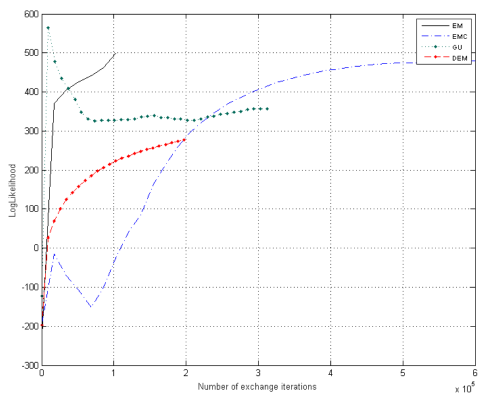
(c) Australian 10



(d) Australian 20



(e) Iris 10



(f) Iris 20

Figure 7.11. The panel on the left shows the evolution of the likelihood for a network of 10 nodes, while the panel on the right shows the behavior of the Likelihood in a network of 20 agents

in terms of number of nodes, which is increased from 5 to 25. As it can be seen from Fig.7.10 the distributed approaches are faster then the centralized procedure, which requires more time for analyzing a bigger quantity of data. A theoretical baseline is also reported. It represents the convergence time of the centralized approach, divided by the number of agents. By analyzing the results, it could be said that the EMC approach obtains the best results. Its performances are comparable with the baseline approach, reporting a slight bigger time that is probably due to the consensus step that is not present in the fully centralized EM. Additionally, the average training time is monotonically decreasing with respect to the overall number of nodes. This is not so trivial because, even if it could be expected that the convergence time becomes lower as the number of nodes increases (because each one of them has to analyze a lower number of data) the time required to find an agreement could becomes bigger if the consensus step is not so optimized. This is the case of the GU approach, which obtains the worst performances and an average time that is certainly non decreasing. The DEM technique is slightly better compared to the GU one, especially for the Iris Dataset; but in the Australian and Pima dataset it could be shown how the error becomes bigger and not decreasing for some network configurations.

7.6 Observation

In this chapter, it has been detailed a distributed algorithm based on a Gaussian mixture approach. It exploits successive convex approximation techniques while leveraging dynamic consensus as a mechanism to distribute the computation as well as propagate the needed information over the network. Experimental results show that the proposed algorithm is able to match very closely the performance of the centralized approach in terms of accuracy and speed, as well the ones of others recently distributed EM versions.

Clearly, the algorithms presented until now can be successfully applied to distributed learning problems laying outside this specific applicative domain, particularly in real-world big data scenarios. This is the main motivation of the successive chapters, where the distributed protocols realized in this thesis have been tested in multiple real-world distributed clustering applications, including low power devices or medical diagnosis.

Part III

Case Studies of Real Applications

Chapter 8

Movement Analysis for tele-rehabilitation

In the previous chapters, the distributed learning problem has been deeply investigated from a theoretical point of view, analyzing and detailing some algorithms ad-hoc realized for this scenario. In the next sections, the focus will be given on real-world data contexts and applications where these techniques are particularly useful. To this end, in this chapter new techniques based on evolutionary intelligences and exhaustive features selection will be presented. Successively, specific focus will be given to some privacy-preserving problems that have to be dealt with when data is distributed among several clinical parties, and moved among them.

In effect, medicine and e-health are two of the most prolific areas for the application of data mining methods, with successful implementations ranging from clustering of patients to rule extraction for expert systems, automatic diagnosis, and many others. Considering the problem of training a classifier to perform automatic diagnosis of a specific disorder (e.g. cancer) starting from standardized medical measurements' set, it could happen that different hospitals have access to historical training data relative to disjoint patients, and it would be highly beneficial to collect these separate sources in order to train an effective classifier. At the same time, however, releasing medical data to a central location (to perform training) generally goes against a number of problems in terms of privacy attacks even if identifiers are removed before releasing it.

This could be a typical scenario of tele-rehabilitation, or telemedicine where several tele-rehab points, like the patients' home, could be linked together in order to produce a network of agents wherein data (e.g. records from different patients) is distributed among multiple clinical parties and the aim is to extract useful information with machine learning techniques and computational intelligence approaches. The ultimate target of this research, concerns the realization of a new multimedia ICT platform that will allow users to access healthcare services for self-rehabilitation of motor disabilities directly from their home. The first steps of the research will be deeply investigated in the next sections.

8.1 Problem Statement

The previous chapters, analyze the distributed scenario where data to analyze is collected across a network of agents. In this section, it is thought challenging to insert this theory in a real-world scenario, showing the potentiality and the opportunities that this reality carries with it. For this purpose, the collaboration with the Biomechanics and Movement Analysis Laboratory, Physical Medicine and Rehabilitation of the University of Rome “La Sapienza” help us point out a typical problem that clinicians usually face in their daily medical lives.

In particular, the number of patients needing rehabilitation has increased in recent years, and several long-term disabilities like Stroke, Parkinson’s disease, Multiple Sclerosis, or neurological disorder require new methods of rehabilitation and care management. Stroke, for example, is one of the serious long-term disability’s leading cause in the adult population and motor disability following a stroke is often due to poor arm function [1]. People lose their ability to perform motor functions as a result of a reduced neural drive from the cortex to motor units, and to a de-synchronized firing rate of motor units. However, if on the one hand, recent researches show that intensive rehabilitation reduces impairments and disabilities even in the chronic stages of disabling conditions, on the other hand, clinicians have to get away from the patient at home before concluding the rehabilitation treatment, treating them in their disease’s acute phase only.

As a matter of fact, the health system is highly centralized since high costs, large spaces that they require, poor transportability and direct experts executing the analysis make this system suitable only for laboratory setting. The idea underlying this project is the realization of an autonomous rehabilitation system, which can be deployed outside hospitals in a distributed setting with reduced or no medical supervision. In particular, a network of hospitals and health centers could allow to monitor patients directly in their home, improving both the quality patients’ life and the accuracy of a therapy.

On this issue, I am currently working on a project involving a new multimedia ICT platform able to collect, integrate, and process data streams through a network connection between clinicians and patients. It is essentially based on the following items:

- low-cost Inertial Measurement Unit (IMU) sensing devices based on accelerometer and gyroscope measurements;
- video acquisitions using smart cameras, depth sensors or RGB-D devices (e.g., the Microsoft Kinect technology);
- data fusion and pattern recognition techniques for movement analysis and classification;
- a multimedia platform for remote control and communication with a Medical Centre.

As illustrated in Fig. 8.1, a monitoring and detection system has to be installed at the patient’s home. It is based on one or more RGB-D devices and wearable IMU sensors, by which the data processing system will be able to track his movements, his

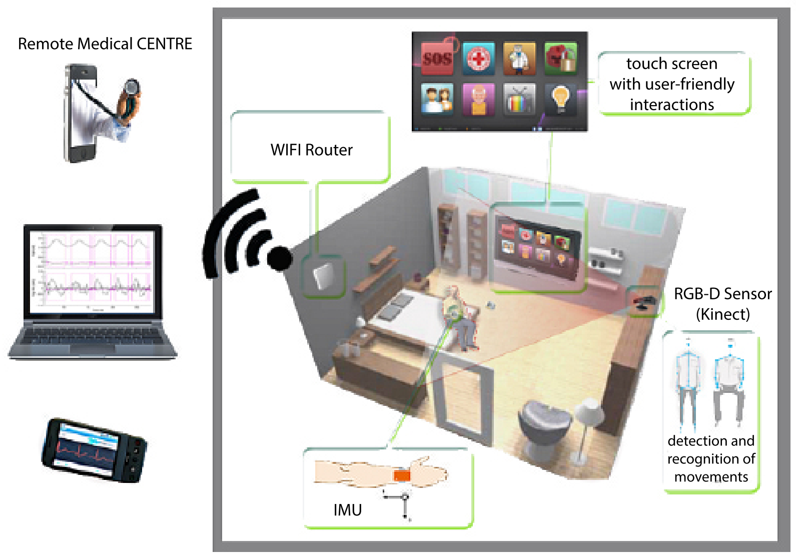

Remote Medical CENTRE
touch screen with user-friendly interactions
WIFI Router
RGB-D Sensor (Kinect)
detection and recognition of movements
IMU

Figure 8.1. Theoretical operating scenario

clinical status and the degree of rehabilitation without the direct, in situ supervision of a doctor or a physiotherapist. A network gateway will be used to transmit the sampled information, and the doctors will be able to see the data, remotely providing the patient with a useful feedback.

The recorded information is inherently heterogeneous and it is represented in heterogeneous space-time domains. This is the case of information related to sensor data. In order to effectively cope with this heterogeneity, appropriate algorithms with high local and distributed intelligence and capable to classify, select, aggregate, fuse, and encode the information to transmit are also needed. In this section, it is proposed a preliminary work where appropriate techniques, based on computational intelligence and machine learning approaches, are realized for this purpose. Using such techniques, which are also adequate for parallel and distributed computing, it is automatically possible to identify data structures, and generate data-driven validation models through a “black-box” approach, without considering the underlying physical and neurological processes that originate the data.

8.2 Gait Analysis

As stated previously, the goal of this research is the realization of a tele-rehabiliation system able to monitor motion for patients affected by specific neurological diseases. Nowadays, in the adult population, an increasing number of pathologies are the leading cause of serious long-term disabilities, such as stroke, Parkinson’s disease, multiple sclerosis and so on. Most of them affect postural control and mobility, and therefore, a careful gait analysis could be the useful indicator for clinicians to form an assessment. The gait analysis is a particular movement test used in several research fields as biomechanics, robotics or sports analysis. However, the most commonly employed application is in the rehabilitation context where it is used to check the biomechanics of patients' walking in addition to kinematic, dynamic and electromyographic parameters. In this way, clinicians are allowed to evaluate impairment's degree of a patient.

The gait is described thanks to the coordinates of the main joints of the body. It is presented as a cycle, where walking's sequences are illustrated in Fig. 8.2. In a more detailed classification, the gait is divided into three main phases and nine sub-phases, whose names are self-descriptive, and based on the movements of the foot. A brief description of each one is given below:

Stance Phase: is the main phase that involves about the 60% of a gait cycle. Starting from the heel phase, when the heel touches the ground, and the toes do not yet, it represents the phase in which the foot and the leg bear the body weight. It lasts immediately before the detachment of the fingers from the floor and can be summarized in the following sub-phases:

- Heel Strike: it represents the starting point of a gait cycle during which the body's center of gravity is at its lowest position;

- Foot Float: it represents the moment in which the plantar surface of the foot touches the ground, and the body begins absorbing the impact of the foot by rolling in pronation.

- Mid Stance: it represents the moment when the body's center of gravity reaches the highest position since the body is supported by a single leg (the controlateral foot passes the stance phase).

- Heel-off: it represents the moment in which the heel leaves the floor. During this phase, the triceps surae muscles cause the push-off and the plantar flexes the ankle.

- Toe-off: it represents the ending point of the stance phase, during which the toes leave the ground.

Swing Phases: involves the 40% of a gait cycle, and lasts from the 'toe off' to the successive contact of the foot with the ground. It is composed by the following subphases:

- Acceleration: it represents the moment in which the foot leaves the ground and the hip flexor muscles accelerate the leg forward.

- Mid Swing: it represents the moment in which the foot goes under the body, while the controlater is in the midstance phase.

- Deceleration: it represents the moments in which the muscles try to slow the leg and stabilize the foot in preparation for the next heel strike.

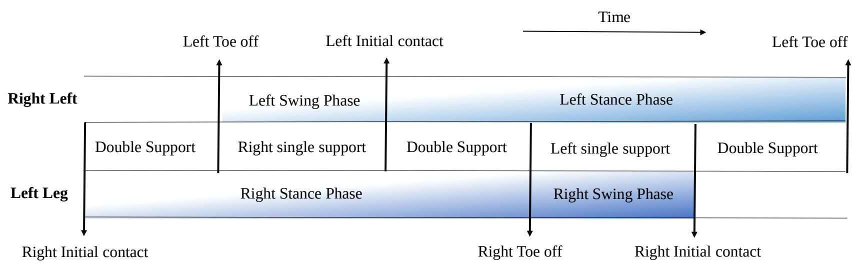


Figure 8.2. Gait cycle

Double Support time: it appears within the previous two phases, and it is the time in which both feet are in contact with the ground. In each double support phase, there is a foot leaning on the ground, and the other that has just left it. In the double support phase, the leg in front is usually known as the "leading" leg and the leg behind is the "trailing" leg. So, in each gait cycle, there are thus two periods of double support, and two periods of single support.

8.2.1 Kinematics Analysis

The gait kinematics is the study of measures involved in evaluating the spatial-temporal movements in the three planes of the space: frontal plane (Abduction/Inversion), sagittal plane (flexion/extension) and transverse plane (Int rotation/ext rotation). The sagittal plane lies vertically, and divides the body into right and left parts, the frontal one lies vertically, and divides the body into anterior and posterior parts, the transverse one lies horizontally, and divides the body into superior and inferior parts. Clinically, there is the tendency to emphasize on the frontal plane motion during the gait analysis, while the others two are ignored. However, there is a group of pathologies where the two other planes could give useful information. The cyclic nature of the gait is a potential instrument to extract different parameters from it. In literature, there are hundreds of parameters expressed in terms of the percent cycle, but often, there are more features than those really required, which are those used by clinicians. For this reason, thanks to the experience and the collaboration with the Biomechanics and Movement Laboratory, the most important ones, for the scope of this work, are described below:

Spatial Parameters: they are related to the length of the stride and the step. From these measures, it is possible to extract information on the frequency, the symmetry and the speed of the gait.

- *Step Length* [m] is the distance traveling from the heel contact of one foot to the heel contact of the controlateral foot. It is correlated to the swing phase so that a short swing phase will generate a short step length. In a pathological gait, the step length is often reduced. In particular, for those pathologies that affect one side of the body, the amount time spent on the "bad" foot is generally shorter than the corresponding one on the "good" foot. Therefore, a short step length on a foot generally means a problem on the controlateral one.

- *Stride Length* [m] is the distance traveling from the heel contact of one foot to the successive contact of the same one. It is composed of two steps length, left and right, each one of them is the foot's distance which moves forward compared to the one left behind, and generally is the equivalent of a gait cycle. It does not actually matter which instant is chosen, but it is very common to find that it starts with the foot contact. Generally, in a healthy gait, the stride length for one side must be the same as that for the other one. While, in a pathological gait the length of the two steps is different, or could have zero values (when the foot is put next to the other, rather than in front of) or negative ones (when one foot does never reach the other one). It can be measured directly, or indirectly by the ratio between the mean speed and the cycle time.

- *Base width* [mm] represents the side-to-side separation of the feet and assumes the same values for both the left and the right steps. Usually, it is considered as the midpoint of the back of the heel and is evaluated in millimeters. Generally, this parameter is evaluated by the distance from the heel, which can be easily measured from the video.

- *Swing Speed* [m/s] is the speed during the Swing phase. It is evaluated by using the stride length and the swing duration.

- *Mean Speed* [m/s] is the distance covered by a person in a given time. It can be evaluated as the sum of stride lengths divided by the cycle time, or as the product of the stride length and the cadence.

In Fig. 8.3 are reported the step length, the stride length and the step width for both the side.

Temporal Parameters: they are related to the different lengths of the step phases, both on the left and the right side.

- *Stance Phase* starts from the heel contact to the toe-off of the same foot. It can be expressed in second, or as a percentage of a gait cycle.

- *Swing Phase* starts from the toe-off of a foot to the heel contact of the same one. It is expressed as a percentage of the gait cycle.

- *Stride Phase* is the time of a gait cycle of a foot expressed in second.

- *Double support* is the percentage of gait when both feet are in contact with the floor.

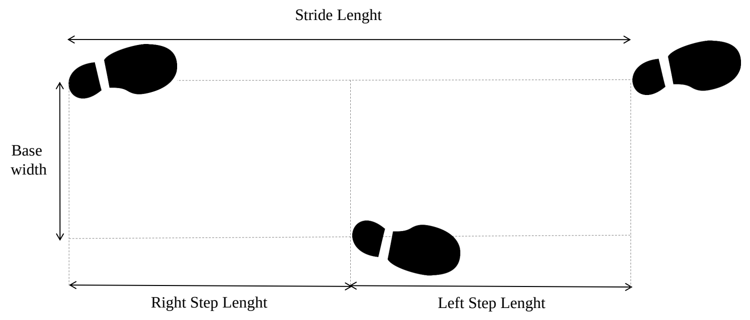


Figure 8.3. Length of step, stride and width of both feet

- *Cadence* [step/min] is the number of steps executed in a given time. It refers to the number of strides per minute, even thought it is often expressed as the number of steps per minute. Generally, it is better to refer to the cadence instead of the stride time, which is the duration of one gait cycle.

Therefore, the kinematic analysis is concerned with the motion analysis without considering the forces involved during the gait.

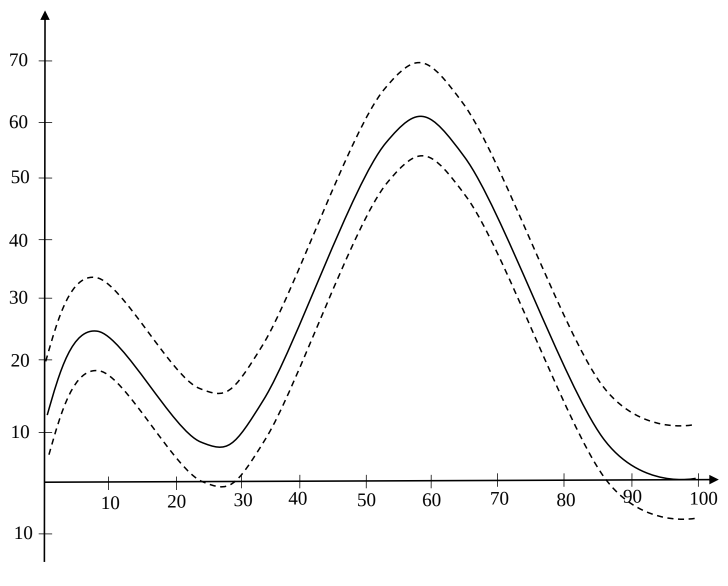


Figure 8.4. Knee motion, normal range during a gait cycle for free walking

8.2.2 Kinetics Analysis

Kinetics is the study of net forces, moments, mass and acceleration executed by the body. It is not related to the position or the orientation of the object involved, but it focuses on the ground reaction forces, inertia and muscle contractions.

Generally, forces platform are used to collect ground reaction forces, which are analyzed in the three main planes, divided into the vertical, front-back and median-side components. Combined with the kinematic analysis, they provide a useful instrument for studying moments and powers of specific articulation. Most biomechanical data characterizing human movements appear as temporal waveforms representing specific joint measures such as angles, moments, or forces. The three main behaviors used by the doctors are explained as follows:

Knee Angles: during each stride, the knee passes through four arcs of motion, with flexion and extension that occur in an alternating fashion. Fig. 8.4 shows the knee flexion angle waveform for a normal subject, normalized to 100% of the gait cycle. It can be noted a regularity in the movement, with two peaks occurring in correspondence of the maximum knee flexion during the stance phase, and the maximum knee flexion during the swing one. Generally, the angle flexion peak is reached between the $60 - 80\%$ of the gait cycle, with a value around 50°.

Ankle Angles: The ankle angle is evaluated as the angle between the tibia and an arbitrary line in the foot. Conventionally, the initial contact is considered as the neutral position, and the ankle angle is put to 0°, even when it is around 90°. Successively, the movement in positive or negative directions occurs with the dorsiflexion or the plantarflexion. Fig. 8.5 shows the ankle flexion angle waveform for a normal subject, normalized to 100% of the gait cycle. As it can be seen, the

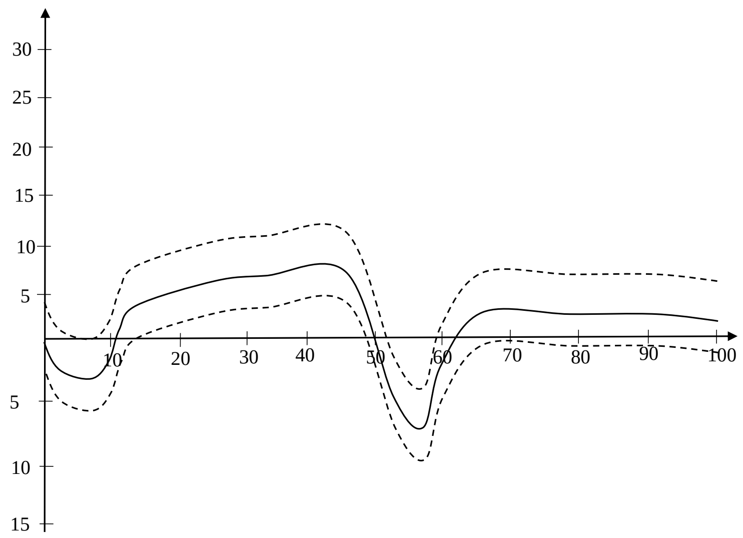


Figure 8.5. Ankle motion, normal range during a gait cycle for free walking

ankle motion is not large. It is composed of four arcs of motion, three occurring during the stance phase, and one, the dorsiflexion, in the swing one.

Hip Angles: it could be measured in two different ways, the angle between the vertical and the femur, or the angle between the pelvis and the femur. In this work,

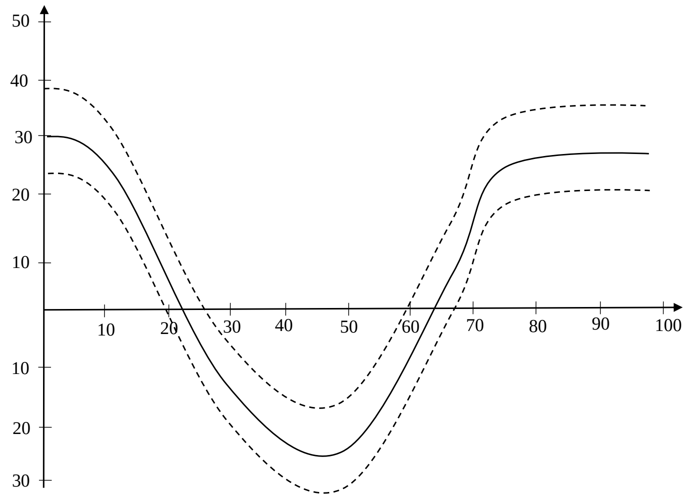


Figure 8.6. Hip motion, normal range during a gait cycle for free walking

the second definition is used. Most of the movement is related to the sagittal plane, where two arcs of motion are encountered: the extension during the stance phase, and the flexion during the swing one. The flexion peak is reached in the middle of the swing phase, with a value around 30°, while the extension peak reaches values around 10°. In fig. 8.6 is shown the knee flexion angle waveform for a normal subject, normalized to 100% of the gait cycle.

8.2.3 Optoelectronic system

The most adopted tools for gait analysis are based on motion capture systems exploiting active or passive markers, electromyography (EMG), inertial systems, electromagnetic sensors, dynamometric platforms and so on. Recently, the best accuracy in gait analysis is obtained by using optoelectronic motion sensors in stereophotogrammetry. Such systems employ several infrared (IR) cameras to record patient's movements, and subsequently, reconstruct and analyze theirs behavior over the whole recording time. A typical scheme of an optoelectronic stereophotogrammetric system is illustrated in Fig. 8.7. In this case, the three-dimensional kinematic data was collected using the 8 camera ELITE stereophotogrammetric system (provided by BTS®, Milan, Italy) sampling at 100 Hz. Data is then filtered using a fourth-order, zero lag, low-pass Butterworth filter with a cut-off frequency of 6 Hz.

Before beginning the gait analysis, some parameters used for the final clinical evaluation, like knee and ankle diameter, basin height and width, need to be taken for each side of the body. Then, 20 retro-reflective spherical markers are applied on the main joints of the subject's body, as illustrated in Fig. 8.8. It was asked to the subjects to wear a minimum number of garments to allow the marker application by means of a double-sided adhesive hypoallergenic. The analysis consists of 4 or 5 trials during which the subject walks at a normal pace, starting at $3m$ of a walkway

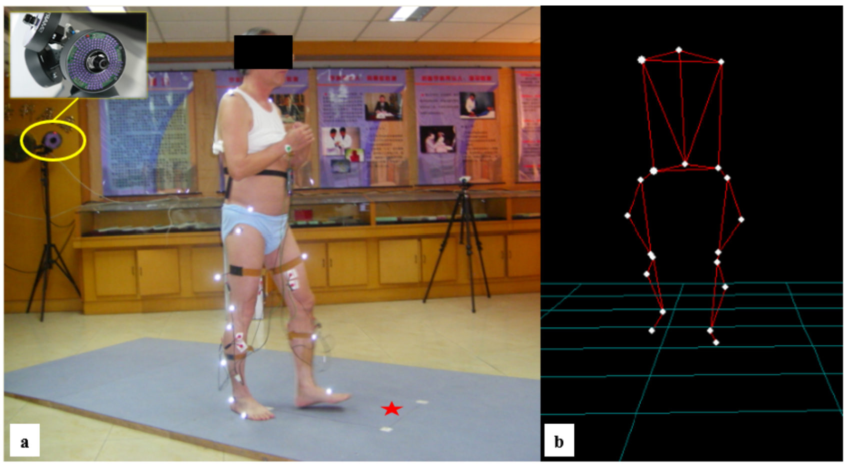

[redacted]

Figure 8.7. Example of a stereophotogrammetric system composed by several IR cameras placed around a dynamometric footboard

$10m$ long and stopping when the end of the footboard was reached.

The data obtained by IR cameras, which point at the markers worn by the walking subject, is integrated into a central computer with those captured by the dynamometric footboard, providing trajectories, angular sizes, speeds and accelerations by thus realizing the final clinical report. So, the system is able to provide information related to the gait cycle that will be used a posteriori to classify the patient's motions.

All of the experimental procedure and the data access were approved by the Ethics Committee of the university hospital.

8.3 Features extraction for pattern recognition

The previous paragraph has introduced some definitions and a software used to produce those clinical features able to characterize the movements of patients with neurological disorders. Since the proposed research aims at finding out a new multimedia ICT platform to deliver healthcare services accessible by the user from his home, the following section will provide some techniques able to manage the captured data.

In this context, clinicians have to face a huge amount of data coming from several locations, which could be the homes of the patients or the different hospital centers. Additionally, the complexity of the problem is increased by the presence of multiple kinds of data (raw data, filtered data, statistical data) captured from different technologies.

For these reasons, decision making for gait analysis will be supported by the use of computational intelligence techniques, machine learning and ad-hoc algorithms that could be able to extract only the actually relevant information useful for the clinical analysis. Some extracted features can serve as potential biomarkers in the definition of a disease, while extra features could increase the complexity of the learning process. In effect, the presence of irrelevant or noisy features might significantly reduce the performance of the synthesized model by impairing the

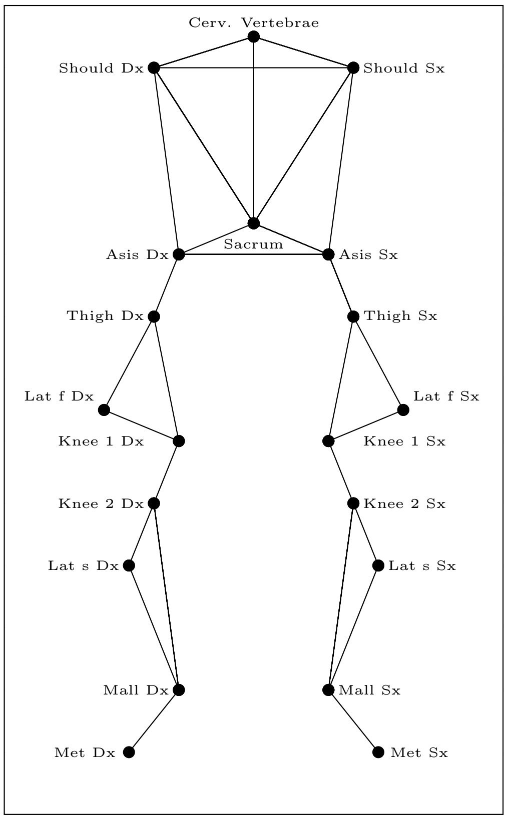


Figure 8.8. The main joints of a skeleton body: seventh cervical vertebrae; acromion clavicular join (Should); anterior superior iliac spine (Asis); Sacrum; lateral aspect of the great trochanter (Thigh); middle point of the lateral aspect of the femur (Lat f); Knee 1 and 2; middle point of the lateral aspect of the shank (Lat s); the lateral malleolus (Mall); the head of the fifth metatarsal bone (Met).

convergence of training procedure and the estimation of the model parameters.

In this case, it is useful to apply techniques allowing to sort the measured features based on their contribution of information in the context of the considered modeling problem. In fact, among all of these features, a specific subset might be more suitable to discriminate among several diseases or to establish the level of functional limitation due to the pathology. In several post-stroke hemiplegia, for example, spatial-temporal variables, like gait speed, represent the most important markers of deficit severity and functional ability characteristics. Similarly, gait in Parkinson disease is mainly characterized by alterations of spatial-temporal features like reduced gait speed and step length, start hesitation freezing and fenestration.

Feature selection is also intended to reduce the dimensionality of the data space, thus increasing the computational efficiency. This stage relies on the choice of a specific procedure able to determine which is the best subset of features. Generally, this is performed by minimizing a particular fitness function able to measure the behavior of the employed classification model. Several approaches can be applied for feature selection [114, 115]; e.g., greedy algorithms based on heuristic search [116, 117], exhaustive search optimization [118], or particle swarm optimization [119].

Also, Principal Component Analysis (PCA), Independent Component Analysis (ICA) and Projection Pursuit (PP) are used to extract relevant information from noisy or redundant datasets. For instance, PCA can reduce a complex dataset to a lower dimensionality with a simplified structure. However, all of these methods, generally perform a transformation of the feature space or a projection of the dataset into a possible, reduced data space, altering the original nature of the data.

In the present case, it is important to maintain the physical and clinical meaning of the measured features representing the dataset, in order to facilitate the identification of the relationship between the outcome of the classification diagnosis and the medical condition of the patient. Consequently, the feature selection task must be performed by using a selection approach working in the original data space, in order to preserve the physical nature of the selected features.

To this end in the next sections a genetic algorithm (GA) and an exhaustive search are adopted for the automatic selection of the meaningful features, considering the misclassification rate of healthy/diseased patients as the fitness function (the lower is better) of the diagnostic model.

8.3.1 Experimental setup

In the experimental process, which will be described in the detail successively, it will be evaluated the gait of 5 sets of individuals. Precisely, four different diseases, which generally present similar impairments in the gait of the subject, are considered: Parkinson's disease, Multiple Sclerosis, Ictus (Stroke), and Coxarthrosis. 85 people who present a gait disorder and 30 healthy subjects with no history of neurological disorders or gait's impairment are called to participate in the analysis. So, a total of 115 people were involved in the research, with 52 females and 63 males and an average age of 56.65 (ranged from 23-86).

The considered features extracted are all the spatial and temporal parameters described in Sec.8.2.1 with the addition of two indexes specifically designed for measuring the dissimilarity of the gait. The Index 1 is evaluated as the ratio between the double support time of the right foot and the left one, while the Index 2 is evaluated as the ratio between the stance phase time of the right foot and the left one. For all of the parameters, both the left and the right values were considered, with the exception of the cadence and the mean speed that have a unique value for both the feet. So, a total of 16 features were processed and, in order, they are the following: Step (R), Step (L), Stride (R), Stride (L), Mean Speed, Cadence, Index 1, Index 2, Width (R), Width (L), Stance (R), Stance (L), Double Support (R), Double Support (L), Swing Speed (R), Swing Speed (L). The characteristics of participants for the five groups are presented in Table 8.1, 8.2.

First of all the features are normalized with a linear transformation in the range between 0 and 1. Let P the number of patterns of the dataset $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_P\}$ and N be the number of attributes of each pattern, that in this case is equal to 16. Thus, each pattern, which is associated to a subject, is represented as a N-tuple of real numbers as follows:

$$\mathbf{x}_h = [x_{h1} \; x_{h2} \; \dots \; x_{hN}] \; , \; h = 1 \dots P \, . \tag{8.1}$$

As the data features have a different physical nature, patterns are normalized column-wise using the following substitution:

$$x_{hj} \longleftarrow \frac{x_{hj} - b_j}{a_j - b_j}\,,\ h = 1 \ldots P\,,\ j = 1 \ldots N\,, \tag{8.2}$$

where $a_j = \max\limits_{h=1\ldots P} \{x_{hj}\}$ and $b_j = \min\limits_{h=1\ldots P} \{x_{hj}\}$.

After normalization, all of the spatio-temporal parameters normally acquired in a biomechanical laboratory are analyzed in order to find the most relevant features able to discriminate between diseased and healthy subjects. To this end, computational intelligence techniques, and evolutionary computation will be used to support decision making for gait analysis.

Group description			Step R	Step L	Stride R	Stride L	Speed	Cadence	Index 1
	N	age	Mean $\pm$ σ	Mean $\pm$ σ	Mean $\pm$ σ	Mean $\pm$ σ	Mean $\pm$ σ	Mean $\pm$ σ	Mean $\pm$ σ
Parkinson's disease	17	64.35 (45-81)	0.369 $\pm$ 0.147	0.361 $\pm$ 0.126	0.794 $\pm$ 0.308	0.718 $\pm$ 0.268	0.580 $\pm$ 0.240	96.122 $\pm$ 14.123	1.016 $\pm$ 0.069
Multiple Sclerosis	29	49.76 (23-69)	0.335 $\pm$ 0.140	0.335 $\pm$ 0.135	0.664 $\pm$ 0.273	0.677 $\pm$ 0.251	0.403 $\pm$ 0.264	67.563 $\pm$ 25.310	1.098 $\pm$ 0.679
Post-stroke hemiplegia	16	56.13 (32-80)	0.334 $\pm$ 0.130	0.351 $\pm$ 0.093	0.681 $\pm$ 0.188	0.742 $\pm$ 0.360	0.408 $\pm$ 0.196	68.457 $\pm$ 18.767	1.015 $\pm$ 0.111
Hip osteoarthritis	23	67.59 (45-86)	0.457 $\pm$ 0.121	0.457 $\pm$ 0.121	0.461 $\pm$ 0.100	0.900 $\pm$ 0.207	0.580 $\pm$ 0.240	96.122 $\pm$ 14.122	1.015 $\pm$ 0.069
Healthy controls	30	31.40 (20-75)	0.638 $\pm$ 0.069	0.635 $\pm$ 0.070	0.701 $\pm$ 0.182	1.270 $\pm$ 0.124	0.910 $\pm$ 0.210	91.989 $\pm$ 13.005	0.991 $\pm$ 0.077

Table 8.1. Cardinality and mean age for each class as well as mean and standard deviation of 7 features for each group.

Group	Index 2	Width R	Width L	Stance R	Stance L	D.Supp R	D.Supp L	Speed R	Speed L
	Mean $\pm$ σ	Mean $\pm$ σ	Mean $\pm$ σ	Mean $\pm$ σ	Mean $\pm$ σ	Mean $\pm$ σ	Mean $\pm$ σ	Mean $\pm$ σ	Mean $\pm$ σ
Parkinson's disease	0.947 $\pm$ 0.476	0.173 $\pm$ 0.118	0.159 $\pm$ 0.099	59.427 $\pm$ 15.554	58.715 $\pm$ 15.688	11.258 $\pm$ 3.995	13.272 $\pm$ 5.357	13.271 $\pm$ 5.357	1.547 $\pm$ 0.584
Multiple sclerosis	1.141 $\pm$ 0.614	0.174 $\pm$ 0.062	0.168 $\pm$ 0.063	68.717 $\pm$ 6.446	68.569 $\pm$ 12.142	20.034 $\pm$ 8.427	19.297 $\pm$ 7.805	1.142 $\pm$ 0.588	1.202 $\pm$ 0.557
Post-stroke hemiplegia	2.222 $\pm$ 1.589	0.178 $\pm$ 0.044	0.171 $\pm$ 0.037	67.788 $\pm$ 10.263	66.875 $\pm$ 8.058	24.700 $\pm$ 16.690	11.969 $\pm$ 4.712	1.249 $\pm$ 0.350	1.180 $\pm$ 0.384
Hip osteoarthritis	0.947 $\pm$ 0.476	0.173 $\pm$ 0.118	0.159 $\pm$ 0.099	59.427 $\pm$ 15.554	58.715 $\pm$ 15.687	11.259 $\pm$ 3.995	13.271 $\pm$ 5.357	1.555 $\pm$ 0.528	1.547 $\pm$ 0.584
Healthy Controls	1.034 $\pm$ 0.417	0.128 $\pm$ 0.033	0.123 $\pm$ 0.039	61.570 $\pm$ 5.207	62.221 $\pm$ 4.449	11.465 $\pm$ 3.845	12.287 $\pm$ 5.213	1.705 $\pm$ 0.452	1.777 $\pm$ 0.463

Table 8.2. Mean and standard deviation of 9 features for each group.

8.3.2 Genetic optimization

First of all it is proposed a technique based on a GA for selecting the optimal subset of features in order to reduce the dimension of the data space, improve the automatic classification accuracy and, above all, support the clinicians in evaluating the most important spatio-temporal parameters when the aim is to identify an anomaly of the gait.

Proposed Genetic Optimization

As stated before, the first underlying idea of the proposed approach is the adoption of a genetic algorithm for the automatic selection of the meaningful features. It is able to optimize a feature selection task by using a fitness function with no reliance on any analytical cost function associated with the stereophotogrammetric, Kinect or IMU measuring system and/or with the physiological model of the patient.

Algorithm 6 Genetic algorithm

```
1: **Initialization**: a population $G_0$ with $P$ individuals is created and set as the current generation.
2: for $k = 0$ to $M_{\mathbf{gen}}$ do
3:     The individuals of $G_0$ are sorted by ascending values of the fitness function.
4:     The next generation $G_{k+1}$ is produced from the current one $G_k$ as follows:
         • The last two individuals of $G_k$ are deleted.
         • The best individual of $G_k$ is cloned and put in $G_{k+1}$ (elitism).
         • The second individual of $G_k$ is mutated with probability equal to $M_r$ by using a 'Uniform' function, then it is put in $G_{k+1}$.
         • do while $G_{k+1}$ contains exactly P individuals:
             – A pair of parents are randomly selected by using a 'Roulette Wheel' procedure. With probability $C_r$, the two parents generate their offspring by means of a 'Two-point' crossover. Each of the two resulting individuals is mutated with probability equal to $M_r$. The two resulting individuals are placed in $G_{k+1}$.
         • end do
5:     The next generation becomes the current one.
6: end for
```

The genetic algorithms are adaptive or meta-heuristic search approaches used for solving optimization problems, and belonging to the particular class of biologically inspired optimization techniques. They start from a random population of candidate solutions, called individuals, and repeatedly modify them in an iterative process obtaining a succession of sets of individuals (i.e. the generations). Starting from the current kth generation G_k, the next one, G_{k+1}, is determined by applying selection, mutation and crossover operators. In other words, in each generation the fitness of each individual is evaluated, multiple individuals are firstly randomly selected from the current population (based on their fitness) and then modified (mutated or recombined) to realize the new generation. In this way, the population 'evolves' over successive generations, toward an optimal solution obtained by the improvement of the fitness of the best individual [120].

In the proposed approach, the genome of each individual is represented by a binary string of $N = 16$ bits: each bit represents one spatio-temporal features and takes value 1 if the corresponding feature will be selected for the classification process and 0 otherwise. The adopted fitness is represented by the misclassification error, which is the percentage of incorrectly classified patterns. Each pattern is labeled either as healthy subject (in the Control group) or a diseased subject (affected by Parkinson's disease, Multiple Sclerosis, Ictus or Coxarthrosis). The details of the adopted GA are summarized in the Alg.6. The behavior of the whole algorithm depends on the values of P and $M_{\mathbf{gen}}$ as well as on the mutation rate (M_r) and on the crossover rate (C_r), which are two probability thresholds that control mutation and crossover operators respectively.

Classification Models

When a subset of spatio-temporal features is identified by the bit string associated with the genome of any individual of the population, its fitness is evaluated by training a classification model using the reduced dataset.

It is important to remark that the genetic optimization proposed in this work is independent of the classification model adopted in this regard, although the final classification performance depends on the chosen model. A number of widely used classification algorithms, whose performance has been already ascertained in many real-world problems with respect to well-known classification benchmarks, have been tested for the sake of comparison [121]. A detailed description of them is given in Appendix C, while they are just listed as follows:

- K Nearest Neighbor (KNN);
- Probabilistic Neural Network (PNN);
- Classification And Regression Tree (CART);
- Naive Bayes classifier;
- Support Vector Machine (SVM);
- Fuzzy Inference System (FIS);
- Linear Discriminant Analysis (LDA);
- Quadratic Discriminant Analysis (QDA).

The experiment protocol has been performed by using a 10-fold cross-validation process [122]. The original data is randomly partitioned into 10 subsets with equal sizes and it is guaranteed that all classes are covered in each subset. For each round, the classifier is trained with the patterns of the training set only, and the number of correctly classified is evaluated using the hold out patterns from the test set. Each fold uses a different subset of validation and the final result is determined using the average misclassification error over the 10 folds. Therefore, the cross-validation output is the misclassification error ϵ:

$$\epsilon = \frac{\widehat{L}}{L}, \tag{8.3}$$

where $\widehat{L} \in [0, L]$ is the number of misclassified patterns (subjects).

Experimental Results

In this section, the obtained resulted of each classifier are summarized. First, it is considered the run for which the best individual is obtained among all the best ones associated with each run. For such a run and for each classifier, the number of generations necessary to reach convergence and the selected features of the best individual are summarized in Table 8.3. Taking into account all the simulated runs,

Classifier	Generations	Selected Features	Number of Features	Best Fitness ($\mu \pm \sigma\%$)	Average Fitness ($\mu \pm \sigma\%$)
PNN	6	0110000000110011	6	2.609 ± 0.000	3.650 ± 0.164
KNN	14	0111010100111111	11	2.609 ± 0.000	3.700 ± 0.248
FIS	8	1110000000001010	5	2.767 ± 0.800	4.857 ± 0.900
Naive Bayes	7	1111000000000010	5	3.339 ± 0.262	4.577 ± 0.210
CART	29	0110010001111010	8	3.399 ± 0.300	5.780 ± 0.864
SVM	6	1111000000111001	8	3.478 ± 0.000	4.470 ± 0.282
LDA	12	1010110110000010	7	4.502 ± 0.519	6.810 ± 0.429
QDA	13	0100110000000010	4	6.403 ± 0.119	8.889 ± 0.127

Table 8.3. Number of generations to reach convergence and selected features of the best individual. Best and average fitness obtained in the final population are also reported.

the mean and standard deviation of the fitness of the best individual and the mean and standard deviation of the average fitness of the final population are also reported in Table 8.3. A discussion regarding the overall results is reported in below, splitting the classification models into three different subsets according to the fitness values.

Classifier Subset #1: KNN, PNN and FIS
The first subset consists of KNN, PNN and FIS classifiers. These models are able to achieve the best fitness value, representing the more suitable choice in terms of classification accuracy among all the tested classifiers.
In particular, PNN algorithm appears as the best one, since it obtains the best fitness of 2.609% with a relatively low number of generations, (only 6). Additionally, it uses a small subset of 6 features only (i.e., left step length, left and right swing speed, left and right stance phase, right stride length) to accomplish this result. Three of these (left step length, right stride length, and right swing speed) are the most employed by all the 8 considered classifiers, thus appearing as the most relevant features for gait analysis. FIS and KNN are able to achieve similar results in terms of best fitness value (2.609% and 2.767%), but increasing the number of generations necessary to reach convergence or the dimension of the features set.
For the sake of illustration, the convergence analysis of the genetic algorithm applied to the PNN classifier is reported in Fig. 8.9, showing that PNN is able to achieve a quick converge with also a limited spread of fitness among the individuals of the final population.

Classifier Subset #2: Naive Bayes, SVM, and CART
The second subset consists of Naive Bayes, SVM and CART classifiers. Theirs fitness values (3.399 % and 3.478%) are very close to the best. All algorithms in this subset

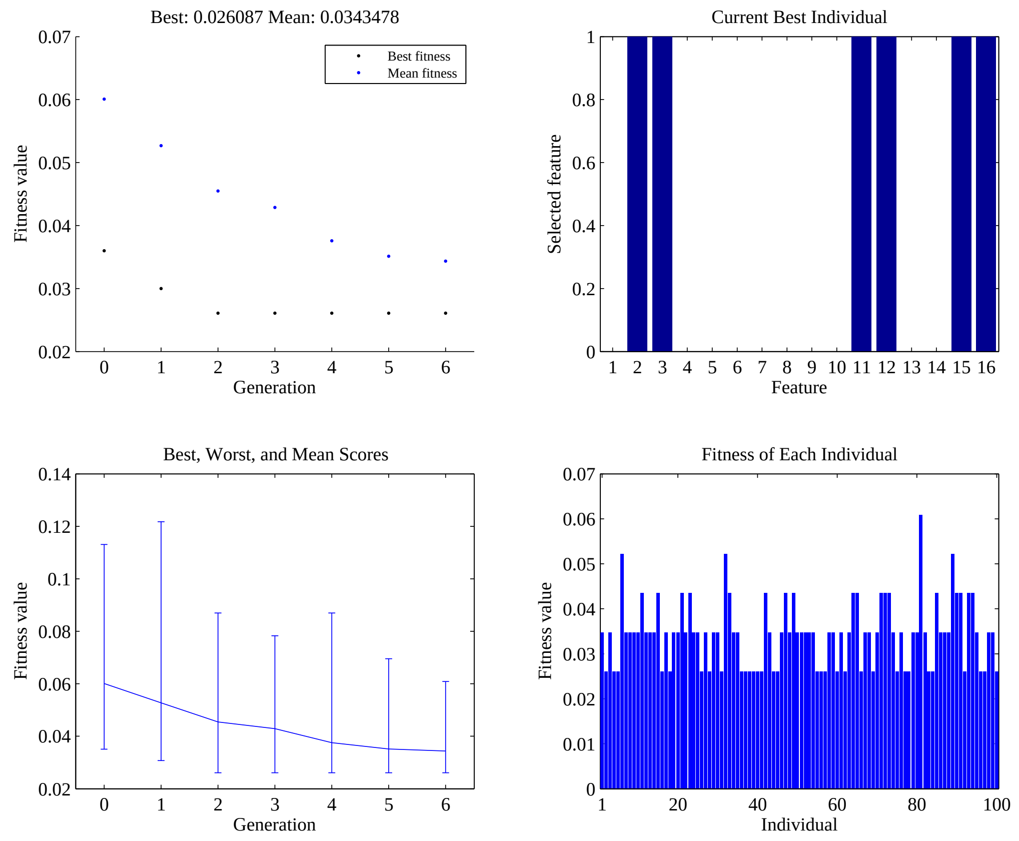


Figure 8.9. Convergence analysis of the genetic algorithm applied to the PNN classifier.

employ a similar number of both iterations and employed features. The last ones are relatively small if are compared with the overall complete set.

Classifier Subset #3: LDA and QDA
The third subset consists of LDA and QDA classification algorithms. They perform the worst fitness value (4.502% and 6.403%). In spite of this, QDA is able to exploit a smaller number of features (i.e., only 4), while LDA adopt a larger number of features.

Final Discussion
Considering the results achieved by all of the employed classifiers, it is possible to compare the different features in terms of number of times that they are selected for classification. In Fig. 8.10 are reported the number of algorithms for which each feature has been considered in the best individual selected by the genetic algorithm. Independently of the left or right side, these results show that the most frequently selected features are:

- step;
- swing speed;
- stride length.

It can be also remarked that such features are also the ones selected by the classifiers that reach the best fitness value (i.e., KNN, PNN and FIS) and, more important, they correspond to the ones considered by the experience of the clinicians when have to perform a diagnosis.

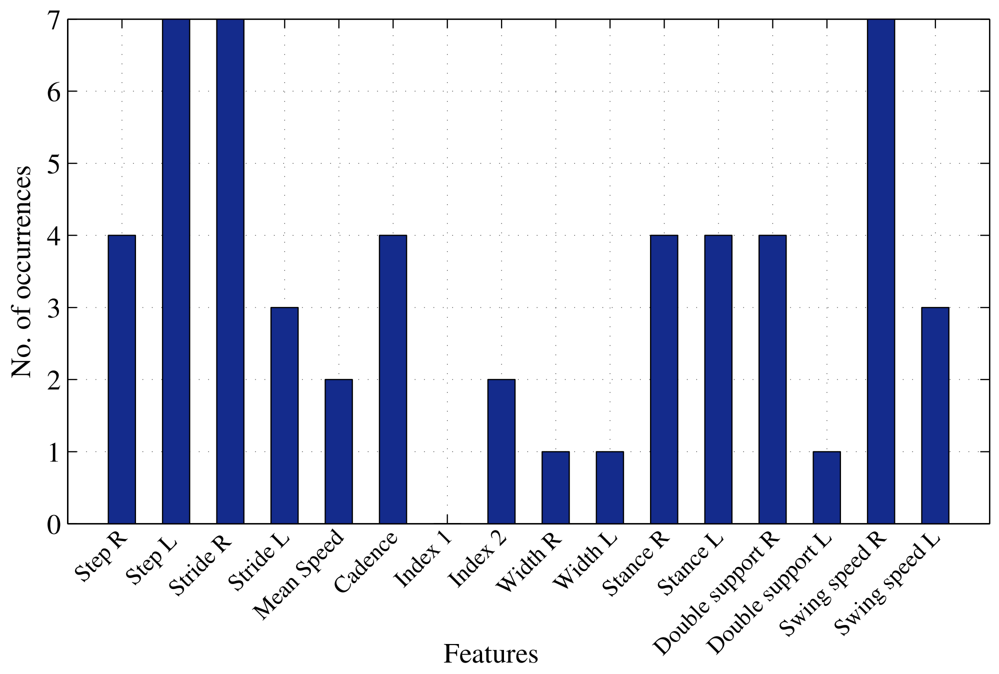


Figure 8.10. Number of algorithms in which each feature has been considered within the best set of selected features.

8.3.3 Exhaustive Features Selection

In the previous paragraph, it has been introduced a genetic optimization to perform a heuristic search able to find a specific subset of suitable features for gait classification. However, since it is an heuristic, it provides only an approximation of the real optimal solution. Now, the interest is in knowing the exact solution of the problem. Thus, firstly a feature selection method able to exhaustively evaluate all the possible combinations of the input features is introduced and then, the procedure to select the best subset one is detailed. In this way, the number of redundant features may be reduced, gaining the maximal performance of the learning algorithm in terms of quality, accuracy and also upgrading the results of the genetic search.

Proposed Exhaustive Search

Also in this case, the considered features are all the spatial and temporal parameters described in Sec. 8.3.1.

The high motion variability due to patient's body function impairment can significantly increase the classification difficulty. Since there is not a unique definition of normal gait, as shown in [123], a feature selection method to exhaustively evaluate all possible combinations of input parameters is proposed in this section. In this way, the best informative subset could be given to the clinicians for help them in making a diagnosis of the subjects.

After normalization, all of the spatio-temporal parameters are analyzed in order to find the most relevant features able to discriminate between diseased and healthy subjects. To describe the dataset a string of 16 bits is used, each bit represents a feature and takes value 1 if the corresponding feature is selected and 0 otherwise. The input set is composed of 16 features so since there are $2^{16} - 1$ possible combinations of them, a total of 65535 subsets of features should be considered.

The correct choice of the most relevant features is crucial so, also in this case, different classification's functions, listed in Appendix C, are used to select the most representative ones. For each possible combination of the input features, the classification model is trained using all the P patterns in $\mathcal{D}$ but considering, for every pattern, the selected features only.

Results are evaluated and compared in terms of the final classification error and confusion matrix.

Validation procedure

The dataset is shuffled into a training set and a test one; precisely, a 10-fold cross validation process is performed where 10 rounds of classification are carried out for each classifier [124]. The numerical parameters to be set in advance for each algorithm, have been determined using an inner three-fold cross-validation on each training subset; they have been selected as explained below.

In the SVM a radial basis function is used with an SVM Karush-Kuhn Tucker (KKT) violation level equal to 0.05 [122]. In the KNN, the value K of nearest neighbors is varied in the range $(2, 10)$ and, by using the Euclidean distance, the final choice becomes $K = 3$. The Naive Bayes classifier adopts as prior the normal distribution with diagonal covariance. Finally, in the FIS classifier the number of Mamdani-type fuzzy rules has been varied in the range $(1, 10)$ by selecting at the end the value of 5.

The results are evaluated in terms of classification rate over 10 folds, which is the percentage of patterns incorrectly classified, and considering the confusion matrix, which gives us the total number of false positives, false negatives and subjects correctly classified. A confusion matrix is a table that contains information about actual and predicted classes. Each column represents the instances in a predicted class, while each row represents the instances in a true class:

- element $\{C_1, \widehat{C}_1\}$ is the number of healthy subjects that are correctly classified as healthy subjects;
- element $\{C_1, \widehat{C}_2\}$ is the number of healthy subjects that are incorrectly classified as diseased subjects;
- element $\{C_2, \widehat{C}_1\}$ is the number of diseased subjects that are incorrectly classified as healthy subjects;
- element $\{C_2, \widehat{C}_2\}$ is the number of diseased subjects that are correctly classified as diseased subjects.

Classifier	Minimum error	Maximum error	Average error	Best subset	Cardinality
KNN	0.020	0.356	0.065	1111001000101101	9
FIS	0.031	1.000	0.795	0111000000001010	5
Naive Bayes	0.035	0.318	0.104	0100000001000010	3
CART	0.039	0.367	0.090	1110011000110101	9
SVM	0.043	0.594	0.190	1100000000000010	3
LDA	0.055	0.566	0.110	1010110110000011	8
QDA	0.064	0.590	0.151	0100110000000010	4

Table 8.4. Classification results in terms of minimum, maximum and average classification error. The last two columns report the best subset of features and its cardinality.

When more than one subset of features yields the same performance, only the one having the minimum complexity is selected. Namely, for equal classification errors, the subset that involves the minimum number of features is considered as per well-known results of learning theory.

The performances of the classification algorithms are also evaluated in terms True Positive (T_P), False Positive (F_P), False Negative (F_N), False Positive (F_P). Namely, they are the basis for the following performance indexes:
Sensitivity: is the proportion of true positives that are correctly identified by the test [125]:

$$SE = \frac{T_P}{(T_P + F_N)} \tag{8.4}$$

Specificity: measures the proportion of true negatives that are correctly identified [126]:

$$SP = \frac{T_N}{(T_N + F_P)} \tag{8.5}$$

Accuracy: is the proportion of the true results, either true positives or true negatives [127]:

$$ACC = \frac{T_P + T_N}{(T_P + T_N + F_P + F_N)} \tag{8.6}$$

Each pattern is labeled either as healthy subject (in the Control group) or as a diseased subject (multiple sclerosis, post-stroke hemiplegia, hip osteoarthritis or Parkinson's disease). This approach aims at showing how the results are influenced by the feature selection, showing also what happens when new information is added to the dataset by the selection of new parameters.

Experimental results

The main classification results are summarized in Table 8.4, where for each classification algorithm, are reported minimum, maximum and average classification error among all of the possible combination of features, as well as the best solution and the related cardinality. Since some algorithms require considerable time to compute the errors, in Fig. 8.11 is illustrated the convergence time for each classification model. It can be shown that KNN, QDA, and LDA are the faster classifiers, while SVM is associated with the worst computational time. In Table 8.5 a better description of

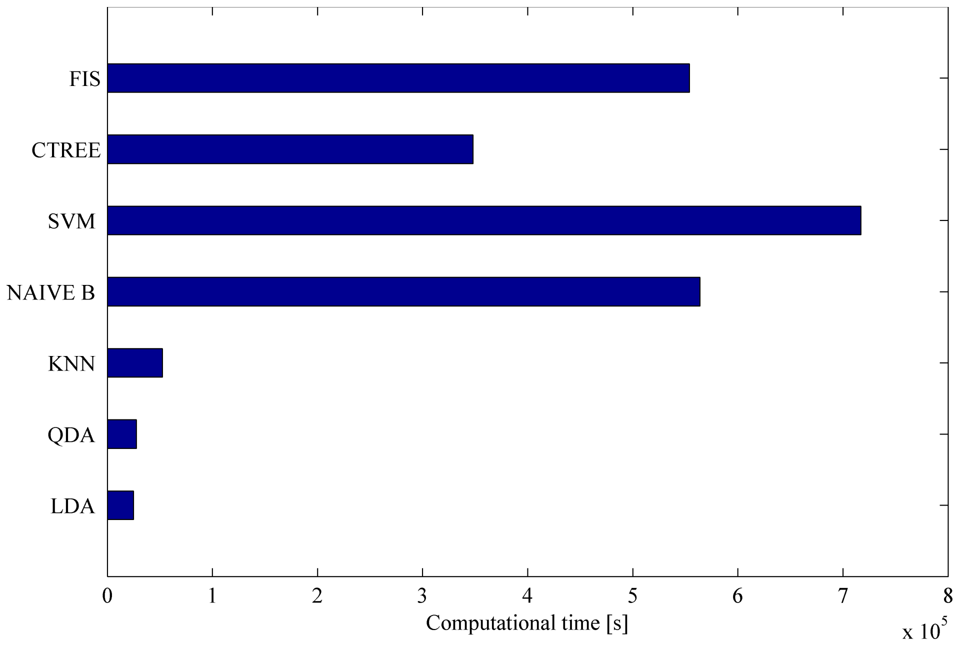


Figure 8.11. Computational time for each classification algorithm.

the input dataset is provided, there are 16 rows (one per feature) and 8 columns (one per each algorithm plus the number of times that each feature is selected in an optimal dataset); each element of the Table takes value 1 if the corresponding feature is selected and 0 otherwise. The number of occurrences reported in the last column of Table 8.5 is illustrated as a histogram in Fig. 8.12. The confusion matrices are reported in Table 8.6, 8.7 for each classification algorithm. As the tests were repeated several times, each entry of such confusion matrices contains average value and standard deviation. Finally, the performance indexes introduced in (8.4)-(8.6) are summarized in Table 8.8. The best feature set for each classification algorithm and the relative classification error is reported in Table 8.4. Only the subsets having minimum complexity are shown, but it is interesting to underline that even when there is more than one subset scoring the lowest classification error, the one with minimum complexity is always unique. A discussion regarding the overall results is reported in the following, splitting the classification models into three different subsets according to the range of classification performances.

Classifier Subset #1: KNN, Naive Bayes and SVM
It is clear from Table 8.4 that the KNN has the best performance in terms of classification error that is around 2.00% only, while the features' set cardinality is not among the best ones since it selects 9 features. The NaiveBayes and the SVM classifiers should be considered better because, despite the small increase of classification error, which reaches 3.5% and 4.3% respectively, they are able to select a small number of features that is equal to 3. This means that with three features only, these algorithms are able to efficiently discriminate among the different pathologies. It is also interesting to notice that the Step L and Swing speed R are selected

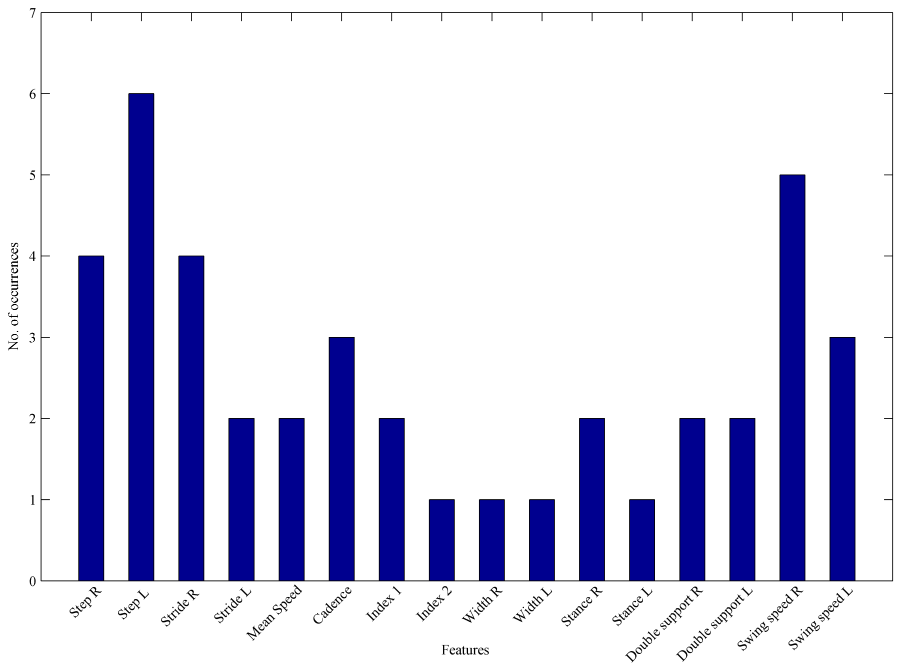


Figure 8.12. Number of algorithms for which each feature has been considered within the best set of the selected ones.

Features set	LDA	QDA	KNN	Naive	SVM	CART	FIS	Occurrences
Step R	1	0	1	0	1	1	0	4
Step L	0	1	1	1	1	1	1	6
Stride R	1	0	1	0	0	1	1	4
Stride L	0	0	1	0	0	0	1	2
Mean Speed	1	1	0	0	0	0	0	2
Cadence	1	1	0	0	0	1	0	3
Index 1	0	0	1	0	0	1	0	2
Index 2	1	0	0	0	0	0	0	1
Width R	1	0	0	0	0	0	0	1
Width L	0	0	0	1	0	0	0	1
Stance R	0	0	1	0	0	1	0	2
Stance L	0	0	0	0	0	1	0	1
D.Support R	0	0	1	0	0	0	1	2
D.Support L	0	0	1	0	0	1	0	2
Swing R	1	1	0	1	1	0	1	5
Swing L	1	0	1	0	0	1	0	3

Table 8.5. Best input dataset for each classification algorithm and numbers of times that each feature is selected in the optimal dataset.

Class	LDA		QDA		KNN		Naive Bayes	
	$\widehat{C}_1$	$\widehat{C}_2$	$\widehat{C}_1$	$\widehat{C}_2$	$\widehat{C}_1$	$\widehat{C}_2$	$\widehat{C}_1$	$\widehat{C}_2$
C_1	28.80 ± 0.42	1.20 ± 0.42	28.00 ± 0.00	2.00 ± 0.00	28.90 ± 0.31	1.10 ± 0.31	28.50 ± 0.70	1.50 ± 0.70
C_2	5.10 ± 0.73	79.90 ± 0.73	5.40 ± 1.07	79.60 ± 1.07	1.20 ± 0.42	83.80 ± 0.42	2.50 ± 0.52	82.50 ± 0.52

Table 8.6. Confusion matrix for each optimal subset and for LDA, QDA, KNN and Naive Bayes algorithms.

Class	SVM		CART		FIS	
	$\widehat{C}_1$	$\widehat{C}_2$	$\widehat{C}_1$	$\widehat{C}_2$	$\widehat{C}_1$	$\widehat{C}_2$
C_1	29.00 ± 0.00	1.00 ± 0.00	27.70 ± 0.68	2.30 ± 0.68	27.80 ± 0.79	2.20 ± 0.79
C_2	3.90 ± 0.88	81.10 ± 0.88	2.20 ± 0.42	82.80 ± 0.42	1.40 ± 0.70	83.60 ± 0.70

Table 8.7. Confusion matrix for each optimal dataset and for SVM, CART and FIS algorithms.

Algorithm	Sensitivity (%)	Specificity (%)	Accuracy (%)
LDA	98.5%	85.0%	94.5%
QDA	97.5%	83.8%	93.6%
KNN	98.7%	96.0%	98.0%
Naive	98.2%	91.9%	96.5%
SVM	98.8%	88.1%	95.7%
CART	97.3%	92.6%	96.1%
FIS	97.4%	95.2%	96.9%

Table 8.8. Performance indexes for the optimal subset and for each classification algorithm.

by both the results. This confirms that these two features have certainly a good capability of discriminating the presence of a gait disorder, in fact, the first is selected by other four algorithms and, the last one, by the majority of them (six out of seven).

Classifier Subset #2: FIS and QDA
Despite these two classifiers select a small number of features, 5 and 4 respectively, the classification error is among the highest values. In fact, even if FIS obtains a minimum error of 3.1%, the maximum error and the average one are poor, (the latter reaches values around 80%). However, even if the QDA obtains the worst minimum error (6.4%) when is compared to the other classifiers, it is also able to to discriminate the subjects with a good accuracy.

Classifier Subset #3: CART and LDA
Finally, CART and LDA present slightly worse results with a bigger cardinality of

the features set, 9 and 8 respectively, and similar behavior on the classification error that ranges from 3.9% to 5.5%.

Final Discussion
The algorithms are tested also in terms of three indexes, in order to evaluate the best one in terms of accuracy. As it can be seen from Table 8.8, KNN confirms its best results, reporting the best values of sensitivity, specificity, and accuracy; in general, the other algorithms do not present very different values of such indexes. The results presented above confirm a high accuracy that allows us to obtain very low errors in discerning among diseased and healthy subjects. In fact, in the worst case of QDA, only 7.4 people out of 115 in mean are wrongly classified. While in the best case of KNN only 2.3 subjects are inserted in the wrong class, as it can be seen in Tables 8.6, 8.7.

The important aspect to underline when different algorithms are tested, is the number of selected features; some algorithms are very efficient considering only 3 features out of the total 16, but in other cases, as for KNN or CART, the performances are not so good because 9 features are included.

An exhaustive search among all of the possible combinations of the input features demonstrates that some of them are more important when the aim is to discriminate between diseased and healthy subjects. It can be noticed, by observing Table 8.5 and Fig. 8.12, that the features most involved in this task are Step L and the Swing Speed R that are selected by almost all the used algorithms (6 and 5 times, respectively). Also, the stride length R, the cadence and the step length are quite informative since they are selected by the majority of the algorithms. This is confirmed by recent works that show the alteration of the step length in the Parkinson's disease patients [128], or of the cadence and the gait speed in the post-stroke hemiplegia ones [129, 130]. Regardless what is often underlying about the importance of the gait speed, [131, 128], in this case, it has not been found as one of the main parameters able to discriminate among pathological and physiological gait.

The two indexes added to the spatio-temporal parameters are important especially for those diseases that affect only one side of the body, where there is a dissymmetry of the gait but also an increasing of the double support time on a single foot. The width or the stance length are certainly the features that bring the minimum information and that should be excluded when the aim is to discriminate among several diseases.

8.3.4 Observation

In the previous paragraph, two techniques able to extract useful information from medical data have been presented. In this regard, an optimization strategy based on GA was firstly introduced in order to estimate the performance of several classification algorithms, both in terms of number of selected features and misclassification error. Results show that the proposed method is able to reduce the dimensionality of the data space and to classify the patient's status with a suitable classification accuracy (higher than 97%). Since the number of features extracted in this study is not so prohibitive, it is also added an exhaustively feature selection search able to evaluate the classification results when all the possible combinations of the input parameters

Classifier	Minimum error GA	Maximum error ES	No feature selection
KNN	0.027	0.020	0.122
PNN	0.027	0.026	0.261
CART	0.039	0.027	0.104
Naive Bayes	0.035	0.035	0.139
FIS	0.035	0.031	0.122
LDA	0.055	0.044	0.130
QDA	0.064	0.044	0.174

Table 8.9. Classification results in terms of minimum error for the GA, ES and a no feature selection approach

are presented as an input to the model. In this way, it has been provided an optimal benchmark to the genetic approach, underlying its potentiality to be used in a more complex context, like the one of tele-rehabilitation, where the high dimension of the input data make impossible the evaluation of the optimal solution in a closed form.

In particular in Tab.8.9 the results of the procedures have been flanked, with also a no feature selection method. It is evident that the exhaustive search is able to provide the best results, however, the performances of the genetic approach are very comparable. Putting together all of the previous results, it can be shown that the several algorithms help us in finding out the best subset of features in discriminating among diseased and healthy subjects. The step length, the swing speed, and the cadence are the most representative features for detecting the presence of a gait disorder since their lonely presence, with two additional features at most, is sufficient for achieving good results in terms of classification accuracy. The number of redundant features may be reduced and the outcomes show a suitable classification accuracy, higher than 97%, which is also confirmed by some performance indexes ranging from 83.8% to 98.8%. The KNN, Naive Bayes and SVM classifiers are the best algorithms for this purpose since they obtain the lowest classification error and the smallest number of selected features. Nevertheless, the proposed approaches make gait classification simple to be solved, providing a direct information about the relevant clinical features and avoiding transformation or projection of them into a different data space.

This should be a great contribution especially in the perspective of a home-based rehabilitation system, providing support to the clinicians in realizing a remote diagnosis for the patients.

8.4 Privacy Preserving data mining for distributed context

In the previous paragraph, it has been introduced a general approach able to detect several diseases and that could help the clinicians and the patients in the early stage of a disease as well as in the successive ones. Data processing procedures are used to combine the spatial temporal data of gait into a limited set of variables. Such model, either based on neural network or not, are able to provide meaningful medical information (such as optimal clustering of the subjects), also when are applied from heterogeneous and spread data like the ones coming from several sensors or different patients. However, in the medical context, each part is forbidden to disclose its local dataset to a centralized location, due to privacy concerns over sensible portions of the dataset. To this end, in this section a framework that essentially extends the one introduced in chapter 6 is detailed. It allows involved parties to perform (in a decentralized fashion) any data mining procedure relying solely on the Euclidean distance among patterns, including kernel methods, spectral clustering and so on. Once all the agents have access to the global estimate of the EDM, many data mining techniques can be applied directly (e.g. spectral clustering) or by simple in-network operations (e.g. SVMs), as it will be discussed subsequently. Additionally, if there is the need of applying more than one technique, the same estimate can be reused for all of them, making the framework particularly useful whenever data must be used in an 'exploratory' fashion, without a particular predefined objective in mind. And this is particularly suited for the medical context.

8.4.1 Techniques for privacy preservation

Starting from the algorithm 3, which is extremely general, it can be underlined that since it has an efficient implementation, it requires the distributed computation of a small subset of distances (step 1 in the algorithm). In the medical context these patterns cannot be exchanged over the network for privacy reasons, so, in this section, two suitable protocols for privacy-preserving similarity computation have been introduced. More formally, given two training patterns $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$, belonging to different agents, we want to compute $\mathbf{x_i}^T\mathbf{x_j}$, without exchanging the original data. In the following, there will be described the two techniques used for this purpose.

Random projection-based technique

The first introduced technique is the random projection-based, developed in [80]. Supposing that both agents agree on a projection matrix $\mathbf{R} \in \mathbb{R}^{m\times d}$, with $m < d$, such that each entry R_{ij} is independent and chosen from a normal distribution with zero mean and σ^2 variance, then the following definition can be given:

Theorem 8.1. *Given two input patterns $\mathbf{x}_i, \mathbf{x}_j$, and the respective projections:*

$$\mathbf{u}_i = \frac{1}{\sqrt{m}\sigma}\mathbf{R}\mathbf{x}_i, \text{ and } \mathbf{u}_j = \frac{1}{\sqrt{m}\sigma}\mathbf{R}\mathbf{x}_j\,, \tag{8.7}$$

we have that:

$$\mathbb{E}\left\{\mathbf{u}_i^T\mathbf{u}_j\right\} = \mathbf{x}_i^T\mathbf{x}_j\,. \tag{8.8}$$

Proof. Let r_{ij} and ϵ_{ij} be the i, j-th entries of a matrix R and $R^T R$ respectively.

$$\begin{aligned} \epsilon_{ij} &= \sum_{t=1}^{p} r_{t,i} r_{t,j} \\ \mathbb{E}\{\epsilon_{ij}\} &= \mathbb{E}\{\sum_{t=1}^{p} r_{t,i} r_{t,j}\} = \sum_{t=1}^{p} \mathbb{E}\ \{r_{t,i} r_{t,j}\} \end{aligned} \tag{8.9}$$

Since the entries of a random matrix are independent and identically distributed (i.i.d.):

$$\mathbb{E}\{\epsilon_{ij}\} = \begin{cases} \sum_{t=1}^{p} \mathbb{E}\ \{r_{t,i}\} \mathbb{E}\ \{r_{t,j}\} & \text{if } i \neq j \\ \sum_{t=1}^{p} \mathbb{E}\{r_{t,i}^2\} & \text{if } i = j \end{cases} \tag{8.10}$$

Noting that $\mathbb{E}\ \{r_{i,j}\} = 0$ and $\mathbb{E}\ \{r_{i,j}^2\} = \sigma_r^2$:

$$\mathbb{E}\{\epsilon_{ij}\} = \begin{cases} 0 & \text{if } i \neq j \\ p\sigma_r^2 & \text{if } i = j \end{cases} \tag{8.11}$$

Hence, $\mathbb{E}\ \{R^T R\} = p\sigma_r^2 I$. Similarly, we have $\mathbb{E}\ \{R^T R\} = q\sigma_r^2 I$ □

In light of Lemma 8.1, exchanging the projected patterns instead of the original ones allows to preserve, on average, their inner product. A thorough investigation on the privacy-preservation guarantees of this protocol can be found in [80]. Additionally, it can be observed that this protocol provides a reduction in the communication requirements since it effectively reduces the dimensionality of the patterns to be exchanged by a factor m/d.

k-anonymity technique

The second introduced technique is the k-anonymity realized in [132]. In this case, the pattern $\mathbf{x}_i$ is assumed to be composed by the following fields:

- Identifiers: are all of those values which are guaranteed to be unique for each object among patterns (e.g. Name).
- Quasi-identifiers: are all of those attributes that are not themselves unique identifiers, but that are sufficiently well correlated with the object of the patterns, that they could be combined with other information to become personally identifying fields (e.g. age);
- Sensitive attributes: are all of those fields that can not be transmitted from one site to another one since are strictly related to the object of the pattern (e.g. diagnosis);

In this case, it is assumed that the pattern $\mathbf{x}_i$ is composed by both quasi-identifiers fields and sensible fields.

Definition 8.2. We say that a dataset is k-anonymous if, for any pattern, there exist at least $k - 1$ other patterns with the same quasi-identifiers.

Dataset	Features	Instances	Classes
Pima Indians Diabetes	8	769	2
Breast Cancer Wisconsin	32	569	2
Parkinson Speech	26	1040	6

Table 8.10. Detailed description of each dataset.

In other words, the information of each person contained in a dataset cannot be distinguished from at least $k-1$ individuals whose information also appears in the dataset. Let suppose to have a table with n rows and m columns, each row represents a record, while each column an attribute associated with the member of the population. There are two methods to preserve k-anonymity:

- The "suppression", is the simplest way to proceed, where the values to be preserved are replaced by an asterisk '*';
- The "generalization" on the dataset, wherein the quasi-identifiers are binned in a set of Q predefined bins, and only the information on the corresponding bins is included in the dataset. Different values for Q correspond to different privacy values for k, with an inverse relation. In this case, the values to be protected are replaced by with a broader category.

In the next section, an analysis on the influences of these approaches on the distributed spectral clustering is given. For this reason, the generalization is performed artificially on the full dataset, while a fully decentralized implementation would require a more sophisticated procedure going outside the scope of this work.

Although these two protocols have been used due to their wide diffusion and simplicity, it can be stressed that the proposed algorithm does not depend specifically on any of them.

As follows is presented an algorithm 7, which extends t3, where in the first step it will be applied a technique of privacy-preserving.

8.4.2 Experimental validation

In this section, the performance of the proposed algorithm are evaluated for the decentralized spectral clustering with the privacy-preserving protocols described in Section 8.4.1. Three different (medical) public datasets available on the UCI repository [1] are considered and, a schematic description of them is given in Table 8.10. The number of attributes is always greater than three and depends on the specific features of the dataset. In all cases, for clustering, the optimal solution is known beforehand for testing purpose. Below some additional information on each dataset is given.

- *Pima Indians Diabetes Dataset* [133]: the task is to identify whenever the tests are positive for diabetes or negative.

[1] `https://archive.ics.uci.edu/ml/datasets.html`

Algorithm 7 Pseudocode of the proposed distributed spectral clustering algorithm (with privacy constraints), at the kth agent.

```
1: Input : Local dataset $S_k$, number of nodes $L$ (global), maximum number of
   iterations $T$.
2: for $n = 1$ to $n^{(1)}_{\max}$ do
3: The input data are protected through one of the following privacy-preserving
   strategies:
     • $K-$anonymity
     • Random projection-based
4:     Select a set of input patterns and share them with the neighbors $\mathcal{N}_k$.
5:     Receive patterns from the neighbors.
6: end for
7: Compute the incomplete EDM matrix $\hat{\mathbf{E}}_k$ $n = 1$ to $n^{(2)}_{\max}$
8: for  $n = 1$ to $n^{(2)}_{\max}$ do
9:     Select a set of entries from $\hat{\mathbf{E}}_k$ and share them with the neighbors.
10:    Receive entries from the neighbors.
11:    Update $\hat{\mathbf{E}}_k$ with the entries received.
12: end for
13: Initialize $\mathbf{V}_k[0]$.
14: for  $n = 1$ to $T$ do
15:    Compute $\mathbf{V}_k[n]$ using Eq. (5.12).
16:    Diffuse local information using Eq. (5.14). end for
17: Compute the Laplacian matrix $\tilde{\mathbf{L}}$ from $\tilde{\mathbf{E}}$.
18: Perform spectral clustering using $\tilde{\mathbf{L}}$.
```

- *Breast Cancer Wisconsin Dataset* [134]: it is a binary classification dataset, where the features describe the characteristics of the cell nuclei present in an image. The task is to identify the correct diagnosis (M = malignant, B = benign).

- *Parkinson Speech Dataset*: [135] the dataset contains data of 20 Parkinson's Disease patients (PD) and 20 healthy subjects for which multiple types of sound recording are taken. The aim is to identify the correct type of recorded sound.

Five different runs of simulation are performed for each dataset which is preventively normalized between -1 and 1 before the experiments and randomly partitioned among the agents. A network of 7 agents is considered, where every pair of nodes is connected with a fixed probability $p = 0.5$ according to "Erdos-Rènyi model". The only requirement is that the graph is connected. The following strategies are compared:

Dataset	Algorithm	F-Measure	Rand-Index	F-M Index
Pima Indians Diabetes	Centralized	0.511 ± 0.000	**0.542 ± 0.005**	0.721 ± 0.014
	No-privacy	**0.682 ± 0.106**	0.505 ± 0.000	0.711 ± 0.000
	Randomization	0.679 ± 0.050	0.523 ± 0.004	**0.723 ± 0.003**
Breast Cancer Wisconsin	Centralized	**0.785 ± 0.000**	0.543 ± 0.004	0.728 ± 0.004
	No-privacy	0.772 ± 0.330	0.624 ± 0.169	0.779 ± 0.086
	Randomization	0.609 ± 0.389	**0.682 ± 0.106**	**0.815 ± 0.041**
Parkinson Speech	Centralized	0.665 ± 0.001	0.450 ± 0.000	0.705 ± 0.000
	No-privacy	**0.674 ± 0.0.047**	**0.504 ± 0.000**	**0.710 ± 0.000**
	Randomization	0.672 ± 0.027	0.501 ± 0.003	0.708 ± 0.002

Table 8.11. Experimental results for the randomization. It can be shown that the average and the standard deviation of the indexes. Best results for each algorithm are highlighted in bold.

- *Centralized:* this simulates the case where a dataset is collected beforehand on a centralized location (for comparison).
- *No-privacy:* the dataset is used without any privacy protocol applied to the data;
- *Randomization protocol:* the privacy of the data in step 1 is preserved by computing the distance on the projected patterns according to (8.7); parameter d is chosen in $k = [2, \ldots, 8]$ to minimize RMSE;
- *K-anonymity:* the privacy of the data is preserved by generalization on the quasi-identifiers of the dataset. 4 bins are used for each quasi-identifier.

Dataset	Algorithm	F-Measure	Rand-Index	F-M Index
Pima Indians Diabetes	No-privacy	0.682 ± 0.106	0.505 ± 0.000	0.711 ± 0.000
	Randomization	0.679 ± 0.050	0.523 ± 0.004	0.723 ± 0.003
	K-anonymity	**0.779 ± 0.167**	**0.561 ± 0.031**	**0.749 ± 0.021**

Table 8.12. Experimental results for the k-anonymity. It can be shown the average and the standard deviation of the F-Index, Rand Index, FM-Index for both the Randomization, k-anonymity and free-privacy protocol. Best results for each algorithm are highlighted in bold.

The results of the framework are firstly evaluated by the randomization procedure. Three quality indexes are computed for both the privacy-preserving protocols and the privacy-free algorithm, namely the Rand Index, the Falks-Mallows index and the F-measure. In Table 8.11 are reported the mean and the standard deviation of each quality index averaged over 10 k-means evaluations and over the different agents in the distributed case. The best result for each index is highlighted in bold. The results of the three approaches are reasonably aligned. As it can be noted from the table, the results for all of the dataset are very similar and in some cases, the algorithm with privacy-preservation outperforms the traditional one. For evaluating the k-anonymity, the Pima Indians Diabetes Dataset, described in Section 8.4.2, is used. The first and the eighth feature, which are the number of times pregnant and the age of the subject respectively, are used as quasi-identifiers. In Table 8.12 are reported the three quality indexes for the k-anonymity protocol, the randomization, and the no-privacy transformation strategy. As it can be seen from Table 8.12 the performance are comparable with respect to the privacy-free algorithm, precisely in the k-anonymity protocol the results are even better.

8.4.3 Observation

In conclusion, two techniques able to extract useful information from medical data have been presented. Results are very good both in terms of classification accuracy and misclassification error. However, in a medical scenario, they could not be directly applied. In effect, one of the main requirements in this context is that data could not be transferred from one site to another for privacy concerns over sensible portion of the dataset. To this end, a general framework for performing distributed data mining procedure has been realized, with a specific emphasis on privacy-preserving protocols. Preliminary results on a clustering application show the feasibility of the approach, which is able to reach almost-optimal performance with respect to a fully centralized implementation.

All of the previously cited works, are only the first steps that could be realized for a possible telemedicine context. In effect, in addition to the possibility of achieving a protocol for analyzing and managing distributed data, a deep study on recent low-cost technologies able to acquire those data (from the home's patients) is necessary. In the next section, I will be present my contribution to this issue.

Chapter 9

Low-cost devices for tele-rehabilitation

The previous chapter introduces a genetic optimization and an exhaustive search for detecting the presence of an abnormality of the gait even when data is acquired from several sensors. In effect, analyzing data spread across multiples sources of information is the typical behavior of a tele-rehabilitation context. Since the aim of this part of the thesis is the realization of a complete tool to be used in the home of a patient where there is no or reduced medical supervision, in this section the attention is shifted on the capability of low-cost devices in capturing all of the spatio-temporal parameters used before now. Machine learning techniques are used to analyze and extract data from two low power devices (an RGB-D camera and IMU sensors), whose potentiality in acquiring the gait of a patient are also assessed.

In recent years, a lot of motion sensors started to be used to this end [136, 137], like inertial sensors, gaming device like Nintendo [138, 139] and Kinect Device [140, 141]. In [142] is summarized a comprehensive review of them. Besides gaming purposes, several works prove the potentiality of the device in applications outside gaming [143], showing its ability in assess stride dynamics during walk [144], evaluate the postural control [145], capture the upper extremity movements [146], or perform gait recognition from the frontal view [147]. All of these devices are promising technologies for the medical context, especially in the perspective of an outpatient medical facility. However, a careful analysis on theirs ability to effectively capture all the spatio-temporal gait parameters, which are the ones used by a movement laboratory, is still lacking.

Thus, in this work, the Microsoft Kinect device is used to monitor and quantify the movement of a patient, his clinical status, the degree of rehabilitation and the success of a therapy made ad-hoc for him. It allows the tracking of the movements of a person avoiding need of markers, controllers or force platforms. To have a complete representation of the movement, it will be flanked by the Inertial Measurement Unit (IMU), like accelerometers and gyroscope. They could not be used alone since they can be affected by noise and signal drift. Additionally, they need to be exactly positioned and, since each sensor is able to capture very few gait properties, several, redundant sensors are necessary to achieve an accurate analysis. On the other hand, they are very cheap and do not require complex software systems, thus their use, in combination with camera's data, seems to calibrate well the system.

9.1 Microsoft Kinect Device

9.1.1 Description of the device

The Microsoft Kinect Device is the first technology, proposed in this work, to record patient's motion before and after the training session. It allows a three-dimensional reality perception, similar to that obtained by the humans. It is composed by sensor components able to analyze the scene, and perception ones able to analyze the reaction of the sensor in the scene. Owning an angular field of view of 57°horizontally and 43°vertically, it is composed by:

- a 640 x 480 RGB camera that makes possible the capturing of a color image;
- a depth sensor able to capture a depth image;
- a 3-axis accelerometer used to determine the current orientation of the Kinect;
- a multi-array microphone, used to find the location of the sound source and the direction of the audio wave.

The sensor has different potentiality, it can be used to perform gesture recognition, facial recognition as well as voice recognition. However, the most important innovation that it brought with it, is linked to the capability to inference body position. The device captures the three-dimensional motions of a person thanks to the realization of a depth map. By capturing color and depth images at 30 frames per second, it provides detailed information about the twenty joints of the user's body such as hands, elbows, knees, head and so on. This is obtained by a two-stage process, firstly a depth map is computed, then the extracted information is used to infer the body position.

The infrared laser emitter is used to create a constant pattern of speckles projected onto the scene to be reconstructed. Successively, depth information is obtained in a very simple way: the translation of the pattern features is determined by the difference between a speckle pattern, which is observed by using an infrared (IR) camera, and the reference data at a known depth. In other words, it is possible to recover the distance of the patterns, by measuring their distortion. The depth map is an image whose pixel's values represent the distance from the origin of the sensors. The reference pattern is obtained by capturing a plane at a known distance from the sensor and storing it in the internal memory.

When there is an object with a distance smaller or larger than the one of the reference plane, the speckle pattern is projected on it and its position will be shifted in the direction of the baseline between the laser projector and the perspective center of the infrared camera. The shifts are evaluated for all the speckles thanks to an image correlation procedure able to produce a disparity map. The last one, combined with the known depth of the memorized plane, is used to estimate the depth for each pixel through triangulation.

The sensor can be used in all room light conditions (complete darkness or fully lit room) it does not require the user to wear anything and it does not require calibration. Thus, its capability to track the movements of a person avoiding need of markers, sensors, controllers or force platforms makes it suited for those patients that cannot stand for a long time.

9.1.2 Experimental setup

Fifteen people are asked to participate in the analysis. Precisely, 6 males and 9 females with an average age of 27.57 (ranged from 21 to 52) were recruited. After the informed consent, the participants were examined by experienced doctors and the ones with severe cognitive, perceptual or communication problems or any other health conditions that could have been not suitable for the experiment have been carefully excluded during the selection process.

It was asked to participants to wear a minimum number of garments (a pair of slip) in order to allow the application of the reflective markers by means of a double-side hypo-allergenic adhesive. Successively, the participants perform at least five walks that are analyzed simultaneously with the Kinect Sensor and the stereophotogrammetric system (already introduced in sec 8.2.3).

For the Kinect the analysis begins when the subject is standing in front of the device in a range not visible from the camera. In that instant, the gait's video recording, the depth map and a three-dimensional vision of the movement is obtained by the Microsoft sensor. Simultaneously, a software realized ad-hoc for this purpose is activated to record the three-dimensional coordinates of the skeleton joints.

The captured features are all the spatio-temporal parameters described in Sec. 8.2.1. For all of the parameters, both the left and the right values are considered, with the exception of the cadence and the mean speed that have a unique value for both feet; thus a total of 16 features are processed in the analysis.

9.1.3 Kinect's calibration

A skeleton model was directly obtained from the official Microsoft Software Development Kit (SDK). It allows to recognize people and follows their actions when are standing or sitting, but facing the sensor. Precisely, it was used to acquire the skeleton joint positions in the three axes coordinates. Successively, a software specifically designed for the experiments, able to convert them into the spatio-temporal parameters described in Sec.8.2.1, is realized. The analytic properties of the vectors and angles during the walk are exploited. A brief description of the proceeding is given as follows for each parameter:

Step length: by a video analysis it is possible to retrieve the instants in which the step begins and ends. Let (x_1, y_1, z_1) as the starting instant's coordinate and (x_2, y_2, z_2) as the ending one, then the step length can be evaluated as:

$$\text{Step length} = \sqrt{(y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{9.1}$$

The coordinate along the x-axis is not inserted into the formula since it causes a not realistic displacement into the horizontal plane that falsifies the real value of the parameter.

Stride length: similarly to the step length, it can be evaluated by considering the instants in which the foot touches the ground for two consecutive steps. Let (x_1, y_1, z_1) as initial contact foot's coordinate and (x_2, y_2, z_2) as the ending one, then the stride length can be evaluated as follows:

$$\text{Stride length} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{9.2}$$

In this case, the coordinate along the x-axis is also considered since the displacement in the horizontal plane is not so relevant for a linear gait.

Base width: is evaluated by considering the distance from one foot and the mean axes between the feet. Let (x_1, y_1, z_1) as the first step's coordinate and (x_2, y_2, z_2) as the second one, then the base width can be evaluated as:

$$\text{Base width} = \frac{\sqrt{(x_2 - x_1)^2}}{2} \tag{9.3}$$

Swing Speed considering (x_1, y_1, z_1) as the beginning starts of the swing phase and (x_2, y_2, z_2) as the ending one, then the swing speed can be evaluated as the ratio between the total covered distance during the swing phase and the total employed time to executed it. The space is evaluated thanks to the coordinates, while the time by considering the timestamp of the device.

Mean Speed: considering (x_1, y_1, z_1) as the starting point of the gait, and (x_2, y_2, z_2) as the ending one, then the mean speed can be evaluated as the ratio between the total covered distance during the gait cycle and the time employed to executed it. The space is evaluated thanks to the coordinates, while the time by considering the timestamps of the device.

Stance Phase: since the video of the movement and the timestamps of the device are known, this parameter can be evaluated in a very simple way. It is obtained by subtracting the instant in which the foot leaves the ground and the one in which the heel is in contact with it. It is averaged over all of the stance phases performed during the analysis.

Swing Phase: similarly to the previous parameter it is obtained by only changing the initial and the final instant of the phase. In this case, the parameter is evaluated as the time difference between the heel contact and the toe-off, averaged over all of the swing phases executed during the analysis.

Stride Phase: it is obtained by summing the swing and the stride phase, or by performing the time difference between two successive heel contacts of the same foot, averaged over all of the stride phases executed during the analysis.

Double support: is the percentage of gait when both feet are in contact with the floor. It is evaluated as the difference between the toe off of the controlateral foot and the heel contact of the foot one.

Cadence: it is evaluated by counting the number of steps executed between the initial and the end instants of a subjects' gait (the video analysis help us in obtaining this value). The parameter is evaluated as follows:

$$\text{Cadence} = 60\frac{\text{N. Step}}{\text{Total length[s]}} \tag{9.4}$$

Table 9.1 shows the parameters acquired by the Microsoft Kinect with the above procedure. Specifically, it is compared with the stereophotogrammetric system when a typical behavioral walking cycle is analyzed by both systems. It can be noticed that there are no significant differences among the gait parameters since the obtained values are very similar.

Once considered the kinematic analysis, the attention is shifted on the kinetics one. In the stereophotogrammetric system, a dynamometric platform is used to analyze the reaction forces of the movement in the ground. The same possibility is not allowed for the Microsoft Kinect device, thus the properties of angles, vectors and inner products are exploited to reconstruct the kinetics parameters during the walk. Since the Kinect is not able to well recognize the fine movements of the feet, in this work are considered only the knee's and the hip's angles during the gait. Below is given the procedure to determine them:

- Knee angle: since only the movement joints' coordinates are available from the Kinect device, this parameter is evaluated by exploiting the inner product of the angle to be reconstructed. To this end, the origin of an imaginary Cartesian plane is put in the knee's coordinate. Then, the hip and ankle coordinates are considered as the extremes of two vectors connected with the origin. In this way, the angle's behavior for each time stamp is evaluated thanks to the inner product between these two vectors.
- hip angle: similarly to the previous parameter, the hip's behavior is obtained by considering the inner product between the vector that joins the knee and the hip coordinates (which is the origin of an imaginary Cartesian plane) and the one that joins the hip with the shoulder coordinate.

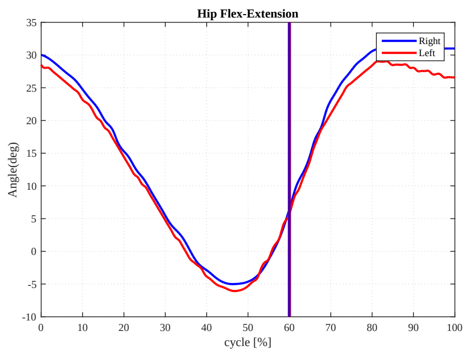
Representative trajectories of hip and knee sagittal range of motion throughout the whole gait cycle have been represented in Fig. 9.1a and Fig. 9.2a for the stereophotogrammetric system and in Fig. 9.1b and Fig. 9.2b for the Kinect device, respectively. The trend is reported for both left and right side with a blue and red line, respectively. The Kinect is very accurate in the reconstruction of movements, with a peak that occurs immediately before the beginning of the swing phase (indicated by the purple vertical line) and at the end of the stance phase. In the knee angle, the flexion peak is reached among the 60-80% of the gait cycle, with a value of 50° that is correctly checked by the proposed system.

9.1.4 Statistical analysis

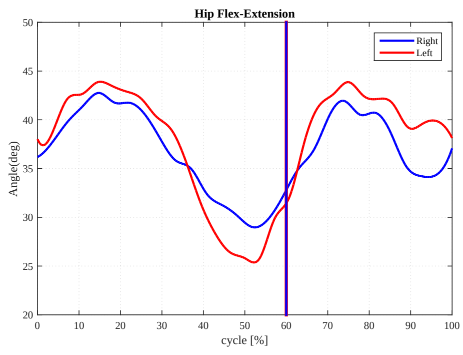
To prove the reliability of the device several statistical tests have been carried on to assess the possible discrepancies among the stereophotogrammetric system and the

	Microsoft Kinect Device		Stereophotogrammetric System	
Feature	$\mu_L \pm \sigma_L$	$\mu_R \pm \sigma_R$	$\mu_L \pm \sigma_L$	$\mu_R \pm \sigma_R$
Step length (m)	0.74 ± 0.02	0.71 ± 0.00	0.76 ± 0.06	0.66 ± 0.04
Stride length (m)	1.44 ± 0.04	1.44 ± 0.00	1.41 ± 0.08	1.34 ± 0.15
Base width (m)	0.05 ± 0.00	0.06 ± 0.00	0.10 ± 0.01	0.08 ± 0.02
Swing speed (m/s)	2.63 ± 0.49	2.73 ± 0.00	2.89 ± 0.11	2.99 ± 0.17
Stance phase (%)	55.70 ± 1.00	57.10 ± 0.30	55.10 ± 2.70	57.20 ± 3.00
Stance phase (s)	0.59 ± 0.03	0.60 ± 1.79	0.60 ± 0.04	0.60 ± 0.04
Swing phase (%)	44.60 ± 1.00	42.90 ± 0.30	44.90 ± 2.70	42.80 ± 3.00
Swing phase (s)	0.47 ± 0.03	0.45 ± 0.26	0.49 ± 0.02	0.45 ± 0.06
Double sup. phase (%)	7.33 ± 0.39	6.29 ± 0.39	7.30 ± 0.00	6.20 ± 0.00
Stride phase (s)	1.07 ± 0.06	1.06 ± 0.18	1.09 ± 0.02	1.05 ± 0.06
Mean speed (m/s)	1.30 ± 0.00		1.29 ± 0.00	
Cadence (step/m)	111.55 ± 5.89		112.45 ± 3.45	

Table 9.1. Comparison between the optoelectronic system and the Microsoft Kinect Device. The mean and standard deviation for both left (L) and right (R) measured values of every feature are reported.



(a) Stereophotogrammetric system

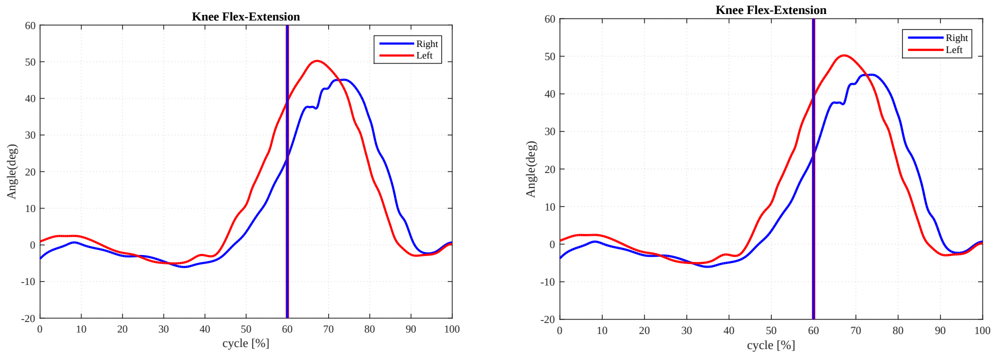


(b) Kinect Device

Figure 9.1. An example of the hip motion during the gait cycle. Blue line is for left hip, red line is for right hip. The purple line represents the end of the stance phase and the beginning of the swing phase for both the stereophotogrammetric system (a) and the Kinect Device (b).

Microsoft Kinect device. Precisely, the results are validated thanks to the following statistical coefficients:

- Pearson coefficient correlation;



(a) Stereophotogrammetric system **(b)** Kinect Device

Figure 9.2. An example of the knee motion during the gait cycle. Blue line is for left knee, red line is for right knee. The purple line represents the end of the stance phase and the beginning of the swing phase for both the stereophotogrammetric system (a) and the Kinect Device (b).

- Bland and Altman Plot;
- Limit of Agreement (LOA);
- Intra-class correlations coefficients.

A further analysis of them is given in Appendix B. The results are reported in Table 9.2 where is shown a great agreement between the two systems with a Pearson coefficient of above 0.9 in most of the cases. The temporal parameters obtain the better results, while the spatial parameters are slightly less good because there is a feature that undergo a great error. This is the base width. Since the Kinect is not able to acquire the fine movements of the feet, a variation of 3 or 4 centimeters on a value of few centimeters' order causes the important underlined error. Therefore, the Kinect is able to reasonably measure the most clinically relevant spatio-temporal parameters. Additionally, in Fig. 9.3 are showed the Bland-Altman plots for two of the captured parameters (Stride phase and Double support L time) where points are uniformly and tightly scattered around the horizontal axis.

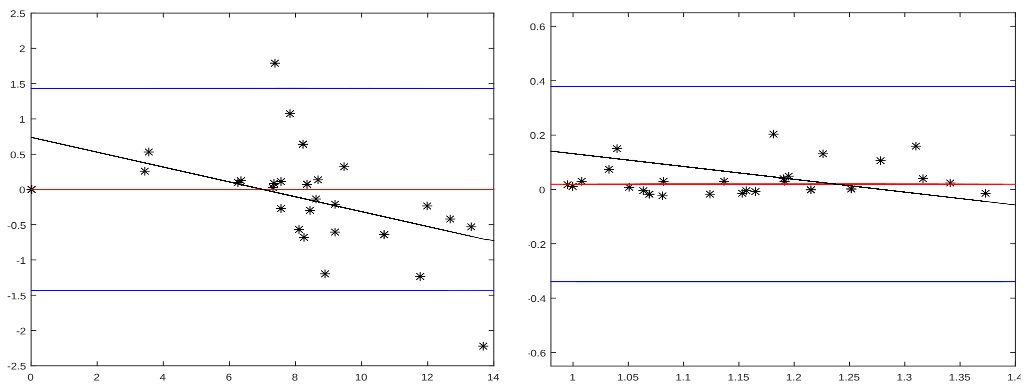
9.2 A smartphone device

Once ascertained the capability of the Microsoft Kinect device in acquiring the most important gait parameters, it will be combined with data obtained by IMU sensors to obtain a complete representation of the gait of a subject. They are a possible alternative for monitoring motion in an economical way. However, these sensors have shown to have possible drift problems caused by the accumulation of measurement errors. For this reason, they must be used in parallel with other devices. In particular, the combination of accelerometer, gyroscope and camera's data seems to calibrate well the system, making it more accurate.

Since the IMU sensors could be difficult to retrieve in a home context, this work will use the sensors contained in a smartphone. Its advantages of being

Feature	Pearson	ICC	R^2	LoA
Step length (m)	0.836	0.905	0.699	[0.14 -0.13]
Stride length (m)	0.965	0.977	0.931	[0.14 -0.07]
Base width	0.517	0.668	0.267	[0.07 -0.07]
Swing speed (m/s)	0.560	0.614	0.313	[0.67 -1.19]
Stance phase (%)	0.938	0.966	0.880	[1.63 -1.51]
Swing phase (%)	0.939	0.966	0.881	[1.51 -1.63]
Double support (%)	0.969	0.977	0.939	[1.42 -1.53]
Stride phase (%)	0.888	0.923	0.788	[0.09 -0.08]
Mean speed	0.971	0.983	0.943	[0.11 -0.08]
Cadence (step/m)	0.981	0.984	0.963	[3.47 -5.05]

Table 9.2. Numerical results of statistical tests.



(a) Double Support L **(b)** Stride Phase (s)

Figure 9.3. Bland-Altman plot with limits of agreement. The difference between the two devices is plotted on the Y-axis, and the mean score on the X-axis.

compact, cost-effective and relatively easy to operate when is compared to other expensive technologies, make it particularly promising for this context. Additionally, the filtered and captured data will be compared with data fusion and pattern recognition techniques able to support clinicians by extracting the important features for performing a diagnosis.

9.2.1 Protocol setup

Similarly to what done for the Microsoft Kinect, in this section it will be performed an analysis on the data captured from the smartphone. To this purpose, the flow-chart 9.4 summarizes the scheme for gait monitoring, while a detailed description of each phase is given as follows:

Collected Raw Data From Sensors: the data acquisition is performed by the following

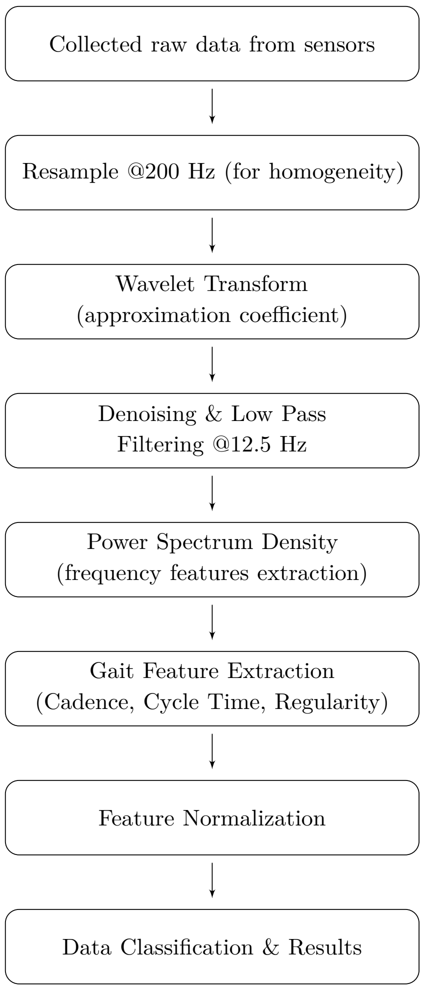

Collected raw data from sensors
Resample @200 Hz (for homogeneity)
Wavelet Transform
(approximation coefficient)
Denoising & Low Pass
Filtering @12.5 Hz
Power Spectrum Density
(frequency features extraction)
Gait Feature Extraction
(Cadence, Cycle Time, Regularity)
Feature Normalization
Data Classification & Results

Figure 9.4. Script's Flowchart

steps:

- The user opens the application of the Android device and sets the recording time;
- The smartphone is put into the band's pocket and fasten around the calf;
- The user presses the "Start" command on the screen and, after a countdown of 3 seconds, the app begins to record;
- During the recording time the user performs the walking test; a vibration of the device will advise him/her that the time expired;
- At the end of the trial, the user has to upload data to the database answering "Yes" to the upcoming question on the screen.

If the user is a voluntary hemiplegic patient, an assistant helps him/her in any demands. Accelerometer, gyroscope, and magnetometer data are captured during each trial. Firstly, the magnetometer ones are discarded because they are too sensible to metal objects eventually present in the environment. In effect, since no video reports were made by Medical Center, magnetometer data disturbs are unpredictable.

Resampling of data: once captured data from sensors, the second step consists in its resampling to achieve a homogeneous set of measures even when different physical devices are used for the acquisition. In effect, the app's code does not establish sampling frequency but it depends on the sensor's technology available on the single device. A sampling rate of 200 Hz is used, evaluated as an average weighted of its lowest (around 50 Hz) and highest (around 350-380 Hz) value.

Wavelet Transform and filtering: the successive step consists in evaluating the wavelet transform firstly to the de-noised acceleration and gyroscope data, and successively to the filtered ones. The sampling rate's choice allows us to set the wavelet coefficients to values that better fit the target of de-noising and low pass filtering. A wavelet de-noise algorithm (*'wden'*) is firstly applied [148], and modeled in the following way:

$$s(n) = f(n) + \sigma e(n) \tag{9.5}$$

whereas n is the time of sample, s is the noisy signal, f is the neat signal, and *e(n)* is a Gaussian white noise *N(0,1)* with σ equal to 1. This algorithm is able to suppress the noise part of the signal s and recover f by the following three steps:

1. Decomposition: a wavelet decomposition of the signal is evaluated by choosing a level W and a type of wavelet.
2. Detail coefficients thresholding: the algorithm applies soft thresholding for each level from 1 to W in order to detail coefficients.
3. Reconstruction: the algorithm computes wavelet reconstruction on the basis of the original approximation coefficients of level W and of the modified detail coefficients of levels from 1 to W.

In this work, the level coefficient W is set equal to 4 in order to scale the signal through Mallat's filter banks down-sampling algorithm at 12.5 Hz. Let us suppose that the average walking frequency of healthy people is about 1.8 Hz [149], then the frequency 12.5 Hz is suitable for a noise reduction without wasting possible pieces of pattern information. Subsequently, both reconstructed and de-noised acceleration and gyroscope 3-axis signals are passed through a level 4 wavelet filter bank. Low pass filters and down-sampling to 12.5 Hz make signals easier to be understood (Figure 9.5). In this way, once obtained the de-noised and filtered signal, the successive step of features extraction will be easier performed.

Power Spectrum Density: the translation of de-noised (but not filtered) acceleration and gyro absolute values is performed through the "periodogram" Power Spectrum Density (PSD). As literature suggests [150], the maximum PSD magnitude value is used as a feature for the classification purpose, for both the acceleration and gyroscope data (Figure 9.7). In particular, it should be noted that in most cases the acceleration's average frequency is located on gyro's second harmonic. However,

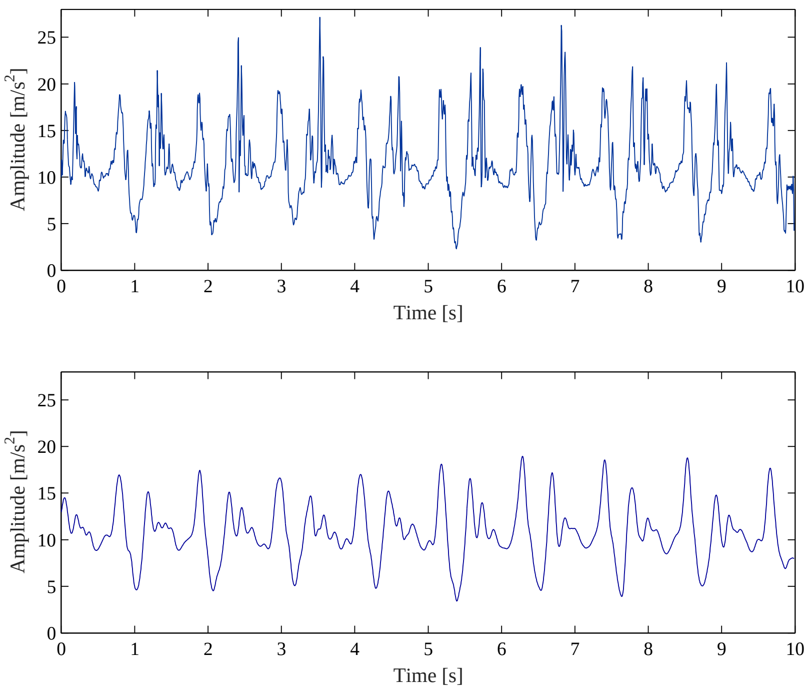


Figure 9.5. Effects of wavelet transform on acceleration signal. In the top figure is reported the raw acceleration, while in the bottom the denoised and filtered one.

because of important fluctuations, this rules could not be always satisfied.

Gait Feature Extraction: as said before, the first features used for the classification problem are the PSD magnitude values. In Fig.9.7 are reported the PSD extraction and the frequency values. The last ones, are then used, with a specific algorithm, to perform the gait stride recognition represented in Fig.9.6. Each stride is recognizable from the acceleration pattern as the time between two “valleys”: in fact, when the foot hits the ground, the sudden acceleration causes a spike followed by a deceleration that is represented by a valley, then the successive leg swinging causes a new acceleration and the process is repeated cyclically. However, this behavior is more evident in healthy people rather than hemiplegic ones. In order to find out the foot contact’ instants, both cycle time durations (from accelerometer and gyroscope data) are considered: the intersection of these two arrays is considered for cycle recognition and discrimination (Figure 9.6). In addition to the PSD magnitude values, other three features are used for gait cycles discrimination: cycle duration, cycle regularity, and cadence. Assuming that the first valley is the starting point of the stride and that the last one is performed during the time gap between two consecutive valleys, the Cycle Duration (C_d) can be defined as:

$$C_d[n] = V[n+1] - V[n] \tag{9.6}$$

whereas V is the array containing valleys’ locations, and n is the time index of them.

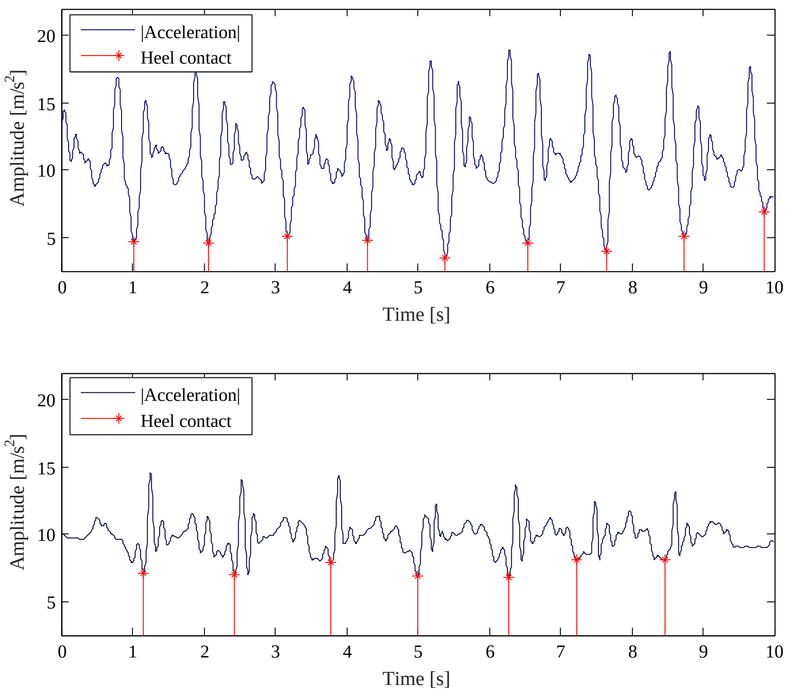


Figure 9.6. Stride recognition. In the top panel is reported the cycle recognition for an healthy subject, while in the bottom one a non-healthy subject behavior is described.

Every value is expressed in seconds. It represents the difference between the valley of a gait cycle.
The Cycle Regularity (C_r) is expressed as the standard deviation of the Cycle Duration, measured in second:

$$C_r = \sigma(C_d) \tag{9.7}$$

It reaches greater regularity when the value becomes close to zero. Anyway, 0 is reached only with one cycle detected.
Finally, the Cadence or "revolutions per minute" (RPM) R_p can be defined as:

$$R_p = \frac{60N_c}{V_l - V_f} \tag{9.8}$$

whereas N_c is the number of cycles taken from the length of cycle duration array, V_l is the last element of the valley location and V_f is the first one. Cadence is the projection of how many strides could be performed in a minute. It is expressed in cycles/min.

All the features extracted are summarized in Table 9.3:

Features Normalization: before running the classifying script it is helpful normalizing features for scaling results to the range between 0 and 1. Data normalization can be defined as follows:

Feature	Description
Mean Cycle Time $[s]$	Mean value of the cycle duration
Cycle Regularity $[s]$	Standard deviation of the stride length during a trial.
Cadence $[step/min]$	The number of step executed in a given time.
Max PSD Acc	Maximum value of the acceleration PSD
Max PSD Gyro	Maximum value of the PSD of the gyroscope

Table 9.3. A detailed description of the spatio-temporal parameters used for the analysis.

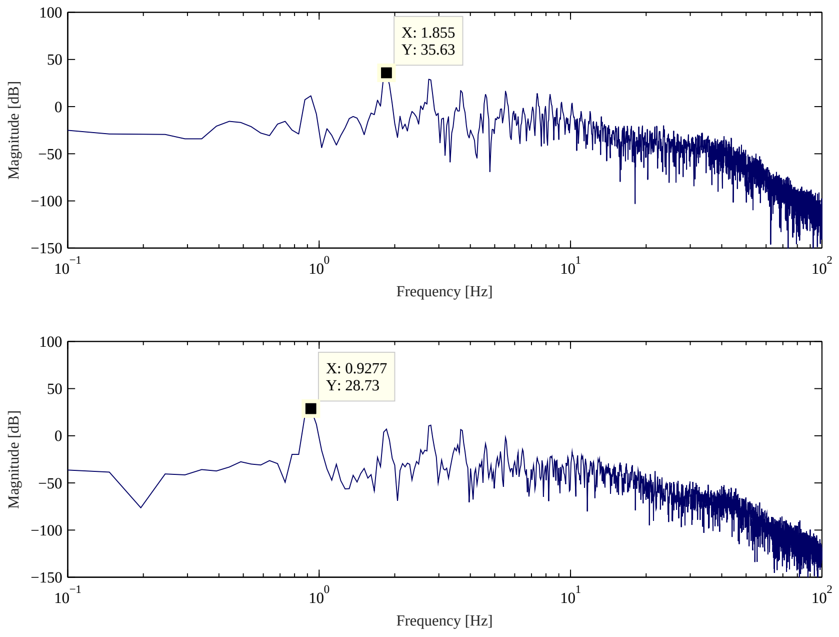


Figure 9.7. Maximum PSD extraction. In the top panel is reported the power spectrum density for the acceleration, while, in the bottom one, is reported the gyroscope behavior.

let M be the number of data files of the dataset $D = x_1, x_2, ..., x_M$ and N be the number of features; i.e., each file in the dataset is represented by a row of N numbers:

$$x_m = [x_{m_1}\, x_{m_2}\, \dots\, x_{m_N}], \qquad m = 1 \dots M. \tag{9.9}$$

Since data features are completely heterogeneous, patterns cannot be normalized globally, but through each column independently:

$$x_{m_j} \leftarrow \frac{x_{m_j} - b_j}{a_j - b_j}, \qquad m = 1 \dots M \,\text{and}\, j = 1 \dots N \tag{9.10}$$

whereas: $a_j = \max_{m=1\dots M}\left(x_{m_j}\right)$ and $b_j = \min_{m=1\dots M}\left(x_{m_j}\right)$.

Data Classification and results: the last step in flowchart consists of running different types of classifiers in order to categorize data of healthy people and patients. Anyway, this kind of passage is useful even to understand if the considered gait features are able to perform this kind of classification. All the possible combinations of input features are performed and saved in a different dataset. A 10 fold cross validation is used and several classification algorithms are compared in terms of misclassification error.

9.2.2 Experimental trials

The experiments are performed by putting the smartphone in the pocket of a band and ties around the user's calf as is shown in Fig. 9.8. The choice of the calf is not casual because most of the gait information and lower limb's angle movements involve only to this muscle. An only smartphone is used for the experiment in

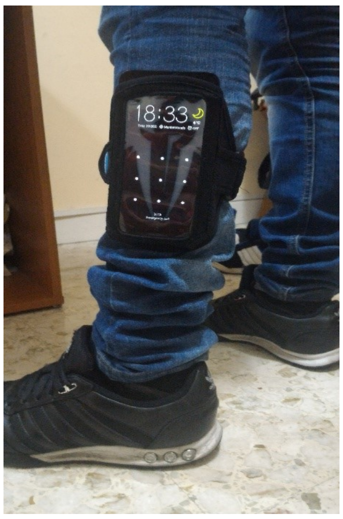


Figure 9.8. Device position

order to realize the most possible simple and cheap approach, without finding what happens when two or more devices are added to the analysis. However, in this way, it was not possible to deal with difficult scenarios, e.g. time synchronism and sensor mismatch between smartphones, missing all of the information on the "not-sensing" limb. Thus, it is not possible provide a full representation of the gait cycle presented in sec:8.2, since one sensor node is not able to achieve this. Rather, the aim of this work is to evaluate some additional features, that could be flanked to those obtained by the Microsoft Kinect device, in order to obtain a more accurate description of the movement for a possible home-based lower limb rehabilitation. In particular, features like the average stride length, the total walking stance or the number of steps are added to the previous ones acquired by the RGB, to make more affordable the gait analysis.

Both healthy people and post-stroke patients took part in the experiments. 60 different walking trials through heterogeneous devices and from different places

Classifier	Maximum error	Average error	Best subset	Best Cardinality
LDA	0.933	0.822 ± 0.128	00001	1
QDA	1.000	0.884 ± 0.135	10101	3
KNN	0.983	0.875 ± 0.125	10011	3
Naive Bayes	0.967	0.904 ± 0.898	01001	2
SVM	1.000	0.898 ± 0.138	00101	2
Neuro-Fuzzy	0.983	0.894 ± 0.112	10001	2
CART	0.950	0.877 ± 0.107	00001	1
PNN	1.000	0.911 ± 0.112	00101	2
FIS	0.967	0.864 ± 0.119	00001	1

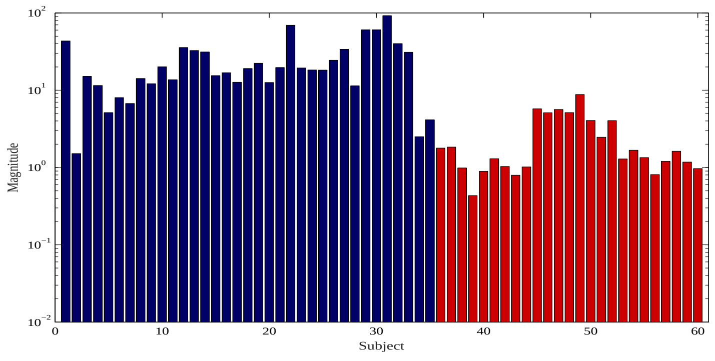
Table 9.4. Classification results in terms of minimum, maximum and average classification error. The last two columns report the best subset of features and its cardinality.

have been collected. Among these 60 records, 25 of them belong to voluntary patients from Rehabilitation Medical Center of the 2nd Hospital of Jiaxing, Zhejiang province, China; the remaining 35 are splitted between researchers of the university and doctors from the previous-cited medical center. Additionally, data differs for the length of the recording session: 41 of them (respectively 13 from patients and 28 from healthy people) are recorded in 10 seconds, the remaining 19 (12 of which are patient) are recorded in 20 seconds. People are asked to perform a walking in a straight path, without deviation.

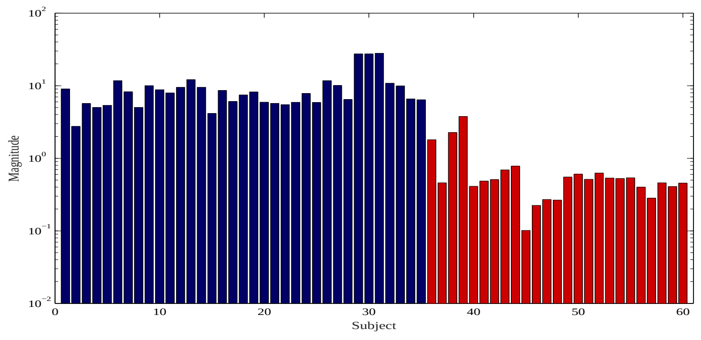
Figure 9.9 immediately explains how fitting maximum PSD magnitude is. Through sorting adjacently the two groups of files and showing magnitude differences, the reader could have a good estimation of classification results using the acceleration and gyroscope's PSD magnitude. Numerical results are reported in table 9.4, where the effectiveness of the device is tested by comparing several classification algorithms (detailed in Appendix C) in terms of misclassification error. The best feature set, after an exhaustive search over all of their possible combinations, is reported. It can be shown that, for each algorithm, the extracted features are able to well discriminate among a diseased patient and a control one.

9.2.3 Observation and conclusions

The beginning of this chapter carries on an analysis on the reliability and the validity of a novel method for full body gait analysis achieved through the Microsoft Kinect device. It is a low-cost portable device that provides us a low-cost portable system able to track the movements avoiding need of markers, sensors or controllers. This is an important aspect to underline, because the traditional systems often require a considerable time for marker application, as well as for the execution of the test that often must be repeated several times for marker detection, creating difficulties for those patients that cannot stand for a long time. Once asserted the accuracy of the Kinect, several statistical tests shown that results are comparable when assessing spatio-temporal parameters during the commonly used clinical tests. To this end,



(a) Acceleration PSD



(b) Gyroscope PSD

Figure 9.9. maximum PSD acceleration's differences between healthy and not healthy users

the traditional optoelectronic system, which is a gold standard in the gait analysis, is used for comparison. Successively, the gait monitoring has been extended by using the sensors contained in a smartphone. Respect to the state of the art applications, the data acquired and filtered from the device is compared with data fusion and pattern recognition techniques able to provide a correct definition of the gait's movements. The results show that the system is able to well acquire some interesting gait parameters, allowing also a good discrimination among diseased and healthy subjects.

In this way, these data could be combined with those captured from the Kinect Device producing a calibrated system to be used for a telerehabilitation project. In this context, the use of technology has become relevant in order to better understand, improve and develop cutting-edge treatment approaches, often allowing a decreasing of the procedures' costs.

Chapter 10

Neural Networks Implemented with Finite Precision Arithmetic

As discussed in Chapter 8, to infer knowledge from big data partitioned over geographically distinct locations is a fundamental problem in the context of tele-rehabilitation. However, this is not the only possible implementation, in fact, in this chapter, the focus is shifted on another field of distributed learning application, which is the one of sensors networks and low-power devices. In this case, traditional centralized techniques could not be applied for the presence of additional constraints on the hardware resources. For these reasons, a Random-Vector Functional Link is used to make possible the learning on simple architectures. Additionally, a genetic optimization will be adopted to determine the quantization level of both the input data and the interval parameters of the network.

10.1 Introduction

Internet of Things (IoT), cloud computing, pervasive computing and so on, have revolutionized the way by which signals are processed and the information is managed. New ways of processing, sensing, and computing have overcome the traditional centralized architectures posing new challenges and new opportunities for distributed networked signal processing. Thus, it is no longer possible to send data in a central node where traditional learning algorithms will extract the important information, while new techniques able to model and manage data are mandatory.

A large class of environments requires the deployment of distributed wireless sensors networks (WSN), e.g. environmental monitoring, habitat sensing, disaster management. They have significant benefits, for example, they can be used to remotely monitor inhospitable and toxic environments or to realize an intelligent patient monitoring as the one introduced in the previous chapter. Typically, a WSN is composed of a large number of low-cost, low-power and heterogeneous devices, called wireless sensors nodes. They are densely deployed in a region of interest in order to sense, process and communicate information from their surroundings. A WSN may be realized rapidly and without large requirements by involving the following items:

- environmental sensors;
- tools to communicate the information among neighbors and to the external users (wifi, Bluetooth...);
- techniques able to extract the information from raw data.

In effect their nature allows an optimum trade-off between precision and implementation (costs and performances): the technological advances allow to build small, inexpensive and battery-powered sensing devices, while the self-organized and infrastructure-less wireless sensor network let be possible the transmission of data packets thanks to the coordination between the sensor nodes. The performances of these devices are characterized by parameters such as power/weight ratio, strength, response rate, physical size, speed of motion, reliability cost and so on. To maximize these features, other aspects, including circuits, architectures, algorithms, and protocols, have to be energy efficient.

Generally, these sensors are used to make inference about the environments that they are sensing by monitoring physical conditions, such as temperature, sound, pressure and so on. However, separate domains of application may largely impose different constraints on the solution, including low computational power at every location, limited underlying connectivity (e.g. no broadcasting capability), or transferability constraints related to the enormous bandwidth requirement. The computational power may not be sufficient to analyze a too high quantity of information. Thus, the problem on which this chapter focuses the attention, is how to best map learning applications onto hardware, implementing low power protocols and overcoming the intrinsic energy constraints.

It is known that neural networks are traditionally used to provide solutions to many problems in the areas of pattern recognition, signal processing, and time series analysis. However, these approaches are computationally intensive and memory demanding, making difficult to deploy them on distributed systems with simple hardware. Usually, neural networks parameters are estimated via learning algorithms running on standard computers with double precision floating point arithmetic. However, uploading the network model on a digital architecture with finite precision arithmetic, after a direct quantization of coefficients, leads to unsatisfactory results due to the non linear nature of the network [151, 152]. Nonetheless, many real-time applications need an adaptive learning, as in the case of the well-known consensus strategy, where the model must be dynamically adapted to new observations even after hardware implementation. There have been various proposals to make inference in contexts with limited hardware resources. A learning procedure for generating multilayer integer-weight neural networks has been presented in [153]. Differential evolution strategies have been also applied to train neural networks with small integer constraints [154]. In [155], a training mechanism with quantized weights that reduces the cost of hardware implementation has been presented and in [156] both approximation and quantization techniques are used for network pruning. However, all of such models are not suited to deal with distributed learning and data processing.

In this work, the attention is given on the Random Vector Functional-Link (RVFL), which can be applied to signal processing applications where function approximation, classification or time series forecasting is required. In particular, this technique has been used to implement distributed learning systems where training data is diffused under a decentralized information structure [157], or to solve specific classification problem [158]. For instance, an RVFL can be implemented on each node of a network of interconnected agents, where the constraints imposed by the hardware implementation of the node (e.g. by using Arduino, Raspberry or other cheap electronic cards) make necessary to deal with a finite precision arithmetic.

So far, only preliminary studies have been proposed for RVFL networks with limited hardware resources; for instance, in [159] the algebraic properties of the mathematical model are investigated, considering the implementation on FPGA architectures with a relatively high number of bits (i.e., more than 16). In the following, a novel optimization strategy based on a Genetic Algorithm (GA) is introduced to estimate the inner parameters of the network under the constraint of finite precision arithmetic. It will be also described a nonuniform quantization of the input data feeding the network, in order to cope with the actual structure of datasets and also to compensate the rounding of internal parameters of the neural model. The problem will be around how to balance between the need for numeric precision, which is important for network accuracy, the speed of convergence, and the cost of more logic areas associated with increased precision.

10.2 RVFL architecture

In Section 6.1.2 is introduced a random vector functional link for a classification purpose, now it will be used for a regression one.

The RVFL can be viewed as a feed-forward neural network with a single hidden layer, resulting in a linear combination of a fixed number of non-linear expansions of the original input. The peculiarity of an RVFL feed-forward neural network is the inner layer fixed a priori with a predefined set of nodes. Given a d-dimensional input $\mathbf{x} = [x_1 \ldots x_d]^T$, the RVFL aims at estimating a scalar output $y \in \mathbb{R}$. The traditional formulation uses a weighted sum of C non-linear transformations of the input:

$$f(\mathbf{x}) = \sum_{m=1}^{C} \beta_m h_m(\mathbf{x}; \mathbf{w}_m)\,. \tag{10.1}$$

Each $h : \mathcal{X} \to \mathbb{R}$ is a *base, hidden* function, or *functional link.* In the following, a sigmoid basis functions is adopted:

$$h(\mathbf{x}; \mathbf{w}, b) = \frac{1}{1 + \exp\{-\mathbf{w}^T\mathbf{x} + b\}}\,. \tag{10.2}$$

The resulting model is linear in the β parameters and the parameters $\mathbf{w}_1, ..., \mathbf{w}_C$ are initially assigned randomly before the training process, according to a uniform probability distribution. A schematic depiction of an RVFL with two inputs, three hidden nodes, and one output is given in Fig. 10.1. Under few assumptions on the smoothness of the underlying function, the RVFL has universal approximation

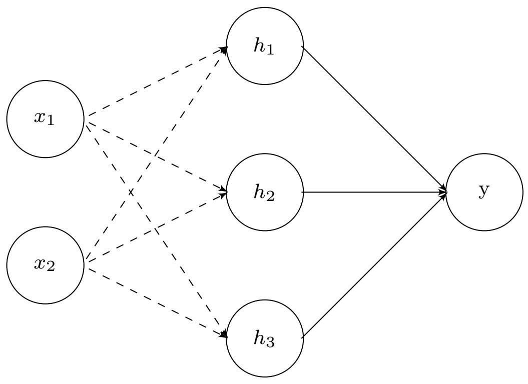


Figure 10.1. Schematic depiction of an RVFL architecture with two inputs, three hidden nodes, and one output. Fixed connections are shown as dashed lines, while trainable connections as fixed lines.

capability if a large number of hidden functions is provided, that is forcing C to be large enough [160].

The problem of learning an RVFL model of the form (10.1) should be reorganized as a linear regression model over the coefficients $\boldsymbol{\beta} = [\beta_1 \ldots \beta_C]^T$. Considering a training set of N input-output pairs $\{\mathbf{x}_i, y_i\}$, $i = 1 \ldots N$, a hidden matrix $\mathbf{H}$ can be organized in the following way:

$$\mathbf{H} = \begin{bmatrix} h_1(\mathbf{x}_1) & \cdots & h_C(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_N) & \cdots & h_C(\mathbf{x}_N) \end{bmatrix}, \tag{10.3}$$

while the output vector is $\mathbf{y} = [y_1 \ldots y_N]^T$.

Definition 10.1. The optimization procedure can be expressed as a standard regularized least-squares (RLS) problem where the optimal $\boldsymbol{\beta}$ for training an RVFL is found by minimizing the following objective function:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^C} \frac{1}{2} \|\mathbf{y} - \mathbf{H}\boldsymbol{\beta}\|_2^2 + \frac{\lambda}{2} \|\boldsymbol{\beta}\|_2^2 , \tag{10.4}$$

where $\lambda > 0$ is the *regularization factor*. The problem (10.4) is strictly convex, so the solution can be found by setting the gradient of $\mathcal{J}(\boldsymbol{\beta})$ equal to 0:

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\beta}} = \mathbf{H}^{\mathrm{T}}\mathbf{H}\boldsymbol{\beta} - \mathbf{H}^{\mathrm{T}}\mathbf{Y} + \lambda\boldsymbol{\beta} = 0, \tag{10.5}$$

from which the well-known solution is obtained:

$$\boldsymbol{\beta}^* = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\mathbf{I}\right)^{-1} \mathbf{H}^{\mathrm{T}}\mathbf{y}. \tag{10.6}$$

where $\mathbf{I}$ is the identity matrix of suitable dimensionality. The computational complexity of RVFL training is influenced by the CxC matrix inversion (10.6).

When $N \ll C$, the solution (10.6) can be simplified by consider that, for any $\lambda > 0$, we have that:

$$(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^{\mathrm{T}} = \mathbf{H}^{\mathrm{T}}(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\mathbf{I})^{-1}. \tag{10.7}$$

Combining equations (10.6) and (10.7) an alternative formulation for the optimal weight vector is obtained, where the matrix to be inverted is reduced to dimensionality NxN:

$$\boldsymbol{\beta}^* = \mathbf{H}^{\mathrm{T}}\left(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda\mathbf{I}\right)^{-1}\mathbf{y}\,. \tag{10.8}$$

However, in the following study it will not be performed any assumption on data thus, for the rest of the chapter, the optimal solution given in (10.6) will be considered. The implementation of RVFL networks on a hardware with finite precision arithmetic imposes that all the computed functions and the related parameters are quantized. How this can be obtained by the combination of specific digital circuits (i.e., adders, multipliers, lookup tables, etc.) is not in the purpose of the work. However, by a theoretical point of view, it is important to consider the quantization procedure on the underlying model.

10.3 Finite precision model of RVFL Networks

In this section, it is firstly tested what happens when all the introduced parameters, input and output values, and the computation of hidden functions are subject to a uniform quantization. Successively, a non-uniform distribution of the input interval is also investigated to underline how the results change. Namely, it is proposed a non linear A/D conversion of input signals by considering the actual structure of data to be processed. In both cases, the genetic optimization is used to tune their implementation on hardware architectures based on a finite precision arithmetic.

10.4 Uniform quantization

In a finite precision implementation of an RVFL network all the introduced parameters, input and output values, and the computation of hidden functions must be quantized. The first proposed approach is based on a uniform quantization.

Let $q_n(\cdot)$ be a uniform quantizer in the respective range of the input variables, with a two's complement binary representation using n bits. It is applied element-wise if the input is either a vector or a matrix. With respect to the previous processing, given the same generation of random of weights, a quantized hidden matrix $\mathbf{H}^{(n)}$ will be arranged on the basis of the following hidden functions:

$$h^{(n)}(\mathbf{x}; \mathbf{w}, b) = q_n\left(\frac{1}{1 + \exp\left\{-q_n(\mathbf{w}^T)q_n(\mathbf{x}) + q_n(b)\right\}}\right)\,. \tag{10.9}$$

Then, the generic output for an n-bit finite precision implementation of an RVFL will be:

$$f^{(n)}(\mathbf{x}) = q_n\left(\sum_{m=1}^{C} \beta_m^{(n)} h_m^{(n)}(\mathbf{x}; \mathbf{w}_m)\right)\,. \tag{10.10}$$

Since the weights are extracted from a uniform distribution, the parameters $\mathbf{w}_1, ..., \mathbf{w}_C$ can be considered as forced to be quantized, as well as the results of the hidden functions, in such a way that the intrinsic randomness of the RVFL input and inner layers is still preserved. However, the main problem arises in the computation of finite precision $\boldsymbol{\beta}^{(n)}$ parameters in (10.10), which can be reformulated as an Integer Least Squares (ILS) problem:

Definition 10.2. The Integer Least Squares Problem, could be defined as in the following:

$$\begin{aligned} \min_{\beta^{(n)} \in \mathbb{R}^C} \ & \frac{1}{2} \left\| \mathbf{y} - q_n\left(\mathbf{H}^{(n)} \boldsymbol{\beta}^{(n)}\right) \right\|_2^2 + \frac{\lambda}{2} \left\| \boldsymbol{\beta}^{(n)} \right\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{\beta}^{(n)} \in \mathbb{Z}^{(n)} , \end{aligned} \tag{10.11}$$

where $\mathbb{Z}^{(n)}$ represents here a generic set of integers to which quantized values can be assimilated. Also, due to the nonlinear nature of the output quantization, the finite precision RVFL in (10.10) is no longer linear in the $\beta^{(n)}$ parameters.

ILS is a common problem in many fields of signal processing as, for example, channel coding, cryptography, radar imaging and global positioning [161, 162]. It has been shown that ILS is a NP-hard problem and the algorithms for solving it have exponential complexity [163, 164]. Computationally, an ILS problem is equivalent to an LS where there is a constraint on the quantization of the results. In fact, a constraint on the bounded representation of the weights can be treated as a box constraint and solved similarly to an ILS problem regardless of the bounds [165].

Assumption 10.3. A straightforward approximation of the ILS solution, which will be the baseline for the successive experiments, is to quantize, with a two's complement representation using n bits, the result obtained from (10.6):

$$\boldsymbol{\beta}_{\text{rnd}}^{(n)} = q_n\left(\left(\mathbf{H}^{(n)T} \mathbf{H}^{(n)} + \lambda \mathbf{I} \right)^{-1} \mathbf{H}^{(n)T} q_n(\mathbf{y}) \right) , \tag{10.12}$$

where outputs $\mathbf{y}$ are in general quantized, as this learning step could be performed on a fixed point hardware as well.

As previously said, such an approach is only an approximation of the correct ILS solution and the performance of the resulting RVFL may be worse due to the nonlinear nature of the overall processing system. For such a reason, it is proposed an optimization of the NP-hard nonlinear ILS problem, which is based on a GA approach, in order to obtain an optimal set of quantized $\beta^{(n)}$ parameters.

10.4.1 Genetic Optimization

The genetic algorithm was already introduced in section 8.3.2, now the algorithm 6 will be used to find out the optimal $\beta^{(n)}$ parameters.

In this case, the quantized parameters $\beta^{(n)}$ we are searching for are binary coded in a finite precision arithmetic. These are combined to form the so-called chromosome, which is a string of bits each one treated as a gene. In the proposed approach, the genome of each individual is represented by a binary string of nC bits as illustrated in Fig. 10.2.

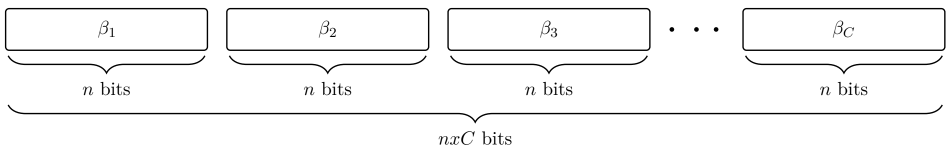


Figure 10.2. Visual scheme of the binary organization of a $\boldsymbol{\beta}^{(n)}$ solution.

Given a dataset on which the GA optimization is being performed, the quantized weights and the related quantized hidden functions $h_m^{(n)}$, $m = 1 \dots C$, are computed once for all at the beginning of the process. Then, for each individual and in every generation, the output (10.10) is computed for all inputs of the dataset by using the particular instance of $\boldsymbol{\beta}^{(n)}$ associated with the chromosome of the individual. The fitness of the individual is the Noise-to-Signal Ratio (NSR):

$$\mathrm{NSR_{dB}} = 10 \log_{10} \frac{\sum_{i=1}^{N} \left(y_i - f^{(n)}(\mathbf{x}_i) \right)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2} , \tag{10.13}$$

where $\bar{y}$ is the average value of the output values y_i on the dataset. Thus, the lower is the NSR the better is the fitness.

10.4.2 Experimental setup

The performances are evaluated on four different public datasets summarized in Table 10.1, which are available on the UCI repository [1]:

- *Airfoil* is a NASA dataset where data of two or three-dimensional airfoil blade section conducted in an anechoic wind tunnel is acquired at various tunnel speeds and angles of attack. The aim is to predict the scaled sound pressure level [166].
- *Concrete* has been already introduced in Sec. 6.3.1.
- *Energy* contains an energy analysis using 12 different building shapes. Varying several characteristics, like the glazing area distribution, the orientation and so on, the aim is to assess the heating load requirement of buildings [167].
- *Istanbul* includes returns of Istanbul Stock Exchange with seven other international indexes [168].

The input and output values of datasets are normalized before training in order to accommodate every feature in the range between 0 and 1. The weights $\mathbf{w}$ of the sigmoid basis functions are extracted randomly from a uniform distribution over the interval $[-1, +1]$, while the scalars b are extracted randomly from a uniform distribution over the interval $[-0.1, +0.1]$. The performance is obtained through a 10-fold cross validation, where the overall NSR is obtained comparing actual values of outputs with the predicted ones in every fold.

[1](https://archive.ics.uci.edu/ml/datasets.html)

Dataset	Features	Instances	Desired output
Airfoil	6	1503	Pressure level
Concrete	9	1030	Compressive strength
Energy	8	768	Heaating Load
Istanbul	8	536	Stock exchange returns

Table 10.1. Detailed Description of Datasets

10.4.3 Discussion

First of all, the standard rounding $\boldsymbol{\beta}_{\mathrm{rnd}}^{(n)}$ parameters obtained from (10.12) is considered as a baseline solution. Four different number of bits for the weight quantization n are then considered: 4, 8, 12, and 16. The performance of such solutions is compared to the one obtained by a software implementation on standard computers using 64-bit floating point arithmetic, which will be assimilated to the analog result $\boldsymbol{\beta}^*$ in (10.6).

In order to compute the optimal number of hidden nodes C and the regularization factor λ, it is executed an inner-fold cross-validation of the training data. In particular, the optimum number of hidden nodes C is searched with a grid-search procedure in the set $\{200, 300, 400, 500\}$, while λ is evaluated in the set $\{2^{-10}, 2^{-9}, \ldots, 2^{9}, 2^{10}\}$.

The optimum C and λ values, for every number of bits, are reported in Table 10.2. It can be noted that the optimum value of C for Airfoil, Concrete and Energy datasets is low for $n = 4$ and $n = 8$ and it increases as the precision increases. Instead, for the Istanbul dataset it is constant (i.e., $C = 300$). The regularization factor has a different trend; as the number of bits varies, the behavior is almost the same for every dataset. In fact, it is as high as possible (i.e., $\lambda = 2^{10}$) for $n = 4$ and then decreases.

Dataset	n=4		n=8		n=12		n=16		float-64	
	C	λ	C	λ	C	λ	C	λ	C	λ
Airfoil	200	2^{10}	200	2^{8}	300	2^{-4}	300	2^{-10}	500	2^{-10}
Concrete	200	2^{10}	200	2^{2}	200	2^{-6}	500	2^{-10}	400	2^{-10}
Energy	200	2^{10}	200	2^{2}	500	2^{-5}	500	2^{-10}	500	2^{-10}
Istanbul	300	2^{10}	300	2^{2}	300	2^{-4}	300	2^{-5}	300	2^{-5}

Table 10.2. Optimal Number of Hidden Nodes (C) and λ Found by The Inner-fold Cross-validation

Taking the optimal choices of C and λ, the baseline performance in terms of NSR versus the number of bits is shown in Table 10.3. For every dataset, the NSR for values of n greater than 8 is similar to the 64-bit floating point precision, with differences as low as 0.1 dB. Therefore, it is worth considering optimized implementations where a number of bits equal to or lower than 8 is considered. The

Dataset	n=4	n=8	n=12	n=16	float-64
Airfoil	-23.816	-26.604	-28.858	-30.175	-30.237
Concrete	-6.644	-11.233	-13.181	-15.038	-15.150
Energy	-8.386	-18.025	-19.909	-23.738	-24.122
Istanbul	1.784	-5.420	-6.674	-6.722	-6.723

Table 10.3. Performance (NSR) of Basic Rounding Vs. Bit Precision

results obtained by using the GA optimization, keeping the same choices of C and λ, are reported in Table 10.4. It can be outlined that GA optimization improves the performance for $n \leq 12$. This can be explained taking into account that, for a higher number of bits, the huge cardinality of the search space makes more sparse and evanescent the GA approach in a relatively limited time.

Dataset	n=4	n=8	n=12	n=16
Airfoil	-25.784	-26.484	-29.067	-30.082
Concrete	-7.982	-11.537	-13.386	-14.673
Energy	-10.779	-18.595	-20.437	-23.137
Istanbul	1.531	-6.026	-6.619	-6.608

Table 10.4. Performance (NSR) of Genetic Optimizer Vs. Bit Precision

Finally, for the sake of illustration, Fig. 10.3 reports the comparison of actual and predicted values of Energy dataset using a software implementation with 64-bit floating point precision, and a GA optimization using 8 bits (Fig. 10.4). Evidently, the behavior is comparable even with a numerical difference of about 5.5 dB.

The experimental results show that the performance gap between the basic rounding and the GA optimization shrinks as the number of bits increases, while the GA procedure obtains better performance using even 4 and 8 bits.

In the following paragraph, it will be seen how happens when the quantization levels become non uniform.

10.5 A nonuniform quantizer

Once tested the RVFL on a uniform quantization over all of the parameters of the network, in this section is proposed a nonuniform quantization of the input data feeding the network, in order to cope with the actual structure of datasets and also to compensate the rounding of internal parameters of the neural model. In effect, it is well-known that, except in the case of the uniform distribution, the non-uniform quantization is superior in the sense that it results in a smaller average quantization error. For these reasons, it is assumed for the rest of the chapter, a nonuniform quantization of input values and a uniform quantization elsewhere.

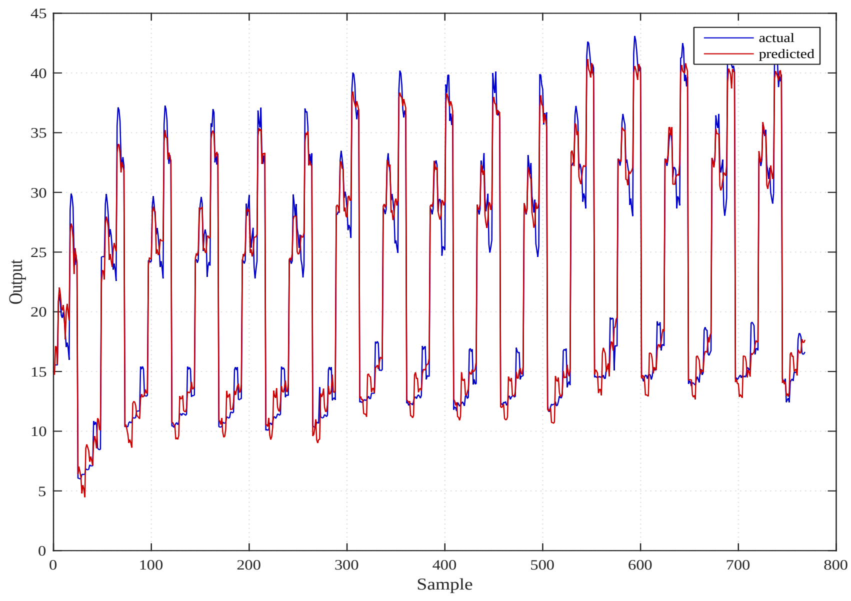


Figure 10.3. Output on Energy dataset using 64-bit floating point precision.

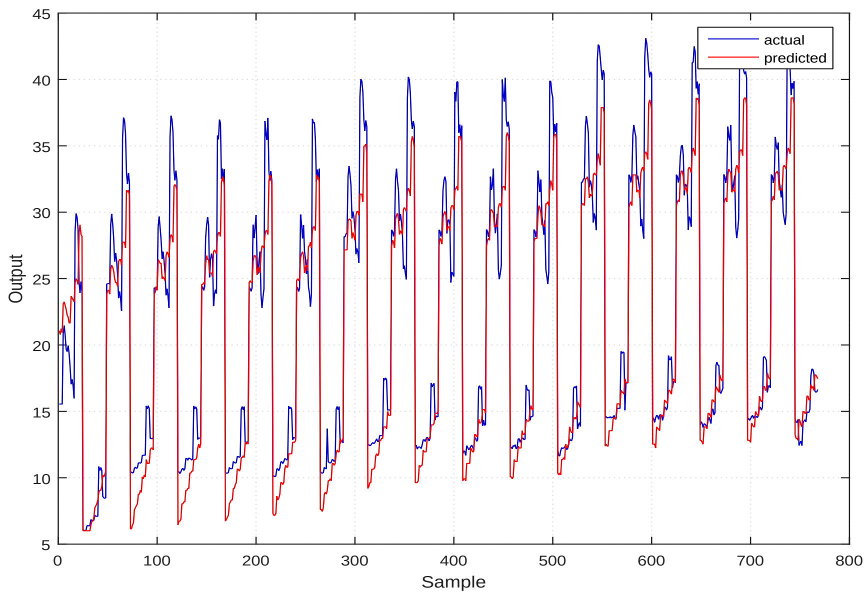


Figure 10.4. Output on Energy dataset using 8-bit precision optimized by GA.

To this end, let introduce $q_n(\cdot)$ as a uniform quantizer within the range of the input variables, with a two's complement representation using n bits. It is applied element-wise if the input is either a vector or a matrix. If the input to q_n lies in the range between $q_{\min}$ and $q_{\max}$, the quantization levels of q_n will be in the form

$q_{\min} + r2^{-n}(q_{\max} - q_{\min})$, $r = 0, 1, \ldots, 2^n - 1$.

Also, let $u_n(\cdot; \theta)$ be a nonuniform quantizer using n bits for representing the quantization values. Since $\mathbf{x}$ is a d-dimensional array, the nonuniform quantizer is applied with different quantization values on each dimension. Thus, θ is a $2^n \times d$ matrix of real numbers and the output $u_n(\mathbf{x}; \theta)$ of the quantizer will be a d-dimensional array where the jth element is $u_{n,j} = \theta_{rj}$, $j = 1 \ldots d$, being $\theta_{rj} \in \theta$ the largest value no greater than x_j in the jth column of θ, for any $1 \leq r \leq 2^n$.

Consequently, given the same generation of random of weights, the quantized hidden matrix $\mathbf{H}^{(n)}$ in this case could be arranged on the basis of the following hidden functions:

$$h^{(n)}(\mathbf{x}; \mathbf{w}, b, \theta) = q_n \left(\frac{1}{1 + \exp\left\{-q_n(\mathbf{w}^T) u_n(\mathbf{x}; \theta) + q_n(b)\right\}} \right) . \tag{10.14}$$

Then, the generic output for an n-bit finite precision implementation of an RVFL will be:

$$f^{(n)}(\mathbf{x}) = q_n \left(\sum_{m=1}^{C} \beta_m^{(n)} h_m^{(n)}(\mathbf{x}; \mathbf{w}_m, \theta) \right) . \tag{10.15}$$

As seen in the previous paragraph, the weights are extracted from a uniform distribution and the nonuniform quantization of input values should also improve the numerical accuracy in representing the quantized outputs of hidden functions, which undergo strong saturation effects. As before, the problem can be reformulated as an Integer Least Square problem:

$$\begin{aligned} &\min_{\boldsymbol{\beta}^{(n)} \in \mathbb{R}^C} \frac{1}{2} \left\| \mathbf{y} - q_n\left(\mathbf{H}^{(n)} \boldsymbol{\beta}^{(n)}\right) \right\|_2^2 + \frac{\lambda}{2} \left\| \boldsymbol{\beta}^{(n)} \right\|_2^2 \\ &\text{s.t.} \quad \boldsymbol{\beta}^{(n)} \in \mathbb{Z}^{(n)} , \end{aligned} \tag{10.16}$$

where $\mathbb{Z}^{(n)}$ refers here to a generic set of integers to which quantized values can be assimilated. Also in this case, it is considered a straightforward approximation of the result obtained in (10.6), by using a finite precision representation:

$$\boldsymbol{\beta}_{\text{rnd}}^{(n)} = q_n \left(\left(\mathbf{H}^{(n)T} \mathbf{H}^{(n)} + \lambda \mathbf{I} \right)^{-1} \mathbf{H}^{(n)T} q_n(\mathbf{y}) \right) , \tag{10.17}$$

where output values $\mathbf{y}$ are in general quantized, as they can be obtained from measures by digital equipment.

10.5.1 Genetic Optimization

Once again, a genetic optimization is used to tune the parameters of the model. Differently, from before, it is used to determine the quantization levels of the non linear A/D converter, considering as analog signals also the data represented in floating point precision.

Hence, the optimization procedure is applied to the θ parameters of the nonuniform quantizer and hence, the chromosome of each individual is represented by an array of $d \cdot 2^n$ real numbers (i.e., quantization values) as illustrated in Fig. 10.5.

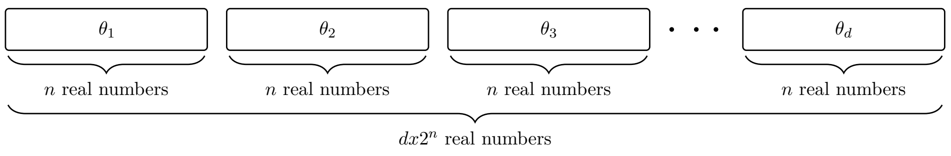


Figure 10.5. A chromosome defining the nonuniform quantizer, where θ_j^T, $j = 1 \dots d$, is the jth column of θ.

It represents a possible solution that implements a nonuniform quantizer. Also in this case, the quantized weights $\mathbf{w}_m$ and the related quantized hidden functions $h_m^{(n)}$, $m = 1 \dots C$, are computed once for all at the beginning of the process. Then, for each individual and in every generation, the RVFL is trained using (10.17) and the hidden matrix $\mathbf{H}^{(n)}$. The former, is evaluated through (10.14) where is used the particular instance of θ associated with the chromosome of the individual. Successively, the output on a test set is computed using (10.15). The fitness of the individual is the Noise-to-Signal Ratio (NSR) (10.13) thus, the lower is the NSR the better is the fitness.

10.5.2 Experimental setup

To validate the proposal, the datasets have been chosen in order to represent a variety of applicative domains and problems. In particular, the performance of

Dataset	n=4		n=8		n=12		n=16		float-64	
	C	λ	C	λ	C	λ	C	λ	C	λ
Airfoil	50	2^9	50	2^0	50	2^{-3}	50	2^{-7}	500	2^{-10}
Concrete	200	2^8	200	2^2	200	2^{-4}	500	2^{-6}	400	2^{-10}
Energy	50	2^9	100	2^0	50	2^{-4}	150	2^{-5}	500	2^{-10}

Table 10.5. Optimal Number of Hidden Nodes (C) and λ Found by The Inner-fold Cross-validation

the proposed approach is evaluated on the three different and well-known datasets available on the UCI machine learning repository [2] and already described in the Sec:10.4.2:

- *Airfoil*;
- *Concrete*;
- *Energy*;

Both input and output values of datasets are normalized before training in order to accommodate every feature in the range between 0 and 1. The weights $\mathbf{w}$ of the

[2]http://archive.ics.uci.edu/ml

sigmoid basis functions are extracted randomly from a uniform distribution over the interval $[-1, +1]$, while the scalars b are extracted randomly from a uniform distribution over the interval $[-0.1, +0.1]$. Every test result of RVFL is obtained through a 10-fold cross validation, where the overall NSR is obtained comparing the actual values of outputs with the predicted ones in every fold. Nonetheless, given the stochastic nature of the RVFL initialization, as well as for GAs, all the performances reported in the following are an average of the results obtained by repeating the same experiment over 10 independent runs.

10.5.3 Discussion

This section describes the details of the proposed implementation. First of all, it is considered the baseline solution where a finite precision RVFL is trained and tested using the above steps and a uniform quantization of input values. Four different values of n (i.e., bits for quantization) are considered: 4, 8, 10, and 12. Higher values of n are not considered because in such cases the hardware complexity makes the performances closer to those achieved by a floating point arithmetic.

The performances are compared with respect to the ones obtained by a software implementation on a standard computer or DSP using 64-bit floating point arithmetic. In order to compute the optimal number of hidden nodes C and the regularization factor λ, it is executed an inner-fold cross-validation of the training data. In particular, it is performed a grid-search procedure where C was varied from 50 to 500 in a step of 50 and λ was in the set $\{2^{-10}, 2^{-9}, \dots, 2^{9}, 2^{10}\}$. The optimum values of C and λ for every number of bits are reported in Table 10.5, respectively. It can be noted that the optimum value of C tends to increase as the precision increases, whilst λ decreases. Taking the optimal choices of C and λ, the performances in terms of NSR versus the number of bits are shown in Table 10.6. As expected, it is worth considering optimized implementations of finite precision RVFLs using 12 or a lower number of bits, as the related performances are different from the one using a floating point architecture. The results obtained by using the GA

Dataset	n=4	n=8	n=12	n=16	float-64
Airfoil	-24.358	-28.055	-28.339	-28.890	-30.369
Concrete	-6.635	-11.233	-11.883	-12.997	-15.326
Energy	-8.578	-18.061	-18.920	-19.862	-24.433

Table 10.6. Performance (NSR) using a Uniform Quantizer of RVFL Inputs

optimization, keeping the same choices of C and λ, are reported in Table 10.7. While the improvements on Airfoil are not relevant, as it probably contains noisy and spread data, for the Concrete and Energy case, the NSR decreases of 1 or 2 dB, more significantly with a higher number of bits as the space of solution enlarges and there is more margin to found a better local optimum. To confirm the efficacy of the proposed approach, Fig. 10.6 reports the quantization levels obtained in the case of a 10-bit finite precision implementation of an RVFL trained on the Energy

Dataset	n=4	n=8	n=12	n=16
Airfoil	-24.453	-28.165	-28.362	-28.851
Concrete	-7.105	-11.573	-11.855	-13.202
Energy	-8.629	-19.424	-21.682	-24.262

Table 10.7. Performance (NSR) using a GA-optimized Nonuniform Quantizer of RVFL Inputs

dataset. The optimal solution found by the GA determines an evident nonuniform distribution of the quantization levels, which is also different for each dimension. Finally, in Fig. 10.7 is reported the comparison of actual and predicted values of

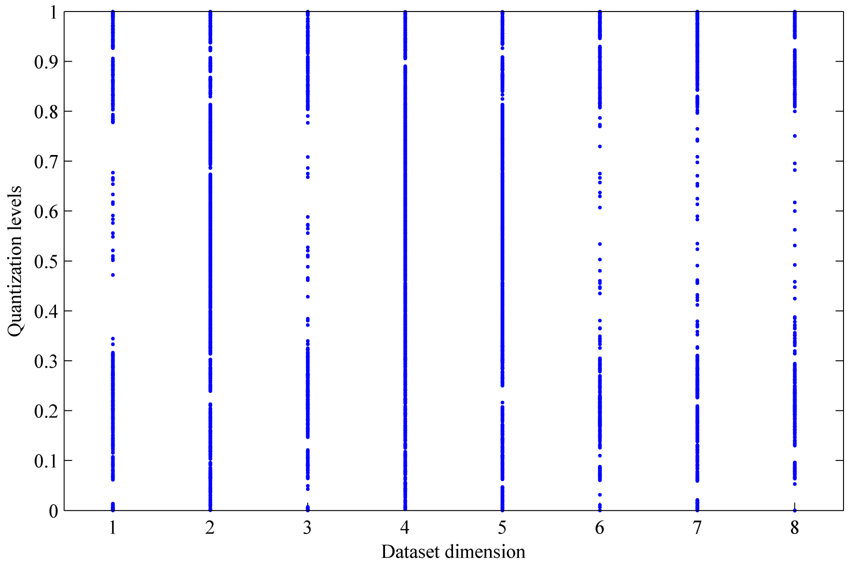


Figure 10.6. Quantization levels of a 10-bit nonuniform quantizer optimized by GA on the Energy dataset.

Energy dataset using an 8-bit nonuniform quantizer optimized by GA. Evidently, the behavior is comparable with a numerical difference of about 1.4 dB and for NSR values around 19 dB.

The numerical results show that the proposed approach is effective even using a small number of bits and it compensates the effects of rounding in the successive layers of the neural networks.

10.6 Conclusion

In this chapter, the problem of distributed learning on pervasive sensor networks and multiple sources of big data has been treated. This scenario needs computationally efficient techniques on simple and cheap hardware architectures. A modified version of RVFL, with finite precision arithmetic, is adopted. In particular, an RVFL can

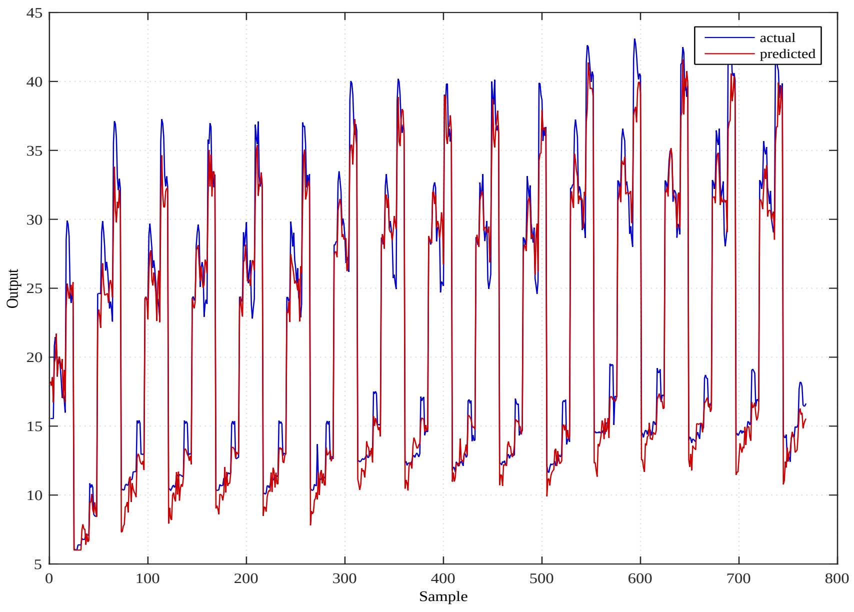


Figure 10.7. Output on Energy dataset using a 8-bit precision and GA optimization.

be implemented on each node of a network of interconnected agents where the constraints imposed by the hardware implementation have been firstly dealt with a uniform quantization on all of the parameters of the model.

Successively, a nonuniform quantization is introduced at the input layer of the neural network. Namely, it has been proposed a nonlinear A/D conversion of input signals by considering the actual structure of data to be processed. In both cases, a genetic optimization is adopted for optimize the performances of the model and tune the quantization levels of both the input data and the internal parameters.

The proposed approaches are assessed by several experimental results obtained on well-known benchmarks for the general problem of data regression. In particular, it has been shown that they are effective even using a small number of bits and it compensates the effects of rounding in the successive layers of the neural network. Hence, they are suited for those contexts where are allowed low computational power at every location and limited underlying connectivity (e.g. no broadcasting capability).

Future works might consider hardware architectures and finite precision adaptive, online strategies reflecting the numerical results herein obtained.

Chapter 11

Prediction in photovoltaic power plants

In this chapter, it is introduced another application for the distributed learning problem. It is related to the capability of model identification and the accuracy of renewable energy plants prediction, in the context of a highly connected network of agents. The main task to be solved to this end is the prediction of every time series related to the measure of some physical parameters (voltage, current, power, wind speed, temperature, etc). Then, agents are enabled to interact and combine such results in a decentralized hierarchical network, so as to exploit the information correlating the stations (e.g. photovoltaic panels), such as movement of clouds, gradient of temperature, and so forth.

In the following, the basic problem of time series prediction in power plants will be focused on, considering four techniques based on neural and fuzzy neural networks: Adaptive Neuro-Fuzzy Inference System (ANFIS); Mixture of Gaussian (MoG); Radial Basis Function (RBF); Higher-Order Neuro-fuzzy Inference System (HONFIS). As it will be illustrated, a predictor based on such function approximation models, whose parameters are estimated by data-driven learning procedures, will be adopted to perform prediction of solar photovoltaic outputs.

In the recent years, distributed generation (DG) has significantly penetrated into the electrical power systems' infrastructure, leading to a new scenario where centralized bulk systems are being progressively replaced by decentralized ones. They enable a new way of seeing the energy production, where smaller generating units are directly connected, through a network, to the consumer. Although, performing inference on data which is partitioned over geographically distinct locations is considered a fundamental problem in many scientific endeavors, renewable energy systems are the most promising, being both wind-electric conversion and photovoltaic systems in advances stages of development.

The diffusion of DG in the energy market, especially from renewable resources, e.g. solar power and wind energy, has received a considerable boost by several economical and environmental reasons, as well as by government financial incentives [169, 170, 171]. However, the entrance of DG with a significant penetration level into the current distribution infrastructure, is often impeded by technical barriers. In fact, the intermittent nature of the renewable energies often do not match the energy load demands [172]. Furthermore, variability and reliability characterizes these non-dispatchable resources [173]. Hence, the usual habits of working of transmission system operator, utility companies, as well as power producers, e.g., voltage and frequency regulation, islanding detection, harmonic distortion, electromagnetic interference, are particularly challenging tasks of interest [174, 175, 176, 177]. Certainly, energy storage systems (EES) could become a valuable response to provide specific support services for renewable energy production. In this way, short-term output fluctuations will be reduced and, consequently, the quality will be improved.

In this chapter, it is underlined the paramount importance to forecast accurately the power output of plants for the next hours or days. The aim is to allow an optimal integration of the DG from non-programmable renewable resources into large scale power systems. The ability to forecast the amount of power fed by renewable energy plants into substations is a necessary requirement for the Independent System Operator (ISO). In this way, short and middle term decisions such as connections to out-of-region trade and unit commitments are optimized. Besides, accurate short-term and intra-hour forecasting are required for regulatory, dispatching and load following issues, while power system operators are more sensitive to intra-days forecasts, especially when handling multiple load zones. In effect, they avoid possible penalties that are incurred to the deviations between forecasted and produced energy.

Forecasting techniques can be broadly divided into four major's categories, namely, long, medium, short and very short-term forecast. It depends on the time horizon's scale, ranging from several years in the long-term forecasting to hours in the medium forecasting, till arriving in minutes in the very short term forecasting. When facing to photovoltaic (PV) power plants, forecasting techniques can be applied directly or indirectly. The direct methods are directly applied on the produced power, while the indirectly ones are initially interested in evaluating the solar irradiation; the produced power is computed only in a subsequent step. Both cases share similar techniques, which can be broadly divided into persistence methods, physical techniques, statistical techniques, and hybrid models. Usually, physical methods try to obtain an accurate forecast using a white box approach, that is to say, physical parametric equations. Oppositely, statistical methods predict the future behavior of a series by using and extracting dominant relations from past data. In addition, both methods can be properly mixed in order to originate the so-called gray box approach.

Generally, in the prediction of time series, the statistical models are the most promising. Being able to extrapolate future values by using past observations, they are usually adopted for describing the underlying relationships among data. They are helpful especially for those contexts where missing and incomplete data can limit the ability to gain knowledge about the unknown process generating observations. In fact, complexity and dynamics of real-world problems require advanced methods and tools able to use past samples of the sequence in order to make an accurate prediction. Additionally, the problem of forecasting future values of a time series is often mandatory for the cost-effective management of available resources.

For many years, regressive statistical approaches have been the only ones considered for prediction; among them, it could be found out Moving Average (MA), Auto Regressive (AR), Auto Regressive Moving Average (ARMA), Auto Regressive Integrated Moving Average (ARIMA) and Autoregressive Integrated Moving Average with Exogenous Variables (ARIMAX). Unfortunately, standard structural models provide a poor representation of actual data and therefore result in a poor accuracy when used for forecasting.

Consequently, many worldwide research activities are intended to improve the accuracy of prediction models. In this regard, computational intelligence is considered as one of the most fruitful approaches. Several forecasting methods, with different mathematical backgrounds, such as fuzzy predictors, artificial neural networks, evolutionary and genetic algorithms, and support vector machines have resorted. However, the stochastic and often chaotic behavior of time series in this field makes necessary ad-hoc procedures to forecast power time series.

The main idea presented in this chapter is to properly tailor new learning approaches for analyzing data associated with renewable power sources [178, 179]. Starting from available techniques based on computational intelligence [180], the goal is to provide intuition into how solar-panel power generation depends on a combination of multiple weather metrics. Such results will be the basis for the successive developments of fully distributed tools in the same context.

The complexity in predicting solar intensity from one or more weather metrics is the main motivation for the next section, where automatically generating prediction models, able to threat chaotic sequences, are introduced. The preliminary results concerning the prediction of power generated by a large-scale photovoltaic plant, in Italy, show that all of the considered approaches are suitable for the proposed goal in normal operating conditions.

11.1 The function approximation approach to time series prediction

There is a huge amount of literature pertaining to time series prediction techniques that rely on neural and fuzzy neural networks [181]. In effect the solution of a prediction problem, can coincide with data regression solved by using neural networks and fuzzy logic [182, 183].

A sequence $S(t)$ can be predicted by considering a function approximation $y = f(\underline{x})$, $f : \mathbb{R}^N \to \mathbb{R}$. For instance, by using linear models each input vector $\underline{x}_t$ is made of N subsequent samples of $S(t)$ and the output y_t is the sample to be predicted:

$$\underline{x}_t = [S(t)\ S(t-1) \dots S(t-N+1)] \ , \ \ y_t = S(t+m) \ , \ \ f_{\text{lin}}(\underline{x}_t) = \sum_{j=1}^{N} \lambda_j x_{tj} \ . \quad (11.1)$$

Then, we obtain

$$\widetilde{S}(t+m) = \sum_{j=1}^{N} \lambda_j S(t-j+1) \ , \quad (11.2)$$

where $\widetilde{S}(t+m)$ is the prediction of the true value $S(t+m)$ at the time distance m. Considering the statistical properties of $S(t)$, as the autocorrelation function, it is possible to determinate the parameters λ_j, $j = 1 \dots N$, of the function $f_{\text{lin}}(\cdot)$.

Generally speaking, input vectors $\underline{x}_t$ are obtained through the so called 'embedding technique', which makes use of previous samples of $S(t)$ to build the vectors themselves. Two are the parameters to be set in this regard, the embedding dimension D and the time lag T, resulting in the following embedding structure:

$$\underline{x}_t = \left[S(t)\ S(t-T)\ S(t-2T)\dots\ S(t-(D-1)T)\right]. \tag{11.3}$$

These two parameters are estimated in two different ways. The first one is evaluated by using the False Nearest Neighbors (FNN) algorithm, while the Average Mutual Information (AMI) criterion is used to find out the time lag [184]. Subsequently, D and T will be chosen by such algorithms thanks to the VRA software [1].

The performances of a predictor that relies on a linear approximation model, as the one in (11.1), are very poor when it is applied to real environments' data sequences. In effect, theirs noisy and chaotic components force the model to wisely choose the embedding parameters, as well as the function approximation model [185].

Conversely, in this context, the underlying, unknown system is observed through $S(t)$ and its state-space evolution is obtained by the trajectories of embedded vectors even in the presence of a non linear behavior of the series.

Overall, the estimated sample $\widetilde{S}(t+m)$ predicted at a distance m will be:

$$\widetilde{S}(t+m) = f\left(\underline{x}_t\right), \tag{11.4}$$

where $f(\cdot)$ is the regression model to be determined. Thus, the function $f(\cdot)$ will approximate the link from the reconstructed state $\underline{x}_t$ to the corresponding output y_t [184]. It is important to note that $f(\cdot)$ must be non-linear since the considered system has a complex behavior. In the following it will be considered the usual case of a 'one-step-ahead' prediction, that is $m = 1$.

By driving the function approximation model with embedding data, neural networks and fuzzy logic are very suited to solve a prediction problem dealing with real-world sequences. In the next section, four models of this kind are proposed for study photovoltaic time series. They are introduced below, considering as a baseline for benchmarking in the successive tests: the linear predictor reported in (11.1), whose parameters are estimated through a common least-squares estimator (LSE) [186]; the well-known ARIMA model, which handles non-stationarity and seasonality and whose parameters are determined by a maximum likelihood approach [187]. Both these models will be estimated by using lagged samples at the input, as determined by the embedding procedure similarly to the adopted neural networks.

11.2 Neural and fuzzy neural approaches

In the following sections, four models are proposed for this particular study.

11.2.1 ANFIS

The ANFIS neural network has been already introduced in Sec:6.1.1.

It is composed of M rules of Sugeno first-order type to approximate a function $y = f(\underline{x})$, $f : \mathbb{R}^N \rightarrow \mathbb{R}$.

[1]http://visual-recurrence-analysis.software.informer.com/4.9/

The structure of a fuzzy rule has also already introduce, thus, in this work the rule output are combined with the input MFs [89], to represent the output of the ANFIS network as:

$$\widetilde{y} = \frac{\sum_{k=1}^{M} \mu_{\underline{B}^{(k)}}(\underline{x})\; y^{(k)}}{\sum_{k=1}^{M} \mu_{\underline{B}^{(k)}}(\underline{x})} \;, \tag{11.5}$$

where $\widetilde{y}$ is the estimation of y and $\mu_{\underline{B}^{(k)}}(\underline{x})$ is the composed MF of the kth rule.

The ANFIS learning of network coefficients is based on a classical Back-Propagation technique associated with a least-squares estimation [86].

11.2.2 MoG

In the MoG neural network M different Gaussian components in the joint input-output space $\Re^N \times \Re$ are used in a mixture density model:

$$p(\underline{x}, y) = \sum_{j=1}^{M} \pi^{(j)} G^{(j)}_{\underline{x},y}(\underline{x}, y)\,, \tag{11.6}$$

where $\pi^{(j)}$ is the prior probability of the jth Gaussian component $G^{(j)}_{\underline{x},y}$, $j = 1 \dots M$. The mixture parameters are estimated by the Splitting Hierarchical Expectation Maximization (SHEM) technique [104].

The conditional density $p(y|\underline{x})$ is computed from (11.6):

$$p(y|\underline{x}) = \sum_{j=1}^{M} h_j(\underline{x}) G^{(j)}_{y|\underline{x}}(y)\,, \tag{11.7}$$

where $G^{(j)}_{y|\underline{x}}(y)$, $j = 1 \dots M$, is the conditional density of $G^{(j)}_{\underline{x},y}$, and $h_j(\underline{x})$ is the projection of $G^{(j)}_{\underline{x},y}$ into the input space. It can be evaluated by the marginal density $G^{(j)}_{\underline{x}}$ of $G^{(j)}_{\underline{x},y}$:

$$h_j(\underline{x}) = \frac{\pi^{(j)} G^{(j)}_{\underline{x}}(\underline{x})}{\sum_{k=1}^{M} \pi^{(k)} G^{(k)}_{\underline{x}}(\underline{x})}\,. \tag{11.8}$$

A least-square estimation is used to predict the output y corresponding to the input $\underline{x}$:

$$\widetilde{y} = \sum_{j=1}^{M} h_j(\underline{x}) \underline{m}^{(j)}_{y|\underline{x}}\,, \tag{11.9}$$

where $\underline{m}^{(j)}_{y|\underline{x}}$ is the conditional mean of each mixture component in (11.7).

11.2.3 Radial Basis Function

A Radial Basis Function (RBF) neural network is used to build up a function approximation model having the following structure:

$$f(\underline{x}) = \sum_{i=1}^{M} \lambda_i \phi(\|\underline{x} - \underline{c}_i\|)\,, \tag{11.10}$$

where $\underline{x} \in \mathbb{R}^N$ is the input vector, $\phi(\cdot)$ is a radial basis function centered in $\underline{c}_i$ and weighted by an appropriate coefficient λ_i. The approximation capability depends on the choice of $\phi(\cdot)$ and $\underline{c}_i$. Commonly used types of radial basis functions include:

- *Gaussian*
$$\phi(r) = e^{-(\epsilon r)^2}\,; \tag{11.11}$$

- *Multiquadric*
$$\phi(r) = \sqrt{1 + (\epsilon r)^2}\,; \tag{11.12}$$

- *Inverse Quadratic*
$$\phi(r) = \frac{1}{1 + (\epsilon r)^2}\,. \tag{11.13}$$

Several methods can be used to minimize the error between the desired output and the model one and hence, to identify the parameters $\underline{c}_i$ and λ_i [188].

11.2.4 HONFIS

This section introduces a new algorithm able to solve the power prediction problem.

Model formulation

A Takagi-Sugeno (TS) fuzzy inference system for data regression is introduced, it is referred to HONFIS model. It is a generalization of ANFIS where the consequent part in (6.1) is a polynomial of order greater than one that combines the input values x_j, $j = 1 \ldots N$. It uses C different fuzzy rules to find out the scalar output $y^{(k)}$ associate with the rule (11.5).

In this work, a training set of N_R input-output pairs $(\mathbf{x}_n, y_n)$ will be considered, where $\mathbf{x}_n$ is an embedded vector at time n, as in (11.3), and y_n is the related sample $S(n+1)$ to be predicted at the usual distance $m = 1$. Thus, the training set $\mathcal{R}$ will be represented by the N_R integers associated with the time indexes $n \in \mathcal{R}$ of the embedded (and already observed) pairs $(\mathbf{x}_n, y_n)$, which are adopted for learning the parameters of the regression model.

In order to maximize the generalization capability of the approach, with the aim to reduce overfitting in case of noise and ill-conditioned data, is reported a constructive procedure for the automatic determination of the rule parameters. In effect, the generalization capability is maximized only if the TS system consists of a suitable number of rules. The regularization of the network architecture is performed by using learning theory and hyperplane clustering methods [189, 190, 191].

As follows, it is illustrated how the parameters of the TS rules are determined through an alternating optimization technique in the joint input-output data space. Let us introduce $\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_C\}$ as the set of C clusters (each associated with a rule output) to be determined. Supposing that every pattern of the training set is assigned randomly to one of these clusters during the initialization phase, the cluster prototypes are estimated by the following iterative steps:

- *Step 1*. The coefficients $\omega^{(k)}$, $k = 1 \ldots C$, of each rule consequent are evaluated by solving a set of N_k nonlinear equations through an iterative least squares estimation [192]. The generic equation is:

$$y_t = h\left(\mathbf{x}_t; \omega^{(k)}\right) , \tag{11.14}$$

  where: y_t is the output associated with the input $\mathbf{x}_t$; index 't' spans the set $\mathcal{R}^{(k)}$ of pairs of the training set assigned to the kth cluster only, that is $t \in \mathcal{R}^{(k)} \subseteq \mathcal{R}$; N_k is the cardinality of $\mathcal{R}^{(k)}$.

- *Step 2*. By using the new values of coefficients $\omega^{(k)}$, all the sets $\mathcal{R}^{(k)}$, $k = 1 \ldots C$, are updated by assigning each pair $(\mathbf{x}_n, y_n)$, $n \in \mathcal{R}$, of the training set to the cluster Γ_{q_n}, $1 \leq q_n \leq C$, scoring the minimum approximation error, which is therefore defined as:

$$\begin{aligned} d_n &= \left| y_n - h\left(\mathbf{x}_n; \omega^{(q_n)}\right)\right| \\ &= \min_{k=1\ldots C} \left| y_n - h\left(\mathbf{x}_n; \omega^{(k)}\right)\right| . \end{aligned} \tag{11.15}$$

- *Step 3*. The convergence is based on the quantity:

$$\Theta = \frac{\left| D - D^{(old)} \right|}{D^{(old)}} , \tag{11.16}$$

  where D is the *global* approximation error over the whole training set in the current iteration, which is defined as

$$D = \frac{1}{N_R} \sum_{n \in \mathcal{R}} d_n , \tag{11.17}$$

  and $D^{(old)}$ is the global approximation error calculated in the previous iteration.

  The algorithm stops when Θ is less than a predetermined threshold θ (set by default to 0.01). Otherwise it goes back to Step 1 (by using the current updated association of patterns to clusters) and it repeats until the stopping criterion will be satisfied.

At the end of the previous procedure, only the consequents of Sugeno-type rules are obtained. To complete the structure of the TS system, and calculate the output $\tilde{y}$ by using (6.1) and (11.5), the firing strengths of the rules' antecedents must be evaluated too. Let q_n, $1 \leq q_n \leq C$, be the label representing the rule (cluster) to which the nth pair, $n \in \mathcal{R}$, of the training set has been assigned during Step 2 of the latest iteration. In this way, a classification model able to assign a fuzzy label to any pattern of the input space must be determined.

To this end, it has been proposed a K-nearest neighbor (K-NN) classification strategy [121] in order to determine the firing strengths. Namely, let $\mathbf{x}_{n_1}, \mathbf{x}_{n_2}, \ldots, \mathbf{x}_{n_K}$ be the K patterns of the training set that score the smallest Euclidean distance from any input $\mathbf{x}$; the firing strengths of $\mathbf{x}$ will be determined as:

$$\mu_{\mathbf{B}^{(k)}}(\mathbf{x}) = \frac{1}{K} \sum_{r=1}^{K} \mu_{\mathbf{B}^{(k)}}(\mathbf{x}_{n_r}) , \quad k = 1 \ldots C , \tag{11.18}$$

where

$$\mu_{\mathbf{B}^{(k)}}(\mathbf{x}_{n_r}) = \begin{cases} 1, & k = q_{n_r} \\ 0, & k \neq q_{n_r} \end{cases} \tag{11.19}$$

Thus, the output $\tilde{y}$ is calculated through (11.5) by means of the firing strengths contained in the fuzzy label and the output consequents whose parameters have been early determined by clustering in the joint input-output space. For instance, in the case of a 3rd order polynomial as adopted in the following (sec. 11.3.2), the kth rule's consequent, $k = 1 \dots M$, is:

$$\begin{aligned} y^{(k)} = \; & a_{31}^{(k)} x_1^3 + \cdots + a_{3N}^{(k)} x_N^3 + \\ & a_{21}^{(k)} x_1^2 + \cdots + a_{2N}^{(k)} x_N^2 + \\ & a_{11}^{(k)} x_1 + \cdots + a_{1N}^{(k)} x_N + a_0^{(k)} \; . \end{aligned} \tag{11.20}$$

As follows, a simple toy problem is considered in order to illustrate the properties of regression and convergence of the proposed approach. This example considers a quadratic function for the output of the k-th rule of the TS system, $k = 1 \dots C$:

$$\begin{aligned} h\left(\mathbf{x}; \omega^{(k)}\right) = \; & \omega_{21}^{(k)} x_1^2 + \cdots + \omega_{2D}^{(k)} x_D^2 + \\ & \omega_{11}^{(k)} x_1 + \cdots + \omega_{1D}^{(k)} x_D + \omega_0^{(k)} \; . \end{aligned} \tag{11.21}$$

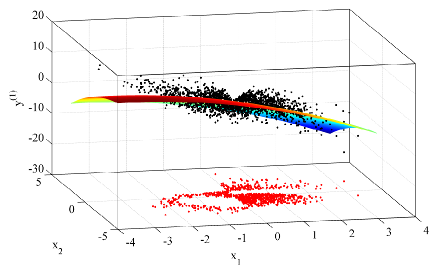
A dataset of 2000 points is generated by the function:

$$y = \begin{cases} 1 + x_1^2 - 9.5\epsilon_1 x_1 + 1.5\epsilon_2 x_2 + 0.4\epsilon_3, & x_1 \leq 0 \\ 1 - x_1^2 + 1.5\epsilon_4 x_1 + 8.0\epsilon_5 x_2 - 0.1\epsilon_6, & x_1 > 0 \end{cases} \tag{11.22}$$

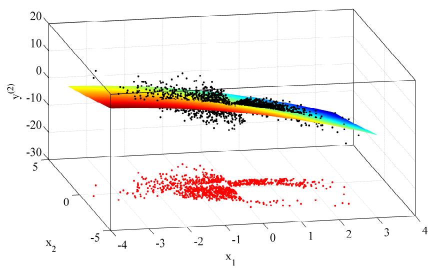
where each pattern $\mathbf{x} = [x_1 \;\; x_2]$ in the input space is obtained by pseudorandom values drawn from a standard normal distribution (zero mean and unitary standard deviation) independently for the two coordinates x_1 and x_2; ϵ_i, $i = 1 \dots 6$, are random values drawn from a standard normal distribution independently of each other.

A TS system having two rules is chosen for the regression (training) of this dataset. In the initialization every pattern of the training set is assigned randomly to one of the two clusters representing the rules. After that, the first iteration of the clustering procedure takes place and the consequents of rule 1 and 2 are the ones illustrated in Fig. 11.1-a and Fig. 11.1-b, respectively. It is evident that the underlying kernels of the dataset, discriminated by $x_1 \leq 0$ or $x_1 < 0$, are not determined accurately; consequently, also considering the firing strengths of the rule antecedents, the total approximation of the dataset is poor, as illustrated by two different 3-D projections in Fig. 11.1-c and Fig. 11.1-d, respectively.

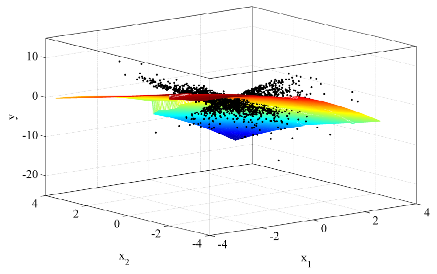
The proposed clustering procedure in the joint input-output space converges very rapidly after 13 steps. At the end, the consequents of rule 1 and 2 are the ones illustrated in Fig. 11.2-a and Fig. 11.2-b, respectively. The two kernels of the dataset, discriminated by $x_1 \leq 0$ or $x_1 < 0$, are determined accurately now; consequently, also considering the firing strengths of the rule antecedents, the total approximation of the dataset is satisfactory, as illustrated by two different 3-D projections in Fig. 11.2-c and Fig. 11.2-d, respectively.



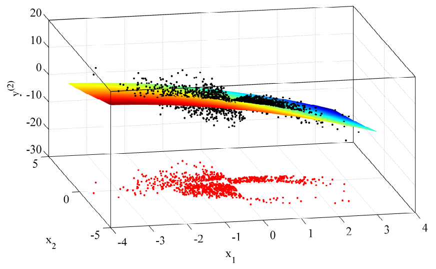
(a) Consequent of rule 1



(b) Consequent of rule 2

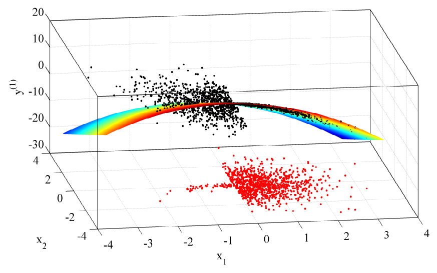


(c) A first 3-D projection.

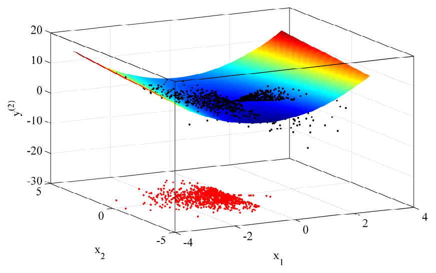


(d) A second 3-D projection

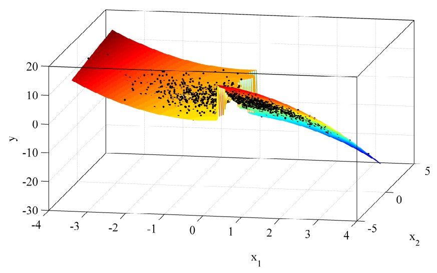
Figure 11.1. In the panel on the top is reported an approximation of dataset (11.22) by the consequent of rule 1-2 after the first iteration of the clustering procedure: patterns projected onto the input space are the ones assigned to rule 1 and 2 by the 3-NN classification procedure. In the bottom panel a global approximation of dataset (11.22) after the first iteration of the clustering procedure is reported.



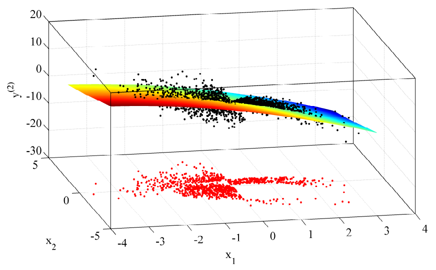
(a) Consequent of rule 1



(b) Consequent of rule 2



(c) A first 3-D projection.



(d) A second 3-D projection

Figure 11.2. In the panel on the top is reported an approximation of dataset (11.22) by the consequent of rule 1-2 after the convergence of the clustering procedure: patterns projected onto the input space are the ones assigned to rule 1 and 2 by the 3-NN classification procedure. In the bottom panel a global approximation of dataset (11.22) after the convergence of the clustering procedure is reported.

Optimization of The Network Complexity

The main problems with the learning in a TS model are the right determination of the number of rules C and the local convergence of the algorithm for the synthesis of rules (which depends upon the random initialization of the parameters of each rule).

Regarding the generalization capability, the optimal number of rules is automatically determined on the basis of learning theory [193, 194]. Specifically, different values of C and different initialization for every value of C are considered to solve the previous problem. Then, the best model is chosen by evaluating a cost function based on network complexity and approximation error [183, 182].

In particular, the mean squared error (MSE) is adopted as a measure of the network performance on the training set:

$$E = \frac{1}{N_R} \sum_{n \in \mathcal{R}} (y_n - \tilde{y}_n)^2 \ , \tag{11.23}$$

where N_R is the number of samples in the training set to be predicted, and $\tilde{y}_n$ is the output generated by HONFIS in correspondence to the t-th input pattern $\underline{x}_n$ of the training set.

The optimal network is selected by maximizing the following cost function that depends upon the maximum allowed number of the HONFIS rules:

$$F(C) = (1-\lambda)\frac{E(C) - E_{\min}}{E_{\max} - E_{\min}} + \lambda \frac{C}{N_R} \, , \tag{11.24}$$

where $E_{\min}$ and $E_{\max}$ are the extreme values of the performance E that are encountered during the analysis of the different TS systems; λ is a weight in the range $0 \leq \lambda \leq 1$, which will be set by default in middle of the interval at 0.5.[2].

The best network is the one showing the best performance $E(\bar{C})$ on the training set. To maximize (11.24), the number of rules is varied from 1 to $C_{\max}$, where the maximum value is the highest complexity that the network can allow. If Q different initializations are carried out for each value of C, the optimization procedure will generate $QC_{\max}$ networks and the optimal one will be selected according to (11.24). The number of initializations has to be determined without increasing too much the computational cost while having a good TS system after the clustering [104].

11.3 Illustrative Tests

In this section, the prediction performances of the considered models were investigated by means of several simulation tests, carried out using data from the De Nittis Power Plant in the Apulia Region, Italy (latitude $\phi = 41°26'16''$, longitude $\lambda = 15°45'47''$). Data is relative to a single photovoltaic plant, organized with eight cabins with two inverters each. The output current is used as the variable to be predicted. The value is sampled with a 5 minutes sample interval and it is collected from 6:00 AM to 10:55 PM (resulting in 204 samples per day). The data stream used is of a single inverter of a single cabin and it is relative to the year 2012. In order to

[2] This weight is found to be not critical, since the results are slightly affected by its variation in a large interval centered in 0.5

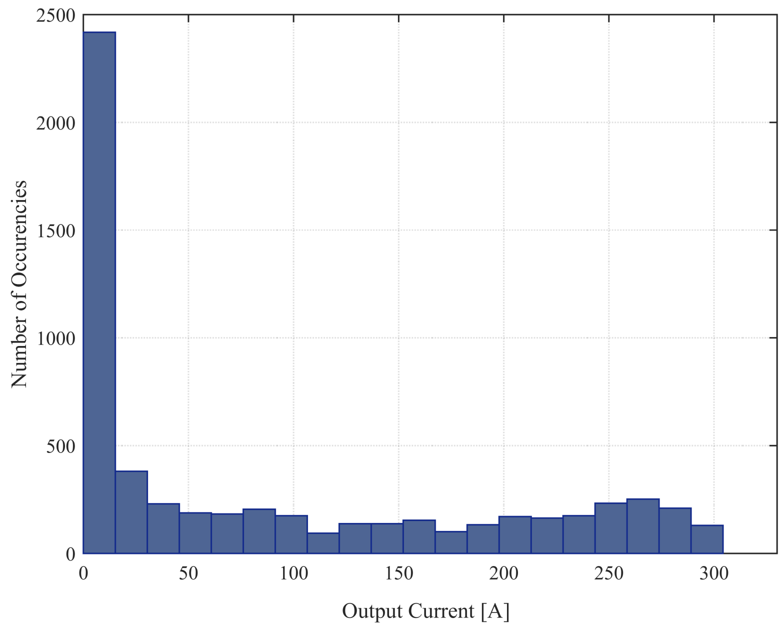


Figure 11.3. Histogram of the output current.

provide a statistical characterization of the handled time series, whose samples are measured in Ampere, the first four statistical moments of the whole 2012 dataset are computed and the relative histogram is reported in Fig. 11.3: Mean 85.1291; Variance $1.0244 \cdot 10^4$; Skewness 0.8151; Kurtosis 2.1082. Successively, the whole time series has been normalized linearly between 0 and 1 in order to cope with the numerical requirements of learning algorithms, especially for neural network models. The prediction performances are measured by different metrics commonly adopted for energy time series, which are independent of the said procedure for data normalization:

Definition 11.1. The Mean Squared Error (MSE) is a risk function that measures the average of the squares error. In other words, it evaluates the difference between the prediction $\tilde{y}$ and the real values y . It corresponds to the expected value of the squared error loss or quadratic loss:

$$\mathrm{MSE} = \frac{1}{N_S} \sum_{t \in \mathcal{T}} (y_t - \tilde{y}_t)^2 \, ; \tag{11.25}$$

Definition 11.2. The Normalized Mean Square Error (NMSE), similarly to the MSE, measures the overall deviations between predicted $\tilde{y}$ and measured values y:

$$\mathrm{NMSE} = \frac{\sum_{t=1}^{N_T} (y_t - \tilde{y}_t)^2}{\sum_{t=1}^{N_T} (y_t - \bar{y})^2} \, , \tag{11.26}$$

where $\bar{y}$ is the average value of samples y_t in the test set. Unlike it, the NMSE normalized the data by the squared error of the predicted value and the average of the samples.

Definition 11.3. The Noise to Signal Ratio (NSR) (in dB) can be defined as the inverse of the Signal to Noise Ratio (SNR). It is expressed as the ratio between the noise power and the signal power. A ratio higher than 1:1 indicates more noise to ratio.

$$\mathrm{NSR_{dB}} = -10\log_{10}\frac{\sum\limits_{t\in\mathcal{T}}(y_t-\tilde{y}_t)^2}{\sum\limits_{t\in\mathcal{T}} y_t^2}\,; \tag{11.27}$$

Definition 11.4. Mean Absolute Range Error (MARE) measures the absolute deviation as a proportion of the maximum possible error. Compared to the RMSE, it is dimensionless and incorporates the inherent property of the data structure being analyzed. It can be calculated as follows:

$$\mathrm{MARE} = \frac{1}{N_T}\sum_{t=1}^{N_T}\frac{|y_t-\tilde{y}_t|}{y_{\max}-y_{\min}}\,, \tag{11.28}$$

where $y_{\max}$ and $y_{\min}$ are the maximum and minimum value of samples y_t in the test set, respectively.

All the experiments have been performed using MATLAB® 2016b, running on a 3.1 GHz Intel Core i7 platform equipped with 16GB of memory.

11.3.1 First set trials

In this section, three of the introduced models are trained using either 7-days or 30 days series with no subsampling (resulting in a training set of 1428 and 6120 samples, respectively). The embedding parameters D and T were chosen analyzing the time series using the 'VRA 4.2' software package [195]. It is very interesting to note that the time delay value (i.e., $T = 54$) can be associated with the intrinsic properties of the solar time series itself. In fact, most of the information of the time series is contained in three or four samples lagged about 4 hours each. This is because the irradiation of the solar panel has a periodic component (linked to the sun rising and setting).

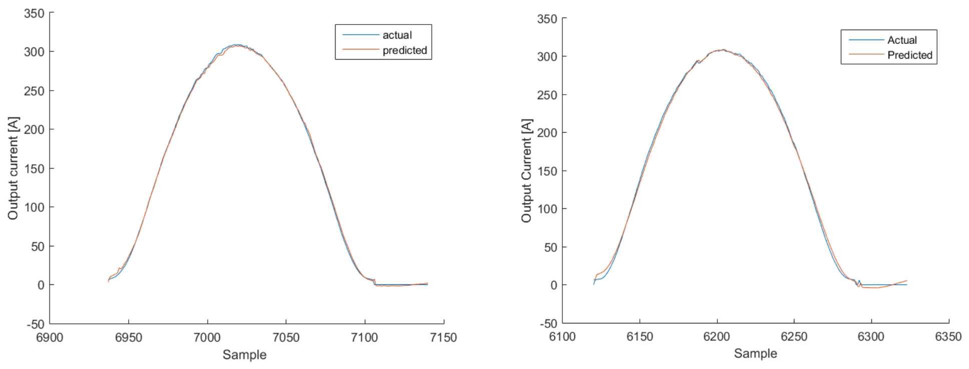
A typical prediction behavior is presented in Table 11.1 and Table 11.2 where the predicted data (i.e., test set) is relative to one day in the month of June 2012 and the training set is based on the previous 7 or 30 days, respectively. The results are reported in terms of the four metrics before introduced, the NMSE (eq. (11.26)), the MSE (eq. (11.25)) the NSR (eq. (11.27)) and the MARE (eq.(11.28)).

The graphical illustration of the actual and predicted time series obtained by using the model performing the best result for such a day, is shown in Fig. 11.4-1 and Fig. 11.4-b, respectively.

From the obtained results, it can be claimed that in normal operational conditions all of the presented solutions are suitable for the proposed goal. In particular, the tests are very accurate when considering a 7-days training set, mostly because in a single week window the solar irradiation is more stable. Good LSE results can

Predictor	NMSE	MSE	NSR (dB)	MARE
LSE training	0.00182	24.37	-31.60	0.00580
LSE test	0.00024	3.340	-40.23	0.00483
RBF training	0.00111	14.96	-33.71	0.00478
RBF test	0.00021	2.898	-40.91	0.00390
ANFIS training	0.00114	15.32	-33.61	0.00468
ANFIS test	0.00350	48.26	-28.63	0.14730
MoG training	0.03804	510.6	-18.39	0.01286
MoG test	0.00039	5.392	-38.15	0.00603

Table 11.1. Prediction Results Using the 7 Days Training Set



(a) RBF **(b)** LSE

Figure 11.4. Prediction using the best model LSE and RBF using the 7 and 30 days training set respectively.

be explained taking in account the intrinsic periodicity of the sequence, given the inherited structure of the dataset and the day/night natural cycle.

In the next section, the experimental trials are changed by adding the HONFIS model as well as the environmental conditions.

Predictor	NMSE	MSE	NSR (dB)	MARE
LSE training	0.08875	1214.6	-14.18	0.04715
LSE test	0.00097	13.69	-34.24	0.00999
RBF training	0.04798	656.7	-16.85	0.04005
RBF test	0.00163	22.93	-32.00	0.01156
ANFIS training	0.06454	887.2	-15.62	0.03893
ANFIS test	0.00278	39.16	-29.68	0.01237
MoG training	0.05567	763.1	-16.33	0.03454
MoG test	0.00105	14.49	-33.85	0.01117

Table 11.2. Prediction Results Using the 30 Days Training Set

11.3.2 Second set of trials

In this section, the proposed HONFIS model is added to the previously approach, and the surrounding environment is also complicated.

Precisely, all of the computational models are trained using time series subsampled at an interval of 1 hour, thus resulting in 17 samples per day. Four different kinds of training conditions are considered: 1-day, 3-days, 7-days and 30-days, associated with training sets composed of 17, 51, 119 and 510 samples, respectively. Every computational model estimated by using one of these training sets, is applied to test the day following the latest one in the training set, which is on a test set of 17 samples. The successive experiments will consider for testing one day for each season of 2012 and the 15th of each month has been chosen for the sake of uniformity[3]. After a preliminary analysis it has been carried out that the embedding parameters do not show a considerable sensitivity to seasonality and to the length of the training set. The last one, is relatively small with respect to the usual length of time series processed by the AMI and FNN methods, hence the values $T = 5$ and $D = 3$ are used for every training set as optimal choice. It can be underlined that all of the test sets have different irradiation and meteorological conditions and thus, they can be considered as a good ensemble for representing the typical behaviors that might be encountered.

In addition to the three proposed neural and fuzzy neural models (RBF, ANFIS, HONFIS), the linear (LSE) and ARIMA predictors are adopted for benchmarking. LSE does not have parameters to be set in advance, while for ARIMA, RBF and ANFIS the default options provided by the software platform for training and model regularization have been adopted. [4] Regarding HONFIS, a preliminary analysis was performed in order to estimate the best model order for the rule consequents. The 3rd order was found as the best model in the almost of the considered case, thus only its values will be considered in the following results. The input space classification is obtained by a 3-NN classifier (i.e., $K = 3$). The optimal number of rules is determined by eq.(11.24), using $\lambda = 0.5$ and varying M from 1 up to 50% of N_T. The prediction are measured only in terms of two of the previously described metrics: the NMSE eq.(11.26) and the MARE eq.(11.28).

The numerical results for each tested day are reported in Tables 11.3–11.6, where the performance of the best model is marked in bold font. As per the following discussion, being HONFIS the model that assures the best performance in most of the situations, the graphical illustration of the actual time series (blue line) and the one predicted by HONFIS (red line), over the four training conditions, is reported in Figs. 11.5–11.8, respectively. In the x-axis there is reported the cumulative index of samples of the considered day, starting from index 1 for the first sample of March 1st, 2012 and considering 17 samples per day. The output current reported in the y-axis is a dimensionless value between 0 and 1, as the whole dataset has been normalized before the model processing. Some negative values may occur because of possible numerical issues of a trained model when its performance is inadequate.

[3] For further analysis, in [196] the analysis is extend to over the year, where the experiments will consider for testing one day for each month of 2012.

[4] ARIMA, RBF, and ANFIS models are trained by using the supported functions in the Econometrics, Neural Network, and Fuzzy Logic toolbox of Matlab, respectively.

Prediction model	NMSE				MARE			
	1-day	3-days	7-days	30-days	1-day	3-days	7-days	30-days
LSE training	0.0399	0.0428	0.1248	0.2093	0.0663	0.0682	0.0946	0.1061
LSE test	0.0393	0.0388	0.0412	0.0463	0.0657	0.0651	0.0673	0.0720
ARIMA training	0.6048	0.3498	2.3712	0.4085	0.2417	0.1752	0.0692	0.0688
ARIMA test	0.0091	0.0190	0.6537	0.5065	0.0292	0.0437	0.0426	0.0687
RBF training	0.0041	0.0002	0.0323	0.1225	0.0168	0.0037	0.0476	0.0715
RBF test	0.0063	0.0002	0.0153	0.0128	0.1508	0.0040	0.0355	0.0310
ANFIS training	0.0001	0.0005	0.0134	0.1268	0.0002	0.0011	0.0269	0.0720
ANFIS test	**0.0021**	0.0003	0.0086	0.0216	**0.0120**	**0.0010**	0.0208	0.0399
HONFIS (3rd ord.) training	0.0001	0.0001	0.0039	0.0012	0.0002	0.0012	0.0090	0.0035
HONFIS (3rd ord.) test	0.9437	**0.0002**	**0.0062**	**0.0003**	0.1985	0.0013	**0.0129**	**0.0035**

Table 11.3. Prediction Results for March, 15th

Prediction model	NMSE				MARE			
	1-day	3-days	7-days	30-days	1-day	3-days	7-days	30-days
LSE training	0.0108	0.0121	0.0096	0.0957	0.0363	0.0383	0.0318	0.0579
LSE test	0.0119	0.0118	0.0118	0.0137	0.0376	0.0375	0.0372	0.0361
ARIMA training	0.7354	1.3923	0.3083	0.3240	0.2934	0.3922	0.1314	0.1667
ARIMA test	0.8885	0.0115	0.0670	0.0205	0.3397	0.0327	0.0831	0.0473
RBF training	0.0003	0.0002	0.0002	0.0367	0.0054	0.0049	0.0046	0.0326
RBF test	**0.0005**	**0.0002**	**0.0002**	0.0075	**0.0073**	0.0049	0.0048	0.0266
ANFIS training	0.0001	0.0005	0.0003	0.0423	0.0005	0.0017	0.0053	0.0353
ANFIS test	0.0010	0.0006	0.0004	0.0118	0.0099	0.0019	0.0064	0.0303
HONFIS (3rd ord.) training	0.0001	0.0006	0.0001	0.0053	0.0001	0.0010	0.0022	0.0068
HONFIS (3rd ord.) test	0.1763	0.0004	0.0006	**0.0004**	0.0482	**0.0008**	**0.0012**	**0.0045**

Table 11.4. Prediction Results for June, 15th

Prediction model	NMSE				MARE			
	1-day	3-days	7-days	30-days	1-day	3-days	7-days	30-days
LSE training	0.7511	0.3141	0.2900	0.2971	0.1845	0.1280	0.1281	0.1192
LSE test	0.7526	0.7263	0.7265	0.7262	0.2181	0.2088	0.2099	0.2114
ARIMA training	1.3369	1.3875	0.9725	0.3571	0.4424	0.2483	0.2423	0.1741
ARIMA test	**0.4237**	0.6639	0.5273	0.3427	0.3354	0.2334	0.1888	0.1604
RBF training	0.2025	0.1243	0.1076	0.1840	0.1016	0.0855	0.0708	0.0883
RBF test	1.2412	0.2118	0.3201	0.3445	0.3014	0.1155	0.1379	0.1362
ANFIS training	0.0001	0.0001	0.0498	0.1953	0.0001	0.0001	0.0467	0.0903
ANFIS test	0.4766	**0.0001**	0.0633	0.4668	**0.1738**	**0.0008**	0.0529	0.1582
HONFIS (3rd ord.) training	0.0001	0.0001	0.0122	0.0032	0.0001	0.0010	0.0138	0.0050
HONFIS (3rd ord.) test	1.7032	0.0002	**0.0068**	**0.0092**	0.3168	0.0011	**0.0136**	**0.0125**

Table 11.5. Prediction Results for September, 15th

Prediction model	NMSE				MARE			
	1-day	3-days	7-days	30-days	1-day	3-days	7-days	30-days
LSE training	0.6335	0.5107	0.5513	0.3814	0.1109	0.1110	0.0970	0.0984
LSE test	**0.4326**	0.3922	0.4051	0.4512	**0.1436**	0.1552	0.1526	0.1518
ARIMA training	1.0407	1.1536	1.9333	1.1766	0.3000	0.2067	0.2194	0.2016
ARIMA test	0.9998	0.8886	0.5563	0.4208	0.2272	0.2090	0.1557	0.1769
RBF training	0.0754	0.0379	0.0540	0.2106	0.0412	0.0481	0.0383	0.0665
RBF test	0.9442	0.0793	0.1061	0.1270	0.2207	0.0768	0.0627	0.0684
ANFIS training	0.0609	0.0043	0.0556	0.2272	0.0401	0.0064	0.0410	0.0719
ANFIS test	1.2890	**0.0057**	0.1005	0.1443	0.2599	**0.0074**	0.0639	0.0643
HONFIS (3rd ord.) training	0.0060	0.0039	0.0198	0.1012	0.0083	0.0069	0.0094	0.0316
HONFIS (3rd ord.) test	1.1232	0.0079	**0.0544**	**0.1017**	0.1951	0.0083	**0.0298**	**0.0416**

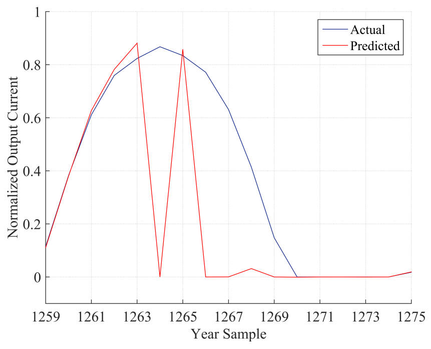
Table 11.6. Prediction Results for December, 15th

Considering the experimental results reported in this work, it can be underlined that all of the proposed approaches are suitable for the prediction of the considered time series, depending on the chosen training set. The numerical results, in terms of either NMSE or MARE, basically agree and are coherent with the graphical behaviors between actual and predicted time series. By the way, a perfect match is obtained when the NMSE is smaller than about $5 \cdot 10^{-4}$.

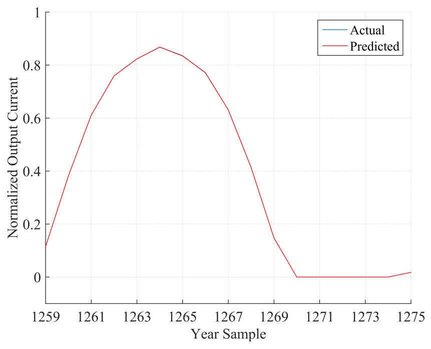
In the days with the most stable meteorological conditions (i.e., March 15th or June 15th), the prediction results are better than the others. Also, there are differences among the results associated with different training procedures. For each algorithm, the prediction with 1-day training data is highly affected by the difference in irradiation between the day of the training and the one of test. In fact, when the meteorological condition varies a lot from the training day to the sequent (test day), the algorithms are trained on a set that is almost not correlated with the test one. It can also be noted that when the irradiation is very similar (no clouds, good weather), the performance is very good also for the 1-day training set.

A similar discussion can be made, with smaller variability, for the 3-days tests. It has to be noted that the 1-day and 3-day tests have been inserted mainly to show the sensitivity of the models. The 7-days tests are much more stable for all of the algorithms and the days. The 30-days tests are stable too, but the sequence is much longer. Hence, if the days are variable in terms of meteorological conditions, the model has a worse performance because the training is more difficult. Overall, a 1-day training is based on a very small number of samples in the training set. This is an interesting issue for the computational cost of the learning procedure when a lot of model parameters must be estimated, which is the well-known curse of dimensionality for neural networks. In fact, the 3rd order polynomial adopted for HONFIS makes it the most complex model among the ones considered in this work, and its performances improve, with respect to the other models, as much as the the number of samples increases in the training set. Although, a larger training set may not be the optimal choice.

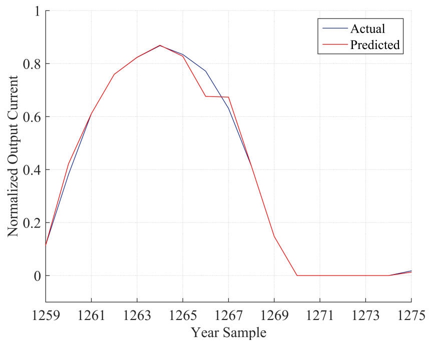
The performance of the prediction is highly affected by the intrinsic seasonality of the considered time series. Anyway, the HONFIS model achieves the best results



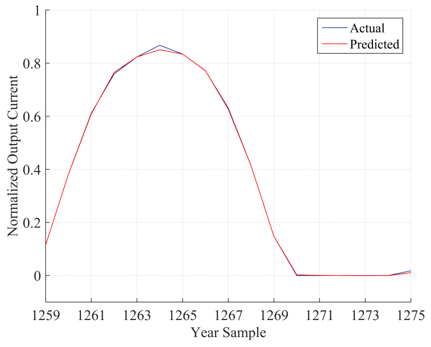
(a) 1-day training



(b) 3-days training



(c) 7-days training



(d) 30-days training

Figure 11.5. Prediction behavior using the best model (HONFIS) for March, 15th.

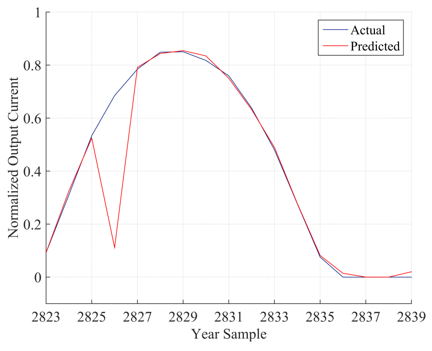
for almost all of the training sets with respect to the other proposed ones. Each of them outperform both the LSE benchmark and the ARIMA approach. The latter, is not a feasible solution for this purpose, due to the the intrinsic chaotic properties of the sequence. This reinforces the fact that the prediction of photovoltaic production is a promising field for the application of neural and fuzzy neural approaches, along with the use of a suitable embedding procedure.

11.3.3 Third set of trials

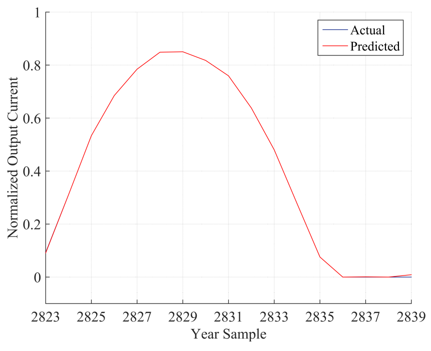
In this section, the focus is given only on the fuzzy neural networks, analyzing the performance of the HONFIS model when the order of the polynomial changes. In effect, in the previous paragraph the best HONFIS order was selected and inserted for comparisons. Now, it is detailed a procedure by which it could be find out.

All of the tests are done using either a 7-days or 30-days time series that stems from a subsampled version of the original sequence, with a sample interval of 1 hour. This results in a sequence with 17 samples per day and a training set of 119 and 510 samples, respectively. The performances of the resulting predictors are tested on the successive 17 samples of the sequence (the single day after the 7 or 30 days used for training).

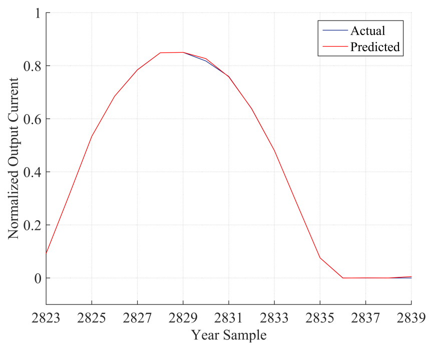
The procedure is run from $C = 1$ to $C = 200$; this value is considered as the maximum complexity allowed by the network. Ten different initializations are carried out for each value of C as well as for the other compared models that undergo a random initialization of their parameters. The average values over the 10 runs are



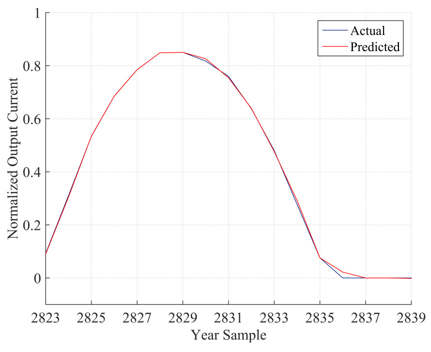
(a) 1-day training



(b) 3-days training

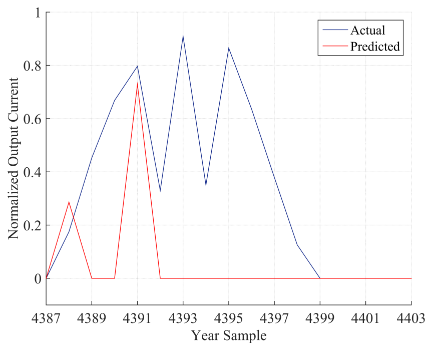


(c) 7-days training

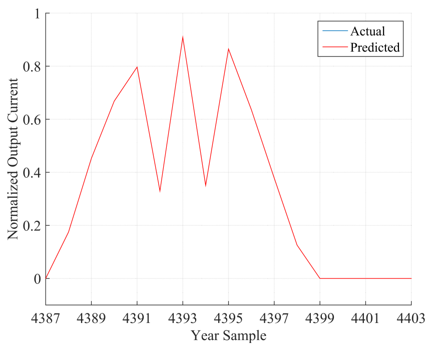


(d) 30-days training

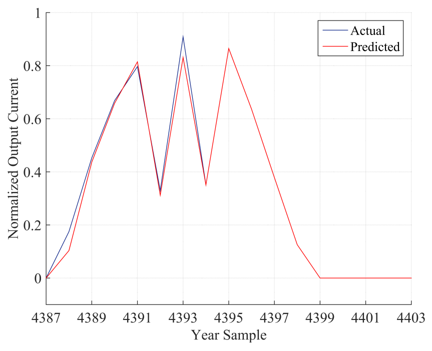
Figure 11.6. Prediction behavior using the best model (HONFIS) for June, 15th.



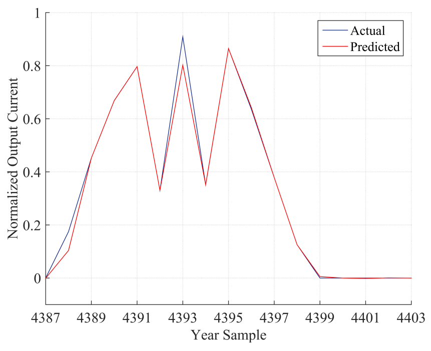
(a) 1-day training



(b) 3-days training

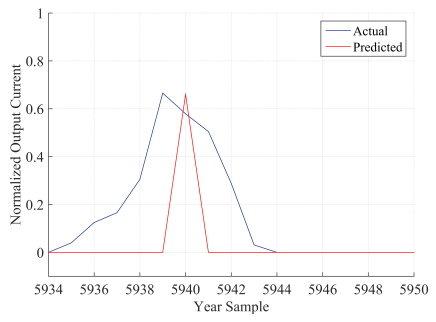


(c) 7-days training

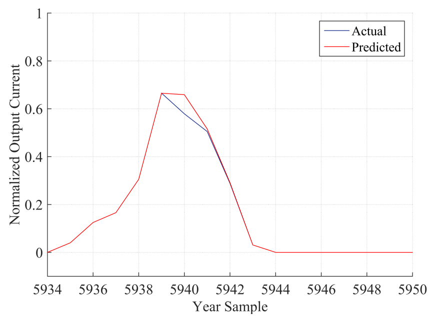


(d) 30-days training

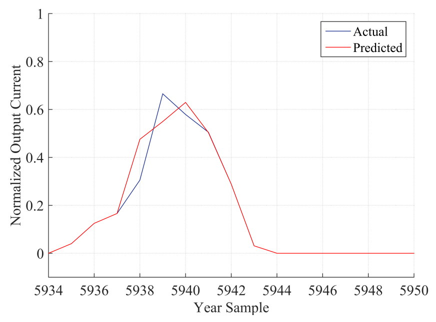
Figure 11.7. Prediction behavior using the best model (HONFIS) for September, 15th.



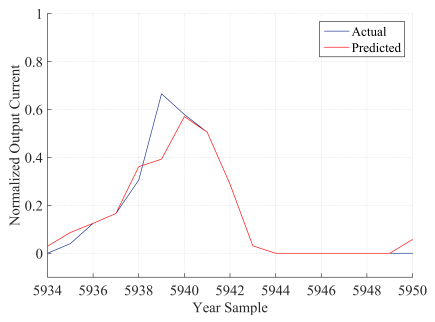
(a) 1-day training



(b) 3-days training



(c) 7-days training



(d) 30-days training

Figure 11.8. Prediction behavior using the best model (HONFIS) for December, 15th.

reported in the following. As aforementioned, for TS predictors the classification in the input space is performed by the K-NN procedure with $K = 3$. The training samples are also applied to determine both the values of D and T by using the methods suggested in [184]; for the sake of comparison $D = 3$ and $T = 5$ are used for all the considered models.

Let us consider a test set $\mathcal{T}$ of N_S pairs $(\mathbf{x}_t, y_t)$ successive to the ones used for training; the test set will be represented by the integers $t \in \mathcal{T}$, which are the time indexes of such pairs. The prediction performance will be measured in terms of NMSE eq.(11.26), MSE eq.11.25, NSR eq.11.27, MARE eq.(11.28).

First of all, the performance of the HONFIS approach are evaluated considering different orders for the polynomial function $h(\cdot)$ of the TS rule consequent. There are considered five different cases, that is from first-order to fifth-order functions. The results are reported in Table 11.7 in terms of NMSE only, as it is sufficient to determine the best order to be used for the next comparisons. Such results are relative to the test performed on the 30th of June as test set. This day has be chosen because of the stability of the sequence in terms of meteorological condition and irradiation. The best results are given by third-order and fourth-order functions for the 7-days and 30-days training set, respectively. Fig. 11.9 illustrates a comparison between the worst performance for 7-days training set, obtained by the fifth-order function, and the best performance one, as said, by the third-order function. Similarly, in Fig. 11.10 is shown a comparison between the worst performance for 30-days training set, obtained by the first-order function, and the best performance one obtained,

TS Rule	7-days	30-days
First-order training	4.043	1.636
First-order test	0.058	0.019
Second-order training	0.596	2.019
Second-order test	0.053	0.018
Third-order training	0.611	1.591
Third-order test	**0.052**	0.017
Fourth-order training	0.639	1.514
Fourth-order test	0.055	**0.016**
Fifth-order training	0.819	1.513
Fifth-order test	0.097	0.016

NMSE values are scaled by 10^{-2}

Table 11.7. Prediction Results (NMSE) for Different TS Orders of The Rule Consequent

in this case, by the fourth-order function. Then, the prediction performances of the proposed TS approach are compared with three prediction models already introduced: LSE, RbF (sec.11.2.3) and HONFIS (sec.11.2.4).

A cross-validation technique is adopted by using a suited early-stopping procedure in order to optimize the model complexity.

In Table 11.8 and Table 11.9 are illustrated the results by using the 7-days and 30-days time series, respectively. Evidently, the proposed TS approach is able to obtain the best performance in both the 7-days and 30-days prediction problems. It also important to remark that, even in the case of a poor choice of the polynomial order in the TS model, its performance still remains better than the ones of compared models. In Fig. 11.11 is reported a comparison between the worst performance

Predictor	NMSE	MSE	NSR_{dB}	MARE
LSE training	7.105	0.709	-15.175	0.056
LSE test	1.541	0.136	-22.287	0.038
RBF training	2.129	0.212	-20.408	0.023
RBF test	0.200	0.018	-31.159	0.014
ANFIS training	1.166	0.116	-23.023	0.027
ANFIS test	1.067	0.094	-23.882	0.028
TS (third-order) training	0.611	0.110	-25.780	0.011
TS (third-order) test	**0.052**	**0.094**	**-38.780**	**0.003**

NMSE and MSE values are scaled by 10^{-2}

Table 11.8. Comparison of Prediction Performances on The 7-Days Training Set

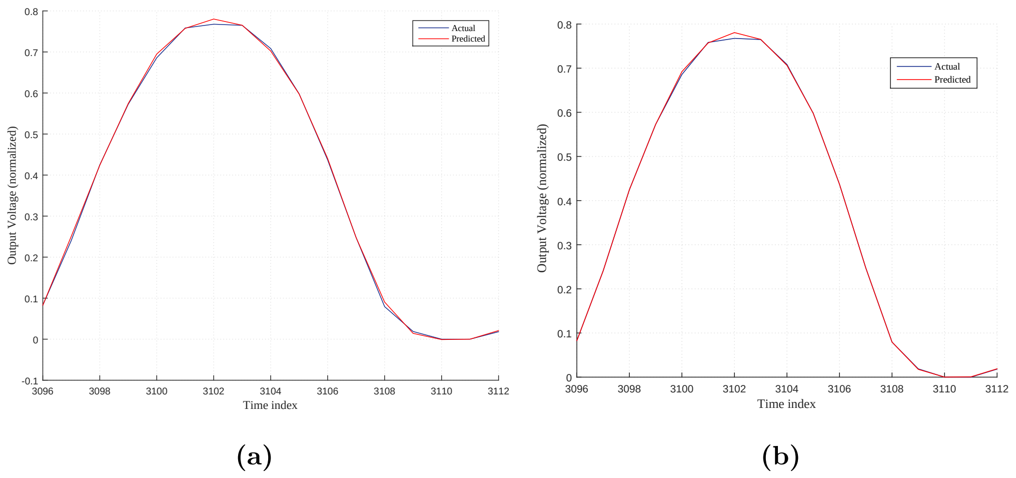


Figure 11.9. TS prediction on the 7-days training set: worst performance of fifth-order consequents (left); best performance of third-order functions (right).

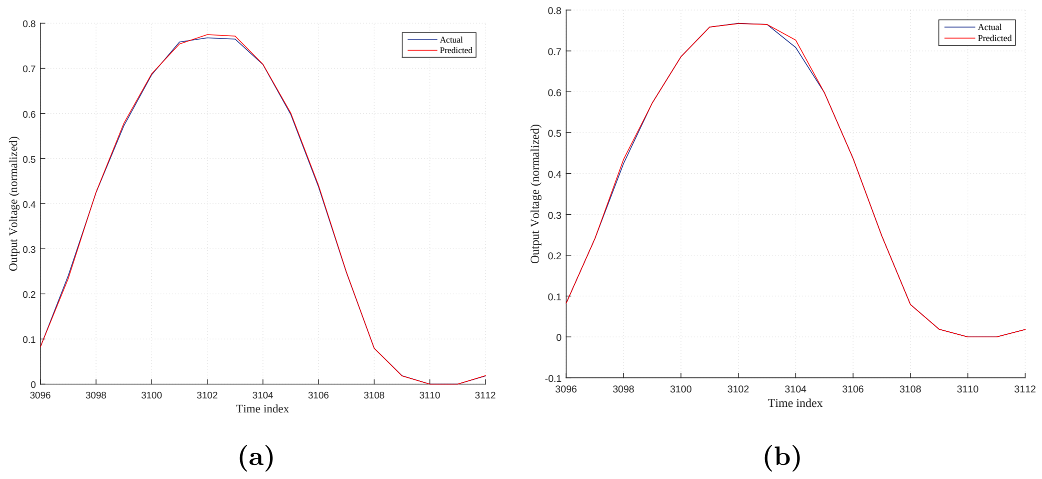


Figure 11.10. TS prediction on the 30-days training set: worst performance of first-order consequents (left); best performance of fourth-order functions (right).

Predictor	NMSE	MSE	NSR_{dB}	MARE
LSE training	14.99	1.498	-11.839	0.078
LSE test	2.162	0.191	-20.817	0.047
RBF training	7.060	0.706	-15.109	0.049
RBF test	1.601	0.141	-22.122	0.032
ANFIS training	9.401	0.940	-13.865	0.057
ANFIS test	1.633	0.144	-22.036	0.041
TS (fourth-order) training	1.514	0.304	-19.328	0.005
TS (fourth-order) test	**0.016**	**0.025**	**-80.251**	**0.004**

NMSE and MSE values are scaled by 10^{-2}

Table 11.9. Comparison of Prediction Performances on The 30-Days Training Set

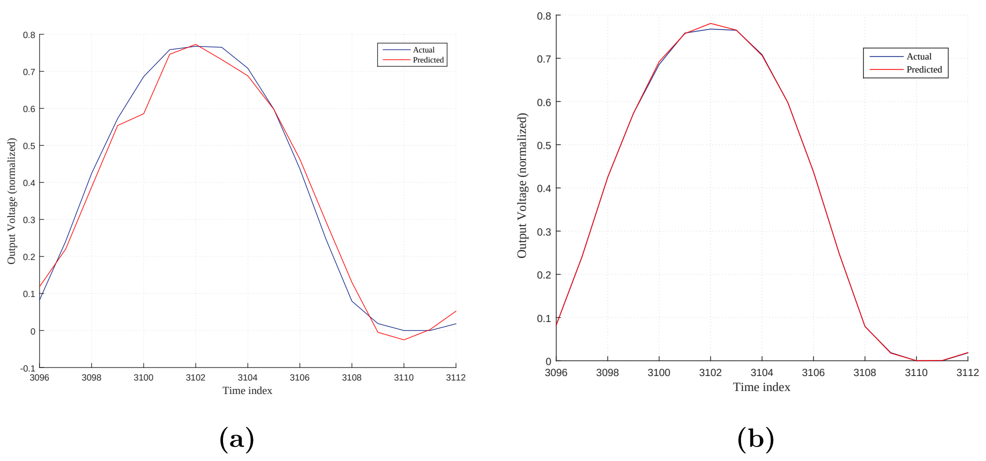


Figure 11.11. Prediction on the 7-days training set: worst performance of LSE predictor (left); best performance of TS model using third-order consequents (right).

for 7-days training set, obtained by the LSE predictor, and the best performance obtained by the TS model using third-order functions. Again, in Fig. 11.12 is illustrated the comparison between the worst performance for 30-days training set, obtained once again by the LSE predictor, and the best performance obtained by the TS model using in this case fourth-order consequents.

11.3.4 Observations

In this chapter, neural and fuzzy neural networks approaches are considered for time series prediction. In particular, an approach based on Takagi-Sugeno fuzzy inference systems using higher-order polynomials for the rules' consequent is introduced. A constructive procedure able to automatically determine the optimal number of fuzzy rules and avoid overfitting is also introduced. In this way, the generalization

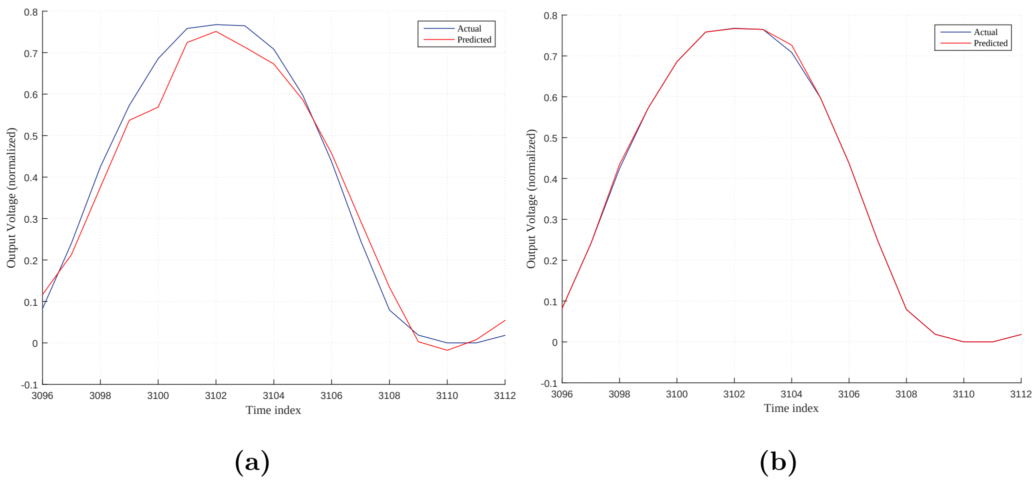


Figure 11.12. Prediction on the 30-days training set: worst performance of LSE predictor (left); best performance of TS model using fourth-order functions (right).

capability of the fuzzy neural network are maximized. The validation performed on historical data shows that the fuzzy approach generates values that are able to replicate accurately the data behavior.

Additionally, new embedding approaches based on neural and fuzzy networks have been properly tailored to be efficiently applied to PV time series prediction. They provide an accurate description of time series dynamics, allowing us to estimate future values over a long time horizon. Nevertheless, the proposed models outperforms in almost all cases well-known benchmarks.

The results are very promising, hence the proposed models could be applied also to different applications in electrical engineering, where data regression and time series prediction represent a key issue for obtaining a better management of economic and energetic resources. Detrending techniques, such as mean-reverting approaches, could be used to remove seasonal differences and spike outliers as well as to improve the training accuracy. Additionally, these proposals could be useful for the distributed learning approach, by which the results could be improved sharing the data from different cabins of the same plant or from different nearby plants.

Chapter 12

Conclusions

This thesis focuses on distributed machine learning problems, wherein data to be analyzed is partitioned into a set of interconnected agents with limited connectivity. It has been pointed out that separate domains of application may impose very different constraints in respect of the traditional centralized solution. Among these, low computational power at every location, limited connectivity, or transferability constraints related to privacy and security reasons, are just few examples that could be mentioned. With this in mind, this thesis looked at the possibility of performing inference on data in a decentralized fashion (i.e., without a centralized authority or coordinator), leaving out the exchange of training data. Previously works have failed to address this issue, since they are often just a parallel version of some centralized techniques. Accordingly, they are not actually distributed since they just split the computation and successively send the results to a central node that makes a decision. In other applications, bridge sensors or a loop through all the networks' node is necessary to reach out a common agreement on final results.

In this context, several innovative solutions have been proposed to overcome the main limits of the existing approaches. The challenge was to avoid that a central node detains all of the information, while reaching a solution that is similar to the centralized one, only through a process of communication and collaboration among the agents. To achieve this, throughout the present thesis, different approaches working on a fully distributed scenario have been presented.

Firstly, an extension of the well-known centralized clustering ensemble technique has been detailed. It is largely adopted in the centralized case because of its capability to improve the quality of individual data clustering. At the same time, its use on the distributed learning setting has not been considered so far; this is the main motivation that has brought us to introduce the V-DEC procedure. The novelty of this approach is the capacity of working in a fully distributed scenario, with a network of interconnected agents that acquire data locally and autonomously. They are allowed to start with a different partition constructed by considering its own data only as well as a different number of initial clusters. The agreement among the partitions is reached by an automatic and mutual refinement of the results until the different partitions become statistically similar with a good internal validity index. The idea behind this approach is very simple, yet it allows us to reach a solution similar to the centralized one, by only exchanging the local representatives onto the network and increasing the efficiency and the speed of the whole procedure.

The V-DEC algorithm, as most of the clustering methods, relies on suitable metrics encoded in a matrix of pairwise distances between patterns. Thus, it is proposed a new solution for building in a totally decentralized fashion the matrix of Euclidean distances EDM among all points. Recasting the problem in this way allow us to leverage over a large number of works on matrix completion and EDM completion, especially in the distributed setting. Moreover, although the attention is focused on the spectral clustering algorithm, nothing prevents the framework from being used with different other approaches based on a similarity measure. To the best of our knowledge, this is the first distributed protocol for its solution and experimental results show that such approach is efficiently able to match a fully centralized implementation for reconstructing a matrix of similarity measures, without requiring the presence of a coordinating node.

Returning to the constraints posed at the beginning of the study, it is possible to claim that in many distributed applications, it is not realistic to assume that the entire training set is available before to begin the learning procedure, but data arrives one-by-one or chunk-by-chunk. In this regard, it has been proposed an on-line learning algorithm under the hypothesis of training data distributed across a network of interconnected agents. In particular, each agent in the network is assumed to receive a stream of data as a sequence of mini-batches. When receiving a new chunk of data, each agent updates its estimate of the model parameters and, periodically, all agents agree on a common model through a distributed average consensus DAC protocol. This approach has been applied to the learning of fuzzy neural network architectures and the learning algorithm is faster than a solution based on a centralized training set. The experimental results on well-known datasets validate the proposal. Nonetheless, the same problem is analyzed from a different point of view, where a non-convex distributed optimization in multi-agent networks with time-varying connectivity is applied to the well-known expectation-maximization EM approach. It exploits successive convex approximation techniques while leveraging dynamic consensus as a mechanism to distribute the computation as well as propagate the needed information over the network. Numerical results show that the new method compares favorably to the existing approaches and to the centralized one.

Successively, several possible applications of the distributed learning are introduced and discussed. The first one concerns the framework of telemedicine, wherein data is distributed across multiple clinical parties. The final aim was to put the basis for a new multimedia ICT platform able to deliver healthcare services accessible by the user from his home. The collaboration with the Biomechanics Laboratory of the Policlinico Umberto I of Rome and the Rehabilitation Medical Center of the 2nd Hospital of Jiaxing has helped us in pointing out the main problems that can arise in a scenario like this. The capability of two low-cost devices, Microsoft Kinect and inertial IMU sensors, are firstly tested and assessed in acquiring clinical relevant information. It has been found that they represent an innovative alternative to the standard systems, thanks to their ability to track the movement of a person avoiding need of markers or controllers. Once obtained a method for acquiring data in a home context, the successive step was involved in the analysis of data acquired at the single locations. To this end, a decision making system for gait analysis has been proposed. It relies on the use of machine learning and ad-hoc algorithms able to extract the relevant information useful for the clinical analysis only. In effect, some extracted features can serve as potential biomarkers in the definition of a disease, while extra features could increase the complexity of the learning process. The experimental results show that the proposed methods are able to reduce the dimensionality of the data space and to determine the patient's status with a suitable classification accuracy higher than 97%. However, in a medical scenario, these methods could not be directly applied since data could not be transferred from one site to another for privacy concerns over sensible portions of the dataset. In this regard, the aforementioned distributed spectral clustering procedure has been extended, by adding suitable protocols for privacy preserving.

The second application context is related to sensor networks. Typically, a wireless sensor network is composed of a large number of low-cost, low-power and heterogeneous devices, which are used to learn from the environment that they are sensing. However, to upload learning algorithms on a digital architecture with a finite precision arithmetic, after a direct quantization of coefficients, leads to unsatisfactory results due to the non-linear nature of the network. To this end, some modified learning algorithms have been proposed for neural networks. Specifically, they relies on random vector functional-link RVFL models, for processing distributed data. The novelty is in the quantization procedure able to make them suited to hardware architectures even based on a simple microcontroller. Both uniform and non-uniform quantization has been introduced in order to optimize the neural network implementation. It has been shown that the proposed approaches are effective even using a small number of bits, which compensates and regularizes the effects of rounding in the successive layers of the neural network.

Finally, the last application of distributed learning that has been considered is the prediction of electric power produced by renewable energy plants. This is an interesting area of application, being both wind-electric conversion and photovoltaic systems in advances states of developments. New embedding approaches based on neural and fuzzy networks have been properly tailored to be efficiently applied to photovoltaic PV time series prediction. Additionally, a Takagi-Sugeno fuzzy inference system using higher-order polynomials for the rules' consequent has been introduced. In Italy, the proposed models outperform in almost all cases well-known benchmarks and the preliminary results concerning the prediction of power generated by a large scale photovoltaic plant are very promising.

In conclusion, this thesis has proposed an extensive research focused on the design and the implementation of useful and practical distributed data mining systems. On one hand, the work focuses on the realization of new distributed algorithms; on the other hand, it also presents applications on large scale real-world problems. The present findings, supported by encouraging experimental results, highlight the importance to set new goals and develop further scientific research for solving a new class of machine learning problems in a fully distributed scenario.

Bibliography

[1] Vladimir Vapnik. *The nature of statistical learning theory.* Springer science & business media, 2013.

[2] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[3] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory.* Psychology Press, 2005.

[4] Tom M Mitchell et al. Machine learning. wcb, 1997.

[5] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes-which naive bayes? In *CEAS*, volume 17, pages 28–69, 2006.

[6] George H John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc., 1995.

[7] Rick L Lawrence and Andrea Wright. Rule-based classification systems using classification and regression tree (cart) analysis. *Photogrammetric engineering and remote sensing*, 67(10):1137–1142, 2001.

[8] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and Klaus-Robert Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop.*, pages 41–48. IEEE, 1999.

[9] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[10] George AF Seber and Alan J Lee. *Linear regression analysis*, volume 936. John Wiley & Sons, 2012.

[11] Douglas M Bates and Donald G Watts. *Nonlinear regression analysis and its applications*, volume 2. Wiley Online Library, 1988.

[12] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[13] Pavel Berkhin et al. A survey of clustering data mining techniques. *Grouping multidimensional data*, 25:71, 2006.

[14] Jeffrey D Banfield and Adrian E Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, pages 803–821, 1993.

[15] Juha Vesanto and Esa Alhoniemi. Clustering of the self-organizing map. *IEEE Transactions on neural networks*, 11(3):586–600, 2000.

[16] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[17] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[18] Peter Hall, Jeff Racine, and Qi Li. Cross-validation and the estimation of conditional probability densities. *Journal of the American Statistical Association*, 99(468):1015–1026, 2004.

[19] Antonello Rosato, Rosa Altilio, and Massimo Panella. Recent advances on distributed unsupervised learning. In *Advances in Neural Networks*, pages 77–86. Springer, 2016.

[20] Clark F Olson. Parallel algorithms for hierarchical clustering. *Parallel computing*, 21(8):1313–1325, 1995.

[21] Xiaobo Li and Zhixi Fang. Parallel clustering algorithms. *Parallel Computing*, 11(3):275–290, 1989.

[22] Ying Zhao, George Karypis, and Usama Fayyad. Hierarchical clustering algorithms for document datasets. *Data mining and knowledge discovery*, 10(2):141–168, 2005.

[23] Ping Ding, JoAnne Holliday, and Aslihan Celik. Distributed energy-efficient hierarchical clustering for wireless sensor networks. *Distributed computing in sensor systems*, pages 466–467, 2005.

[24] Sanpawat Kantabutra and Alva L Couch. Parallel k-means clustering algorithm on nows. *NECTEC Technical journal*, 1(6):243–247, 2000.

[25] M Zhen and G Ji. Dk-means, an improved distributed clustering algorithm. *J. Comput. Res. Dev*, 44(2):84–88, 2007.

[26] Xinghao Pan, Joseph E Gonzalez, Stefanie Jegelka, Tamara Broderick, and Michael I Jordan. Optimistic concurrency control for distributed unsupervised learning. In *Advances in Neural Information Processing Systems*, pages 1403–1411, 2013.

[27] Maria-Florina F Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general topologies. In *Advances in Neural Information Processing Systems*, pages 1995–2003, 2013.

[28] Yingyu Liang, Maria-Florina Balcan, and Vandana Kanchanapally. Distributed pca and k-means clustering. In *The Big Learning Workshop at NIPS*, volume 2013, 2013.

[29] Dimitris K Tasoulis and Michael N Vrahatis. Unsupervised distributed clustering. In *Parallel and distributed computing and networks*, pages 347–351, 2004.

[30] Xiaowei Xu, Jochen Jäger, and Hans-Peter Kriegel. A fast parallel clustering algorithm for large spatial databases. *High Performance Data Mining*, pages 263–290, 2002.

[31] Eshref Januzaj, Hans-Peter Kriegel, and Martin Pfeifle. Towards effective and efficient distributed clustering. In *Workshop on Clustering Large Data Sets (ICDM2003)*, 2003.

[32] W Ni, G Chen, Y Wu, and Z Sun. Local density based distributed clustering algorithm. *Journal of Software*, 19(9):2339–2348, 2008.

[33] Matthias Klusch, Stefano Lodi, and Gianluca Moro. Distributed clustering based on sampling local density estimates. In *IJCAI*, pages 485–490, 2003.

[34] Nhien-An Le-Khac, Lamine M Aouad, and M-Tahar Kechadi. A new approach for distributed density based clustering on grid platform. In *British National Conference on Databases*, pages 247–258. Springer, 2007.

[35] Robert D Nowak. Distributed em algorithms for density estimation and clustering in sensor networks. *IEEE transactions on signal processing*, 51(8):2245–2253, 2003.

[36] Nam Nguyen and Rich Caruana. Consensus clusterings. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 607–612. IEEE, 2007.

[37] Pedro A Forero, Alfonso Cano, and Georgios B Giannakis. Consensus-based distributed expectation-maximization algorithm for density estimation and classification using wireless sensor networks. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 1989–1992. IEEE, 2008.

[38] Roberto D Pascual-Marqui, Alberto D Pascual-Montano, Kieko Kochi, and José María Carazo. Smoothly distributed fuzzy c-means: a new self-organizing map. *Pattern recognition*, 34(12):2395–2402, 2001.

[39] T Srinivasan, Vivek Vijaykumar, and R Chandrasekar. A self-organized agent-based architecture for power-aware intrusion detection in wireless adhoc networks. In *Computing & Informatics, 2006. ICOCI'06. International Conference on*, pages 1–6. IEEE, 2006.

[40] Dusan Jakovetic, Joao Xavier, and Jose MF Moura. Fast distributed gradient methods. *IEEE Transactions on Automatic Control*, 59(5):1131–1146, 2014.

[41] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[42] S Sundhar Ram, A Nedić, and Venugopal V Veeravalli. A new class of distributed optimization algorithms: Application to regression of distributed data. *Optimization Methods and Software*, 27(1):71–88, 2012.

[43] John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 57(3):592–606, 2012.

[44] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[45] Pascal Bianchi and Jérémie Jakubowicz. Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization. 58(2):391–405, 2013.

[46] Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[47] Sergio Barbarossa, Stefania Sardellitti, Paolo Di Lorenzo, R Chellappa, and S Theodoridis. Distributed detection and estimation in wireless. In *Academic Press Library in Signal Processing: Communications and Radar Signal Processing*, volume 2. Elsevier, 2013.

[48] Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

[49] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, 2007.

[50] Shaohong Zhang, Hau-San Wong, and Ying Shen. Generalized adjusted rand indices for cluster ensembles. *Pattern Recognition*, 45(6):2214–2226, 2012.

[51] Prodip Hore, Lawrence O Hall, and Dmitry B Goldgof. A scalable framework for cluster ensembles. *Pattern recognition*, 42(5):676–688, 2009.

[52] Germain Forestier, Pierre Gançarski, and Cédric Wemmert. Collaborative clustering with background knowledge. *Data & Knowledge Engineering*, 69(2):211–228, 2010.

[53] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[54] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[55] FM Frattale Mascioli, G Risi, Antonello Rizzi, and Giuseppe Martinelli. A nonexclusive classification system based on co-operative fuzzy clustering. In *Signal Processing Conference (EUSIPCO 1998), 9th European*, pages 1–4. IEEE, 1998.

[56] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of human genetics*, 7(2):179–188, 1936.

[57] M Forina et al. An extendible package for data exploration, classification and correlation. *Institute of Pharmaceutical and Food Analisys and Technologies, Via Brigata Salerno*, 16147, 1991.

[58] Vincent G Sigillito, Simon P Wing, Larrie V Hutton, and Kile B Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10(3):262–266, 1989.

[59] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[60] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2):107–145, 2001.

[61] T.O. Kvålseth. A coefficient of agreement for nominal scales: An asymmetric version of kappa. *Educational and Psychological Measurement*, 51(1):95–101, 1991. cited By 7.

[62] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y Chang. Parallel spectral clustering in distributed systems. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):568–586, 2011.

[63] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-J en Lin, and Edward Y Chang. Large-scale spectral clustering with mapreduce and mpl. *Scaling up Machine Learning: Parallel and Distributed Approaches*, page 240, 2011.

[64] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[65] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009.

[66] E. J. Candes and Y. Plan. Matrix completion with noise. 98(6):925–936, 2010.

[67] B. Mishra, G. Meyer, and R. Sepulchre. Low-rank optimization for distance matrix completion. In *2011 50th IEEE conference on Decision and control and European control conference (CDC-ECC'11)*, pages 4455–4460. IEEE, 2011.

[68] Q. Ling, Y. Xu, W. Yin, and Z. Wen. Decentralized low-rank matrix completion. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'12)*, pages 2925–2928. IEEE, 2012.

[69] R. Fierimonte, S. Scardapane, A. Uncini, and M. Panella. Fully decentralized semi-supervised learning via privacy-preserving matrix completion. *IEEE Transactions on Neural Networks and Learning Systems*, 2016. accepted with minor revision.

[70] John N Tsitsiklis, Dimitri P Bertsekas, Michael Athans, et al. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. 31(9):803–812, 1986.

[71] Jianshu Chen and Ali H Sayed. Diffusion adaptation strategies for distributed optimization and learning over networks. 60(8):4289–4305, 2012.

[72] Ali H. Sayed. Adaptive networks. 102(4):460–497, 2014.

[73] Francis R Bach and Michael I Jordan. Learning spectral clustering, with application to speech separation. *Journal of Machine Learning Research*, 7(Oct):1963–2001, 2006.

[74] Jane K Cullum and Ralph A Willoughby. *Lanczos algorithms for large symmetric eigenvalue computations: Vol. I: Theory.* SIAM, 2002.

[75] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

[76] J. B. Predd, S. R. Kulkarni, and H. V. Poor. Distributed learning in wireless sensor networks. 23(4):56–69, 2006.

[77] Ali H Sayed. Adaptation, learning, and optimization over networks. *Foundations and Trends in Machine Learning*, 7(4-5):311–801, 2014.

[78] Simone Scardapane, Dianhui Wang, Massimo Panella, and Aurelio Uncini. Distributed learning for Random Vector Functional-Link networks. *Information Sciences*, 301:271–284, 2015.

[79] Abdo Y Alfakih, Amir Khandani, and Henry Wolkowicz. Solving euclidean distance matrix completion problems via semidefinite programming. *Computational optimization and applications*, 12(1-3):13–30, 1999.

[80] K. Liu, H. Kargupta, and J. Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. 18(1):92–106, 2006.

[81] M. Lichman. UCI machine learning repository, 2013.

[82] Athanasios Tsanas, Max A Little, Cynthia Fox, and Lorraine O Ramig. Objective automatic assessment of rehabilitative speech treatment in parkinson's disease. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(1):181–190, 2014.

[83] Marko Bohanec and Vladislav Rajkovic. Knowledge acquisition and explanation for multi-attribute decision making. In *8th Intl Workshop on Expert Systems and their Applications*, pages 59–78, 1988.

[84] M. Newman. *Networks: an introduction.* Oxford University Press, 2010.

[85] Jacob Cohen. A coefficient of agreement for nominal scales, Educational and Psychological Measurement. *Educational and Psychological Measurement*, 1960.

[86] J.-S.R. Jang. ANFIS: Adaptive-network-based fuzzy inference system. 23:665–685, 1993.

[87] James C Bezdek. *Pattern recognition with fuzzy objective function algorithms.* Springer Science & Business Media, 2013.

[88] Lynn Yaling Cai and Hon Keung Kwan. Fuzzy classifications using fuzzy inference networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28(3):334–347, 1998.

[89] J.-S.R. Jang, C.T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing: a Computational Approach to Learning and Machine Intelligence.* Prentice Hall, Upper Saddle River, NJ, USA, 1997.

[90] R. Fierimonte, M. Barbato, A. Rosato, and M. Panella. Distributed learning of random weights fuzzy neural networks. *Proc. of IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE 2016)*, 2016.

[91] Stephen L Chiu. Selecting input variables for fuzzy models. *J. of Intelligent & Fuzzy Systems*, 4(4):243–256, 1996.

[92] Plamen P Angelov and Dimitar P Filev. An approach to online identification of takagi-sugeno fuzzy models. *IEEE Trans. on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):484–498, 2004.

[93] Plamen Angelov and Dimitar Filev. Simpl_ets: a simplified method for learning evolving takagi-sugeno fuzzy models. In *Fuzzy Systems, 2005. FUZZ'05. The 14th IEEE Int. Conf. on*, pages 1068–1073. IEEE, 2005.

[94] Nikola K Kasabov and Qun Song. Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE trans. on Fuzzy Systems*, 10(2):144–154, 2002.

[95] Hai-Jun Rong, N Sundararajan, Guang-Bin Huang, and P Saratchandran. Sequential adaptive fuzzy inference system (safis) for nonlinear system identification and prediction. *Fuzzy sets and systems*, 157(9):1260–1275, 2006.

[96] Raffaele Parisi, Elio D Di Claudio, Gianni Orlandi, and Bhaskar D Rao. A generalized learning paradigm exploiting the structure of feedforward neural networks. *IEEE Trans. on Neural networks*, 7(6):1450–1460, 1996.

[97] Hai-Jun Rong, Guang-Bin Huang, Narasimhan Sundararajan, and Paramasivan Saratchandran. Online sequential fuzzy extreme learning machine for function approximation and classification problems. *IEEE Trans. on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(4):1067–1072, 2009.

[98] I-C Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998.

[99] I-Cheng Yeh, King-Jang Yang, and Tao-Ming Ting. Knowledge discovery on rfm model using bernoulli sequence. *Expert Systems with Applications*, 36(3):5866–5871, 2009.

[100] Pınar Tüfekci. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *Int. J. of Electrical Power & Energy Systems*, 60:126–140, 2014.

[101] Heysem Kaya, Pmar Tüfekci, and Fikret S Gürgen. Local and global learning methods for predicting power of a combined gas & steam turbine. In *Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering*, pages 13–18, 2012.

[102] Aram Galstyan, Bhaskar Krishnamachari, Kristina Lerman, and Sundeep Pattem. Distributed online localization in sensor networks using a moving target. In *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*, pages 61–70. IEEE, 2004.

[103] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[104] Massimo Panella, Antonello Rizzi, and Giuseppe Martinelli. Refining accuracy of environmental data prediction by mog neural networks. *Neurocomputing*, 55(3):521–549, 2003.

[105] Paolo Di Lorenzo and Gesualdo Scutari. Next: In-network nonconvex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

[106] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[107] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[108] Minghui Zhu and Sonia Martínez. Discrete-time dynamic average consensus. *Automatica*, 46(2):322–329, 2010.

[109] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[110] Ying Sun, Gesualdo Scutari, and Daniel Palomar. Distributed nonconvex multiagent optimization over time-varying networks. In *Signals, Systems and Computers, 2016 50th Asilomar Conference on*, pages 788–794. IEEE, 2016.

[111] Dongbing Gu. Distributed em algorithm for gaussian mixtures in sensor networks. *IEEE Transactions on Neural Networks*, 19(7):1154–1166, 2008.

[112] Pedro A Forero, Alfonso Cano, and Georgios B Giannakis. Distributed clustering using wireless sensor networks. *IEEE Journal of Selected Topics in Signal Processing*, 5(4):707–724, 2011.

[113] J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[114] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

[115] Pawel Smialowski, Dmitrij Frishman, and Stefan Kramer. Pitfalls of supervised feature selection. *Bioinformatics*, 26(3):440–443, 02 2010.

[116] Nojun Kwak and Chong-Ho Choi. Input feature selection for classification problems. *IEEE Transactions on Neural Networks*, 13(1):143–159, 2002.

[117] Hussein Almuallim and Thomas G Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1-2):279–305, 1994.

[118] Byung Cheol Song, Myung Jun Kim, and Jong Beom Ra. A fast multiresolution feature matching algorithm for exhaustive search in large image databases. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(5):673–678, 2001.

[119] I. Iguyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[120] John Henry Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press, 1992.

[121] Andrea Proietti, Massimo Panella, Fabio Leccese, and Emiliano Svezia. Dust detection and analysis in museum environment based on pattern recognition. *Measurement*, 66:62–72, 2015.

[122] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[123] Tomohiro Shirakawa, Naruhisa Sugiyama, Hiroshi Sato, Kazuki Sakurai, and Eri Sato. Gait analysis and machine learning classification on healthy subjects in normal walking. *International Journal of Parallel, Emergent and Distributed Systems*, 32(2):185–194, 2017.

[124] Swarnalatha Purushotham and BK Tripathy. Evaluation of classifier models using stratified tenfold cross validation techniques. *Global Trends in Information Systems and Software Applications*, pages 680–690, 2012.

[125] Douglas G Altman and J Martin Bland. Diagnostic tests. 1: Sensitivity and specificity. *BMJ: British Medical Journal*, 308(6943):1552, 1994.

[126] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[127] Russell G Congalton. A review of assessing the accuracy of classifications of remotely sensed data. *Remote sensing of environment*, 37(1):35–46, 1991.

[128] P Mazzone, M Paoloni, M Mangone, V Santilli, A Insola, M Fini, and E Scarnati. Unilateral deep brain stimulation of the pedunculopontine tegmental nucleus in idiopathic parkinson's disease: effects on gait initiation and performance. *Gait & posture*, 40(3):357–362, 2014.

[129] Arlene Schmid, Pamela W Duncan, Stephanie Studenski, Sue Min Lai, Lorie Richards, Subashan Perera, and Samuel S Wu. Improvements in speed-based gait classifications are meaningful. *Stroke*, 38(7):2096–2100, 2007.

[130] Shawnna L Patterson, Larry W Forrester, Mary M Rodgers, Alice S Ryan, Frederick M Ivey, John D Sorkin, and Richard F Macko. Determinants of walking function after stroke: differences by deficit severity. *Archives of physical medicine and rehabilitation*, 88(1):115–119, 2007.

[131] Joseph A Zeni and Jill S Higginson. Differences in gait parameters between healthy subjects and persons with moderate and severe knee osteoarthritis: a result of altered walking speed? *Clinical Biomechanics*, 24(4):372–378, 2009.

[132] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[133] Jack W Smith, JE Everhart, WC Dickson, WC Knowler, and RS Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, page 261. American Medical Informatics Association, 1988.

[134] W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. 1992.

[135] Betul Erdogdu Sakar, M Erdem Isenkul, C Okan Sakar, Ahmet Sertbas, Fikret Gurgen, Sakir Delil, Hulya Apaydin, and Olcay Kursun. Collection and analysis of a parkinson speech dataset with multiple types of sound recordings. *IEEE Journal of Biomedical and Health Informatics*, 17(4):828–834, 2013.

[136] Chandramouli Krishnan, Edward P Washabaugh, and Yogesh Seetharaman. A low cost real-time motion tracking approach using webcam technology. *Journal of Biomechanics*, 48(3):544–548, 2014.

[137] Timothy Exell, Christopher Freeman, Katie Meadmore, Mustafa Kutlu, Eric Rogers, Ann-Marie Hughes, Emma Hallewell, and Jane Burridge. Goal orientated stroke rehabilitation utilising electrical stimulation, iterative learning and microsoft kinect. In *Rehabilitation Robotics (ICORR), 2013 IEEE International Conference on*, pages 1–6. IEEE, 2013.

[138] Ross A Clark, Adam L Bryant, Yonghao Pua, Paul McCrory, Kim Bennell, and Michael Hunt. Validity and reliability of the nintendo wii balance board for assessment of standing balance. *Gait & posture*, 31(3):307–310, 2010.

[139] Yong-Hao Pua, Zhiqi Liang, Peck-Hoon Ong, Adam L Bryant, Ngai-Nung Lo, and Ross A Clark. Associations of knee extensor strength and standing balance with physical function in knee osteoarthritis. *Arthritis care & research*, 63(12):1706–1714, 2011.

[140] João Couto Soares, Ágata Vieira, and Joaquim Gabriel. Development of a kinect rehabilitation system. *International Journal of Online Engineering*, 2013.

[141] Janani Venugopalan, Chih-Wen Cheng, and May D Wang. Motiontalk: personalized home rehabilitation system for assisting patients with impaired mobility. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 455–463. ACM, 2014.

[142] Rosa Altilio, Luca Liparulo, Massimo Panella, Andtea Proietti, and Marco Paoloni. Multimedia and gaming technologies for telerehabilitation of motor disabilities [leading edge]. *IEEE Technology and Society Magazine*, 34(4):23–30, 2015.

[143] Hossein Mousavi Hondori and Maryam Khademi. A review on technical and clinical impact of microsoft kinect on physical therapy and rehabilitation. *Journal of Medical Engineering*, 2014, 2014.

[144] Erik E Stone and Marjorie Skubic. Passive in-home measurement of stride-to-stride gait variability comparing vision and kinect sensing. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 6491–6494. IEEE, 2011.

[145] Ross A Clark, Yong-Hao Pua, Karine Fortin, Callan Ritchie, Kate E Webster, Linda Denehy, and Adam L Bryant. Validity of the microsoft kinect for assessment of postural control. *Gait & posture*, 36(3):372–377, 2012.

[146] Gregorij Kurillo, Alic Chen, Ruzena Bajcsy, and Jay J Han. Evaluation of upper extremity reachable workspace using kinect camera. *Technology and Health Care*, 21(6):641–656, 2013.

[147] Pratik Chattopadhyay, Aditi Roy, Shamik Sural, and Jayanta Mukhopadhyay. Pose depth volume extraction from rgb-d streams for frontal gait recognition. *Journal of Visual Communication and Image Representation*, 25(1):53–63, 2014.

[148] Michel Misiti, Yves Misiti, Georges Oppenheim, and Jean-Michel Poggi. Matlab wavelet toolbox user\'s guide. version 3. 2004.

[149] Tianjian Ji et al. Frequency and velocity of people walking. *Structural Engineer*, 84(3):36–40, 2005.

[150] Stephen J Preece, John Yannis Goulermas, Laurence PJ Kenney, and David Howard. A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data. *IEEE Transactions on Biomedical Engineering*, 56(3):871–879, 2009.

[151] GD Magoulas, MN Vrahatis, TN Grapsa, and GS Androulakis. A training method for discrete multilayer neural networks. *Mathematics of Neural Networks, Models, Algorithms and Applications*, pages 250–254, 1997.

[152] Edward M Corwin, Antonette M Logar, and William JB Oldham. An iterative method for training multilayer networks with threshold functions. *IEEE Transactions on Neural Networks*, 5(3):507–508, 1994.

[153] AH Khan and EL Hines. Integer-weight neural nets. *Electronics Letters*, 30(15):1237–1238, 1994.

[154] VP Plagianakos and MN Vrahatis. Neural network training with constrained integer weights. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages 2007–2013. IEEE, 1999.

[155] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Fixed point optimization of deep convolutional neural networks for object recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 1131–1135. IEEE, 2015.

[156] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[157] Simone Scardapane, Massimo Panella, Danilo Comminiello, and Aurelio Uncini. Learning from distributed data sources using random vector functional-link networks. *Procedia Computer Science*, 53:468–477, 2015.

[158] Simone Scardapane, Roberto Fierimonte, Dianhui Wang, Massimo Panella, and Aurelio Uncini. Distributed music classification using random vector functional-link nets. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE, 2015.

[159] José M Martínez-Villena, Alfredo Rosado-Muñoz, and Emilio Soria-Olivas. Hardware implementation methods in random vector functional-link networks. *Applied intelligence*, 41(1):184–195, 2014.

[160] Boris Igelnik and Yoh-Han Pao. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Transactions on Neural Networks*, 6(6):1320–1329, 1995.

[161] Peter JG Teunissen. An optimality property of the integer least-squares estimator. *Journal of Geodesy*, 73(11):587–593, 1999.

[162] J. Goldberger and A. Leshem. Iterative tomographic solution of integer least squares problems with applications to mimo detection. *IEEE Journal of Selected Topics in Signal Processing*, 5(8):1486–1496, Dec 2011.

[163] Peter van Emde Boas. *Another NP-complete partition problem and the complexity of computing short vectors in a lattice.* Universiteit van Amsterdam. Mathematisch Instituut, 1981.

[164] D. Micciancio. The hardness of the closest vector problem with preprocessing. *IEEE Transactions on Information Theory*, 47(3):1212–1215, Mar 2001.

[165] Xiao-Wen Chang and Qing Han. Solving box-constrained integer least squares problems. *IEEE Trans. Wireless Communications*, 7:277–287, 2008.

[166] Thomas F Brooks, D Stuart Pope, and Michael A Marcolini. Airfoil self-noise and prediction. 1989.

[167] Athanasios Tsanas and Angeliki Xifara. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49:560–567, 2012.

[168] Oguz Akbilgic, Hamparsum Bozdogan, and M Erdal Balaban. A novel hybrid rbf neural networks model as a forecaster. *Statistics and Computing*, 24(3):365–375, 2014.

[169] Toby Couture and Yves Gagnon. An analysis of feed-in tariff remuneration models: Implications for renewable energy investment. *Energy policy*, 38(2):955–965, 2010.

[170] Naim R Darghouth, Galen Barbose, and Ryan Wiser. The impact of rate design and net metering on the bill savings from distributed pv for residential customers in california. *Energy Policy*, 39(9):5243–5253, 2011.

[171] Rodolfo Araneo, Umberto Grasselli, and Salvatore Celozzi. Assessment of a practical model to estimate the cell temperature of a photovoltaic module. *International Journal of Energy and Environmental Engineering*, 5(1):2, 2014.

[172] Yang Du, Dylan Dah-Chuan Lu, Geoffrey James, and David J Cornforth. Modeling and analysis of current harmonic distortion from grid connected pv inverters under different operating conditions. *Solar Energy*, 94:182–194, 2013.

[173] Afshin Samadi, Robert Eriksson, Lennart Soder, Barry G Rawn, and Jens C Boemer. Coordinated active power-dependent voltage regulation in distribution grids with pv systems. *IEEE Transactions on Power Delivery*, 29(3):1454–1464, 2014.

[174] Jeroen Tant, Frederik Geth, Daan Six, Peter Tant, and Johan Driesen. Multi-objective battery storage to improve pv integration in residential distribution grids. *IEEE Transactions on Sustainable Energy*, 4(1):182–191, 2013.

[175] Rodolfo Araneo, Sergio Lammens, Marco Grossi, and Stefano Bertone. Emc issues in high-power grid-connected photovoltaic plants. *IEEE Transactions on Electromagnetic Compatibility*, 51(3):639–648, 2009.

[176] Rodolfo Araneo, Luigi Martirano, Salvatore Celozzi, and Chiara Vergine. Low-environmental impact routeing of overhead power lines for the connection of renewable energy plants to the italian hv grid. In *Environment and Electrical Engineering (EEEIC), 2014 14th International Conference on*, pages 386–391. IEEE, 2014.

[177] Rodolfo Araneo, Salvatore Celozzi, and Chiara Vergine. Eco-sustainable routing of power lines for the connection of renewable energy plants to the italian high-voltage grid. *International Journal of Energy and Environmental Engineering*, 6(1):9–19, 2015.

[178] Xiaochen Wang, Peng Guo, and Xiaobin Huang. A review of wind power forecasting models. *Energy procedia*, 12:770–778, 2011.

[179] Antonello Rosato, Rosa Altilio, Rodolfo Araneo, and Massimo Panella. Embedding of time series for the prediction in photovoltaic power plants. In *Environment and Electrical Engineering (EEEIC), 2016 IEEE 16th International Conference on*, pages 1–4. IEEE, 2016.

[180] Massimo Panella. Advances in biological time series prediction by neural networks. *Biomedical Signal Processing and Control*, 6(2):112–120, 2011.

[181] Thomas Kolarik and Gottfried Rudorfer. Time series forecasting using neural networks. In *ACM Sigapl Apl Quote Quad*, volume 25, pages 86–94. ACM, 1994.

[182] Simon Haykin. *Neural networks: a comprehensive foundation.* Prentice Hall PTR, 1994.

[183] Jyh-Shing Roger Jang, Chuen-Tsai Sun, and Eiji Mizutani. Neuro-fuzzy and soft computing; a computational approach to learning and machine intelligence. 1997.

[184] Henry Abarbanel. *Analysis of observed chaotic data.* Springer Science & Business Media, 2012.

[185] M Panella, A Rizzi, FM Frattale Mascioli, and G Martinelli. Anfis synthesis by hyperplane clustering. In *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, volume 1, pages 340–345. IEEE, 2001.

[186] Massimo Panella, Luca Liparulo, and Andrea Proietti. A higher-order fuzzy neural network for modeling financial time series. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 3066–3073. IEEE, 2014.

[187] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control.* John Wiley & Sons, 2015.

[188] D Lowe. Multi-variable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355.

[189] M. Panella, A. Rizzi, F.M. Frattale Mascioli, and G. Martinelli. ANFIS synthesis by hyperplane clustering. In *Proc. of IFSA/NAFIPS*, volume 1, pages 340–345, Vancouver, Canada, 2001.

[190] Z.i He and A. Cichocki. An efficient K-hyperplane clustering algorithm and its application to sparse component analysis. In D. Liu et al., editor, *Lecture Notes in Computer Science*, volume 4492, pages 1032–1041. Springer-Verlag, Berlin Heidelberg, Germany, 2007.

[191] A. Sharma, R. Podolsky, J. Zhao, and R. A. McIndoe. A modified hyperplane clustering algorithm allows for efficient and accurate clustering of extremely large datasets. *Bioinformatics*, 25(9):1152–1157, 2009.

[192] GAF Seber and CJ Wild. Nonlinear regression. 1989. *Search PubMed*, pages 325–365, 1989.

[193] Zhaoshui He and Andrzej Cichocki. An efficient k-hyperplane clustering algorithm and its application to sparse component analysis. *Advances in Neural Networks–ISNN 2007*, pages 1032–1041, 2007.

[194] Ashok Sharma, Robert Podolsky, Jieping Zhao, and Richard A McIndoe. A modified hyperplane clustering algorithm allows for efficient and accurate clustering of extremely large datasets. *Bioinformatics*, 25(9):1152–1157, 2009.

[195] E. Kononov. Visual recurrence analysis (vra), version 4.2. `http://www.visualization-2002.org/VRA_MAIN_PAGE_.html`, 1999.

[196] Antonello Rosato, Rosa Altilio, Rodolfo Araneo, and Massimo Panella. Prediction in photovoltaic power by neural networks. *Energies*, 10(7):1003, 2017.

[197] Ronald A Fisher. The statistical utilization of multiple measurements. *Annals of Human Genetics*, 8(4):376–386, 1938.

[198] Wojtek Krzanowski. *Principles of multivariate analysis.* OUP Oxford, 2000.

[199] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM, 2001.

[200] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525–533, 1993.

[201] Lior Rokach and Oded Maimon. *Data mining with decision trees: theory and applications.* World scientific, 2014.

[202] Donald F Specht. Probabilistic neural networks. *Neural networks*, 3(1):109–118, 1990.

Appendix A

Clustering Validation Indexes

As introduced in Section 2.3.2, the validation procedure can be difficult in the unsupervised learning context, since no label is able to give information on how the algorithm performs when tracking the underlying structure of the data. In this thesis, two kinds of evaluation criteria have been used. Their are based on either an internal evaluation, or an external one. In the former, the clustering is summed up to a single quality score, whereas in the latter the clustering is compared to an existing ground truth classification. The next sections is about some of the validity indexes used in the overall work to judge the validity of the proposed techniques.

A.1 Internal Validity Indexes

These indexes are based on the data clustered itself, and usually, assign the best scores to the algorithm able to produce clusters with the high similarity within a cluster and the low similarity between clusters. The list below present some of the validity indexes used to evaluate the performances in the unsupervised proposed approaches:

- *Davies-Bouldin Index* [53] is an internal evaluation index, where quantities and features inherent to the dataset are used to verify the efficiency of the clustering. Given n patterns, being C_i the cluster of data points, and X_j the vectors assigned to cluster CC_i, then the average distance of the points belonging to the cluster C_k and their barycenter A is expressed as follows:

$$\delta_i = (\frac{1}{T_i}\sum_{j=1}^{T_i} |X_j - A_i|^p)^{1/p} \tag{A.1}$$

  where A_i is the centroid of C_i and T_i is the size of the cluster i. Usually, the value of p is equal to two, so the Euclidean distance function between the centroid of the cluster, and the individual feature vectors are considered. Denoting the distance between the barycenters A_i and A_j of clusters C_i and C_j:

$$\Delta_{i,j} = \|A_i - A_j\|_p = (\sum_{k=1}^{n} |a_{k,i} - a_{k,j}|)^{1/p} \tag{A.2}$$

among all the clusters the Davies-Bouldin index is the mean value of the quantities M_k:

$$DB = \frac{1}{K}\sum_{k=1}^{K} M_k = \frac{1}{K}\sum_{k=1}^{K} \max_{k' \neq k}(\frac{\delta_k + \delta_{k'}}{\Delta_{kk'}}) \tag{A.3}$$

where M_k is defined as:

$$M_k = \max_{k' \neq k}(\frac{\delta_k + \delta_{k'}}{\Delta_{kk'}}) \tag{A.4}$$

The algorithm with the lowest Davies-Bouldin index will be considered as the best one.

- *Dunn Index* [54] is of the same group of the Davis-Bouldin Index, because it is an internal evaluation scheme, where the result is based on the clustered data itself. The aim is to identify sets of clusters that are compact, with a small variance between members of the cluster, and well separated, where the means of different clusters are sufficiently far apart, as compared to the within cluster variance. Let C_i be a cluster of vectors and x and y be any two n dimensional feature vectors assigned to the same cluster C_i. Introducing the distance between clusters C_k and C_t:

$$d_{ij} = \min_{i,j} \left\| M_i^k - M_j^t \right\| \tag{A.5}$$

and the largest distance separating two distinct points in the clusters:

$$D_k = \max_{i,j i \neq j} \left\| M_i^k - M_j^t \right\| \tag{A.6}$$

the smallest distance between points of different clusters and the largest within-cluster distance can be formulated as:

$$\begin{cases} d_{min} & = \min_{k \neq t} d_{kt} \\ d_{max} & = \max_{1 \leq k \leq K} D_k \end{cases} \tag{A.7}$$

Then, the Dunn index is defined as the quotient of d_{min} and d_{max}:

$$C = \frac{d_{min}}{d_{max}} \tag{A.8}$$

The intra-cluster distance may be measured in several ways, such as the maximal distance between any pair of elements in cluster k. Algorithms that produce clusters with high Dunn index shall be preferred.

- *DW-DB index* [55] the double weighted Davies-Bouldin index is a modified version of the DB Index. It avoids to falling into some local minimums that affect the standard Davies-Bouldin Index. Introducing:

$$FE = \sum_{\vec{x}_i \in E} (\mu_\alpha(\vec{x}_i) - \mu_r(\vec{x}_i)) = \sum_{\vec{x}_i \in E} fe(\vec{x}_i) \tag{A.9}$$

where μ_α is the membership value for the wrong class and μ_r is the membership value to the correct class (whose label is derived from the data set). If the crisp case is considered, then FE will be equal to the total number of misclassified patterns (n_{err}). The second quantity is the overall fuzzy reliability (FR), defined as:

$$FR = \sum_{\vec{x}_i \notin E} (\mu_r(\vec{x}_i) - \max_{j \neq r} \{\mu_j(\vec{x}_i)\}) = \sum_{\vec{x}_i \notin E} fr(\vec{x}_i) \tag{A.10}$$

In the crisp case, FR is equal to the total number of well-classified patterns ($N - n_{err}$). The DW-DB index can be so expressed as follows:

$$FCQ = \frac{\sum_{\vec{x}_i \in E} fe(\vec{x}_i) + \sum_{\vec{x}_i \notin E}(1 - f_r(\vec{x}_i))}{N} \tag{A.11}$$

By considering (A.9) and (A.10), the (A.11) can be rewritten as follows:

$$FCQ = \frac{FE + (N - n_{err}) - FR}{N} \tag{A.12}$$

A.2 External validity indexes

In these criteria, the results are evaluated on the basis of data for clustering, for example by the known class labels. The problem is the usually practical applications, such labels are unknown. On the other hands, labels are only a partitioning of data set, implying that a different, perhaps better clustering might even exists. Otherwise, results are compared with those obtained by benchmark approaches in order to analyze how close the clustering results are in respect to predetermined benchmark classes. In the following list, a further detail on some particular indexes that are used for comparing the results of the proposed approaches is given. In particular, in the indexes we refer on the following quantities:

- True Positive (TP): the object belonging to positive class and classified as positive;
- False Positive (FP): the object belonging to negative class and classified as positive;
- True Negative (TN): the object belonging to negative class and classified as negative;
- False Negative (FN): the object belonging to positive class and classified as negative.

A visual representation can be given as follows:

- *F-Measure* ([60]) is used to reduce the number of false negatives by weighting recall through a parameter $\beta \geq 0$. It is based on two features:

$$\begin{aligned} P &= \frac{TP}{TP + FP} \\ R &= \frac{TP}{TP + FN} \end{aligned} \tag{A.13}$$

Prediction outcome

		p	**n**	total
actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

where P is the precision rate (the fraction of relevant instances among the retrieved ones) and R is the recall rate (the fraction of relevant instances that have been retrieved over the total amount of relevant ones). Then the F-measure can be calculated in the following way:

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{A.14}$$

When $\beta = 0$, the index coincides with the precision, while when it increases, the influence of the recall increases too. The index does not take into account the number of true negatives.

- *Fowlkes–Mallows index* ([60]) is used to evaluate the resulting clustering structure, by comparing it to an independent partition of the data according to a benchmark approach. It can be evaluated by the following formula:

$$FM = \sqrt{\frac{TP}{TP + FP} \cdot \frac{TP}{TP + FN}} \tag{A.15}$$

  The index is the geometric mean between the precision and the recall. High values of the index indicate shows similarity between the partitions. The higher these indices, the more similar the partitions are.

- *Rand Index* ([59]) like the others measures, this index is use to compare the results of clustering algorithms with benchmark classification. The Rand index measures the percentage of the correct decision in the following way:

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \tag{A.16}$$

  In this way, the two kinds of errors, false negative and false positive are equally weighted.

- *K Index* ([61, 85]), is a statistical measure of the agreement, into account the possibility of the agreement occurring by chance. The K index can be described as follows:

$$KI = \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e} \tag{A.17}$$

being p_0 the relative observed agreement among raters, it can be associated with the accuracy of the results, while p_e is the hypothetical probability of chance agreement. The probability of each observer can be evaluated randomly by seeing each category. When the index is equal to one, then there is a complete agreement among the results. Conversely, there is no agreement among the raters other than what would be expected by chance. p_e can be described in the following way:

$$p_e = \frac{1}{N^2} \sum_k n_{k1} n_{k2} \tag{A.18}$$

where N is the number of items, k is the class and n_{ki} represents the number of times rater i predicts category k.

Appendix B

Statistical tests

This section introduces some statistical tests that have been used in the thesis to perform inference on random variables. The statistical analysis is used when is necessary to compare the relationship between two statistical dataset. In particular, they are used to determine whether there is enough evidence to “reject” a hypothesis on a process. Specifically, a statistical test requires:

- H_0: a null hypothesis that has to be confirmed;
- H_1: an alternative hypothesis that might be true.

The test procedure is generally made in order to avoid the risk of rejecting the null hypothesis when it is true. This risk is what is usually called significance level of a test. In particular, when the relationship between the dataset is an unlikely realization of the null hypothesis, we could say that the difference is statistically significant. Obviously, the risk of failing to reject the null hypothesis when it is false is possible and two kinds of errors can occur: error of type 1 and type 2. The first one causes an incorrect rejection of a true null hypothesis, while the second one causes an incorrect retaining of a false null hypothesis.

There is a wide range of statistical tests. Choosing which statistical test to use depends on the type of data, variable and research design. Below are summarized the statistical tests used in this thesis.

- *Pearson Coefficient* (r)*:* is used to test the correlation between two continuous variables. It is based on the realization of the best fit through the data of the two variables, and the measures show the distance between data points and this line. It is evaluated in the following way:

  $$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\sqrt{\sum_i (y_i - \bar{y})^2}} \tag{B.1}$$

  Essentially, it is given by the ratio between the two variables covariance, whose correlation has to be tested, and the single variance of each one. It generally takes value ranging from -1 and 1. A 0 value indicates a no correlation between data. Values greater than 0 indicates a positive correlation, while values less than 0 indicates a negative correlation (if a variable increases, the other decreases). Obviously, the closer the coefficients’ value to 1 or -1, the smaller the variation around the line of the best fit. The index is not able to tell the difference between dependent and independent variables, and the variables should approximately be normally distributed. Additionally, it is not able to capture nonlinear relationship between the variables.

- *Bland and Altman Plot* is a statistical test used to analyze the agreement between two different variables. It proposes an alternative analysis performed in a very simple way: considering n patterns they will be represented in a graph where the mean of the two measurements is reported on the x-axis, whereas the difference between the two values is reported on the ordinate. As a result, the Cartesian coordinates will be evaluated as follows:

$$S(x,y) = \left(\frac{S_1 + S_2}{2}, S_1 - S_2\right) \tag{B.2}$$

  Often, it is better to represent these values with a log transformation of the measurements before beginning the analysis:

$$S(x,y) = \left(\frac{log_2 S_1 + log_2 S_2}{2}, log_2 S_1 - log_2 S_2\right) \tag{B.3}$$

  The test is used to evaluate the agreement between two different instruments, or a possible systemic difference, this time by considering the bias among the variables and the standard deviation of the difference's deviation. Often, the 95% limit of agreement is used to better visualize the number of patterns which are statistically similar among the two methods of measurement. Consequently, the final result is composed by a scatter diagram of the mean difference, represented by a horizontal lines and the limits of agreement which represent the mean difference plus and minus 1.96 times the standard deviation of the differences. It only defines the intervals of agreements, and it does not say whether those limits are acceptable or not.

- *Limit of Agreement (LoA)* is used in the Bland and Altman plot to estimate the interval within which a proportion of the differences measurements lies. It is generally used to analyze the differences between the patterns extracted by two statistical methods. It considers both the random error (precision) and the bias among the measures. It is important to underline that the construction of this type of graph needs the assumption of normality among the differences.

- *Intra-class correlation coefficient (ICC)* measures the correlation among the variables in the same group. There are many kinds of ways to evaluate this model, the most important one concerns the analysis of variance (ANOVA), which is defined in terms of the random effects model:

$$Y_{ij} = \mu + \alpha_j + \epsilon_{ij} \tag{B.4}$$

  where Y_{ij} is the i^{th} observation in the j^{th} group, while μ, α_j, ϵ_{ij} are respectively the unobserved overall mean, random effect shared by all values in group j and unobserved noise term. Generally, the index is evaluated as:

$$\frac{\sigma_\alpha^2}{\sigma_\alpha^2 + \sigma_\epsilon^2} \tag{B.5}$$

where σ_α^2 and σ_ϵ^2 are the variances of α_j and ϵ_{ij}. The index is not always negative, and measures the proportion of total variance that is "between groups". It ranges from 0 to 1. A value close to 1 indicates high similarity between values in the same group, while values close to 0 indicates a low similarity.

Appendix C

Classification algorithms

In chapter 8 and 9 several classification procedures have been used to solve a specific feature selection problem. In particular, eight widely used classification algorithms, whose performances have been ascertained in many real-world problems compared to well-known classification benchmarks, are considered. Below, a short description of them is provided.

Linear Discriminant Analysis (LDA): [197] tries to characterize data using a linear polynomial in order to divide patterns into two or more classes. It maximizes the discriminatory information between classes by using the Fisher discriminant technique for surface separation. In particular, relying on the Bayes' rule, the conditional probability $P(X|y)$ is modeled as a multivariate Gaussian distribution:

$$p(X|y=k) = \frac{1}{(2\pi)^n|\Sigma_k|^{1/2}} exp(-\frac{1}{2}(X-\mu_k)^t\Sigma_k^{-1}(X-\mu_k)) \tag{C.1}$$

In that way, we just need to estimate the class priors $P(y = k)$ from the training data, using class k instance's proportion, the mean μ_k and the covariance matrices Σ_k. In this model, classes are supposed to have the same covariance matrix: $\Sigma_k = \Sigma$ $\forall$ k, so that the homoscedastic hypothesis must be satisfied from input data.

Quadratic Discriminant Analysis (QDA), similarly to the LDA, it tries to make predictions by using the Bayes's rule to model $P(X|y)$ as a multivariate Gaussian distribution (C.1) where the homoscedastic hypothesis is overcome and no assumption on the input data has to be realized [198]. It is more suited for real contexts, where heterogeneous covariance matrices could be presented $\Sigma_i \neq \Sigma_j$ for some $i \neq j$. In this way, we cannot throw away the quadratic terms, and the discriminant function becomes a quadratic decision surface. The procedure is still the same as in the same of the LDA approach, where the aim is to find the class k that maximizes the quadratic discriminant function. The model is able to better fit the data, compared to the LDA procedure, making necessary an increase of the parameters to be trained. Therefore, a trade-off between the fitting and the complexity of the model has to be performed to decide which choice is better between the two models.

Naive Bayes classifier is a statistical technique that tries to verify if an element belongs to a class [199]. The algorithm is based on Bayes's theorem and takes in inputs the attributes of the relative class for each object ($x = x_1, \ldots, x_n$). Then, it calculates various conditional probability and assigns the object to the class with the highest probability of belonging. The correct classification is obtained when the conditional probability of one class C respected to the attributes $\mathrm{X}_n(\mathrm{P}\{\mathrm{C}|x_1, x_2, ..., x_n\})$ is the maximal one:

$$\hat{y} = \underset{k\in\{1,...,K\}}{\arg\max}\, p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{C.2}$$

This is known as the maximum a posterior decision rule. To estimate the parameters for the feature, an assumption on their distributions has to be performed. The most frequent ones are the Gaussian, the Multinomial or the Bernoulli.

Fuzzy Inference System (FIS) is used to partition the feature space into fuzzy classes [200]. The aim is to optimize the system's parameters, such as the membership function defined for each feature of the parametrized t-norms. Any FIS is based on an "if-then" rule, and the learning procedure is based on three different steps:

- Clustering of training data: a subtractive clustering is applied to the training data in order to automatically determine the number of clusters;
- Generation of the initial fuzzy rules: for a cluster j belonging to class C_i the "if-then" rules can be written as follows:

  $$\text{if } X_1 \text{ is } A_{j1} \text{ and } X_2 \text{ is } A_{j2} \text{ and } \ldots \\ \text{then class is } C_j$$

  where X_k is the k-th element of a feature vector X and A_{jk} is the membership function;
- Optimization of the fuzzy rules: each membership function is tuned according to gradient descent procedure.

Once trained, the FIS can classify new vectors using its set of fuzzy rules.

Classification and regression tree (CART) operates by the recursively splitting of the data until ending points, which are defined by some preset criteria, are achieved [201].
Each root node represents a single input variable (x) and a split point on that variable, while the leaf nodes represent the output variable (y). When a new point is presented to the model, the tree is crossed by evaluating the specific input, which starts at the root node of the tree, and ends when a stopping criterion is satisfied, such as the minimum number of training instances assigned to each leaf node of the tree. Being nonlinear relations between features and classes, they represent a correct trade-off among computational complexity and accuracy.

K-Nearest Neighbors (KNN) approach, assigns a class on the basis of the most frequent one of the patterns in the neighborhood [9]. It does not require any assumptions on data, and is based on the following steps:

- Training: space is partitioned into several regions, on the basis of the objects in the training set;
- Distance calculation: the Euclidean distance is evaluated among all the possible pairs in the dataset. On the other hand, the Manhattan or the Minkowski distance could also be used for this purpose;
- Classification: each point is assigned to the closer class;

It is useful to assign weight to the neighbors' contributions so that the nearest ones contribute more to the average than the more distant ones. Obviously, the value of K will influence the results, and the correct choice of the parameter has to be evaluated in terms of training error rate and validation error rate. Generally, a large value of K is more precise as it reduces the overall noise even though there is no guarantee. Cross-validation is another technique to determine a good K value by using an independent dataset to validate it. Historically, the optimal parameter has been between $3 - 10$.

Support Vector Machine (SVM) is a supervised learning approach that should be applied for both regression and classification problems [198]. Based on the solution of a quadratic convex problem, it could be used for finding global minimum also in non linear complex problems. It represents the input data $[(x_1, y_1), \ldots, (x_N, y_N)]$ as points in a graph, and the classification procedure is performed by finding the hyperplane able to maximize margin among the patterns, and by minimizing the mistakes on the training data. If the training data is linearly separable, we can select two parallel hyperplanes to separate them in order to maximize the margin (the region bounded by these two hyperplanes) otherwise, an additional consideration has to be performed. In particular, the potentiality of this model lies non-linear problems solving capability carried out by performing a transformation into an high-dimensional feature space with kernel functions. In fact, in the transformed space, data becomes linearly separable and the problem is easily solved.

Probabilistic Neural Network (PNN) is a statistical algorithm where the operations are organized into a multilayered feedforward network [202]. It is organized in four layers:

- Input layer: composed of $N - 1$ neurons when there are N number of categories,and each one of them represents a predictor variable;
- Pattern layer: contains a single neuron for each case in the training data, and when a new input is presented to the network, it computes the Euclidean distance and then applies the radial basis function kernel;
- Summation layer: is characterized by a neuron for each category of the target variable. It sums up the contribution for each class of inputs and produces its net output as a vector of probabilities.
- Output layer: compares the weighted votes for each target, and the largest one is used to predict the target category. In this way, the associated class label is determined.