University of Bologna

Department of Electrical Engineering

Ph.D. in Electrical Engineering
XVI course

DEVELOPMENT OF ARTIFICIAL INTELLIGENCE SYSTEMS FOR ELECTRICAL INSULATION DEFECT IDENTIFICATION THROUGH PARTIAL DISCHARGE MEASUREMENTS

Ph.D. Thesis of
MARCO CONTI

Tutor
Prof. GIAN CARLO MONTANARI

Ph.D. Coordinator
Prof. FRANCESCO NEGRINI

2000 - 2003

Section 1

Section 1 provides basic concepts about PD detection and PD mechanisms. It is aimed at constituting a guide for PD interpretation by human operators, independently on the derivation of specific quantities through statistical processing.

In this section:

Introduction

The demand for maintenance cost reduction in power supply is progressively increasing, inducing network operators to concentrate on the optimization of technical management and the maximization of the life of existing components. The life of an electrical apparatus in medium or high voltage is strictly related to the insulation condition. The performance of an insulation, it turn, is determined prevailingly by the presence of defects, which cause a drastic decay of its properties, through progressive degradation. Therefore, to achieve predictive maintenance tools, it is fundamental to gain insight to local alteration process, through the identification of insulation defects. In this light, the acquisition and interpretation of partial discharge (PD) signals constitute a very promising tool. In fact, PD are localized electrical discharges that only partially bridge the insulation between conductors [1]. Thus, they are localized in the insulation defects, being, at the same time, symptom and cause of local degradation. Hence, PD may provide precious information about the typology and degradation stage of insulation defects. Such information, linked to the knowledge about the characteristics of the specific HV component, can actually guide the operator in decision making for condition based maintenance (CBM), quality control and on line monitoring. However, although PD based diagnostics has great potentials, it must be underlined that PD measurements, to be useful, must be carried out following a proper acquisition strategy, supported by suitable instruments.

Object of the present study - summary

It is generally agreed, [2], that PD interpretation requires systematic field experience, together with the understanding of the physical phenomena associated to discharges. Indeed, human experts cannot be available every time PD interpretation is needed, hence it is fundamental to get a machine to evaluate the acquired data automatically, on the basis of prior knowledge. Furthermore, the knowledge about specific aspects of PD phenomena is still inadequate, also because of the variety of industrial applications, therefore techniques are needed to derive interpretation rules from an appropriate database, which collects the result of both laboratory tests and field experience.
In this light, the aim of the present study can be summarized in three issues, correspondent to the three sections of the thesis.

1. Introduction to those basic concepts which constitute the pre-requisite for the effectiveness of PD interpretation. These concepts are pertinent to two main aspects: discrimination of different PD sources (and noise rejection) and general description of the discharge mechanisms.

2. Description of the techniques used to process automatically the acquired PD data in order to derive significant and robust diagnostic markers. Description of the identification strategy and of the subsequent database construction. Discussion of the principles and assumptions which constitute the basis for PD interpretation. Testing and examples.
3. Description of the mathematical methods which were developed in order to transfer human experience to the machine, as well as to derive rules and concepts directly from a training database, by means of artificial intelligence techniques.

Discharge detection

Partial discharges are the consequence of local electrical field concentrations in the insulation or on the surface of the insulation, normally, in correspondence of insulation defects. A PD consists of a charge transfer in the defect, which produces a current pulse to flow through the electrodes. PD may occur under either AC or DC voltage and can be detected in many different ways, e.g. electrically, acoustically, optically, etc. In this thesis only the case of AC stress and electrical detection is considered. The circuit used for the electrical detection of PD, according to [1], is the one represented in figure 1.1.

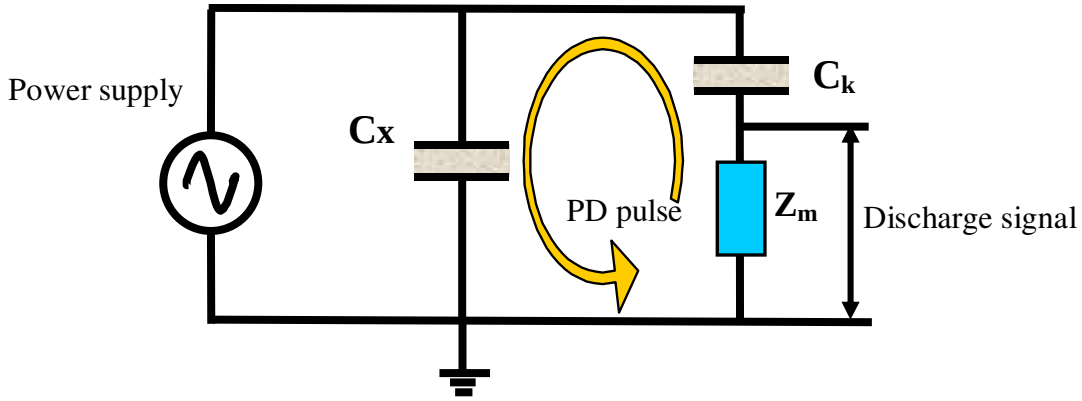


Fig. 1.1: *representation of the test circuit for electrical detection of PD.*

In Fig. 1.1, C_X schematizes the object under test, C_X a coupling capacitor and Z_m the measurement impedance. Since PD pulses have a rise time of the order of nanoseconds, they constitute high-frequency-content current signals. Therefore, the parallel of test object and coupling capacitor provides a low impedance path where PD pulses are likely to flow. In this way, PD pulses are detected as voltage drops across a measurement impedance, which is put in series to either the test object (*direct type circuit*) or the coupling capacitor (*indirect type circuit*).
Measuring systems which follow this principle of electrical detection can be divided in two categories, with respect to their acquisition strategy.

- *Wide band measuring system* – In combination with the coupling device, the PD detection

circuit is characterized by a transfer impedance Z(f) having fixed values of lower and upper limit frequencies. The bandwidth of the instrument, Δf, ranges from tens of MHz, up to GHz. This acquisition strategy allows a time-resolved detection of the pulse waveform.

- *Narrow band measuring system* – These instruments are characterized by a small bandwidth (Δf is of the order of tens of kHz) and a mid-band frequency which can be varied over a wide frequency range. In this case, the acquired pulse is somehow subjected to an integration process.

Quantities that are made available by the measurements are:

- PD pulse amplitude [mV];
- PD phase with respect to the applied voltage [degrees];
- PD time of occurrence [ms];
- PD pulse shape (only in the case of wide band measuring systems).

- Amplitude

To quantify discharges, a quantity called *apparent charge* is commonly used. Apparent charge is defined, [1], as that charge which, if injected within a very short time between the electrodes of the test object in a specified test circuit, would give the same reading on the measuring instrument as the PD current pulse itself. This quantity is expressed in pC and is usually referred to as *discharge magnitude*. The amplitude of acquired pulses differs from the correspondent discharge magnitude (i.e. the apparent charges associated to those pulses) for a scale factor. The determination of such a scale factor requires a *calibration* process, which consists of the injection of short-duration current pulses of known charge magnitudes across the terminals of the test object (more details about the calibration procedure can be found in [1]). The purpose of calibration is to obtain a quantitative estimation of PD charge which is independent from the test circuit (capacity of the test object, coupling capacitor, measurement impedance, acquisition device, etc.). In addition, calibration allows to verify that the measuring system is able to measure a specified PD magnitude correctly. In this light, the concept of *measuring sensitivity* is introduced. The measuring sensitivity is determined by the smallest apparent charge (PD magnitude) which the acquisition system can measure with negligible effect of background noise. *Background noise* includes (according to the definition provided by [1]) all signals detected during PD tests, which do not originate in the test object[1]. However, it must be observed that the apparent charge is not equal to the actual amount of charge locally involved at the site of the discharge, which cannot be measured directly [1]. In this light, it has to be considered that PD may occur at some location remote from the point of measurement, thus signals will be attenuated and distorted due to the propagation from the discharge site to the PD sensor. Indeed, the calibration process accounts for the attenuation of the

[1] Hence, background noise can be composed either of white noise in the measurement system or broadcast radio or other continuous or impulsive signals.

PD signal due to the whole measuring system, but does *not* account for the attenuation which depends on the location of the PD site within the test object. Such a consideration is particularly important when the object under test is large and/or complex, e.g. generators, [3].

- Phase

The evaluation of phase angles of PD events with respect to the applied voltage is carried out by means of a *synchronism* signal (proportional to the applied voltage and obtained through a voltage divider), and assuming that the duration of a PD pulse (μs) is negligible with respect to the period of the applied voltage (ms). One important concept to evaluate PD activities at AC voltage is *discharge polarity*. The polarity of a PD is determined by the sign of the electric field by which it is driven (local field), therefore it is evaluated on the basis of the PD phase. On the contrary, the sign of the detected pulse signals depends on the type of measurement circuit (direct or indirect). The convention adopted for polarity assignment is that a PD is positive if driven by a positive local field, negative if driven by a negative local field, as exemplified by figure 2.2.

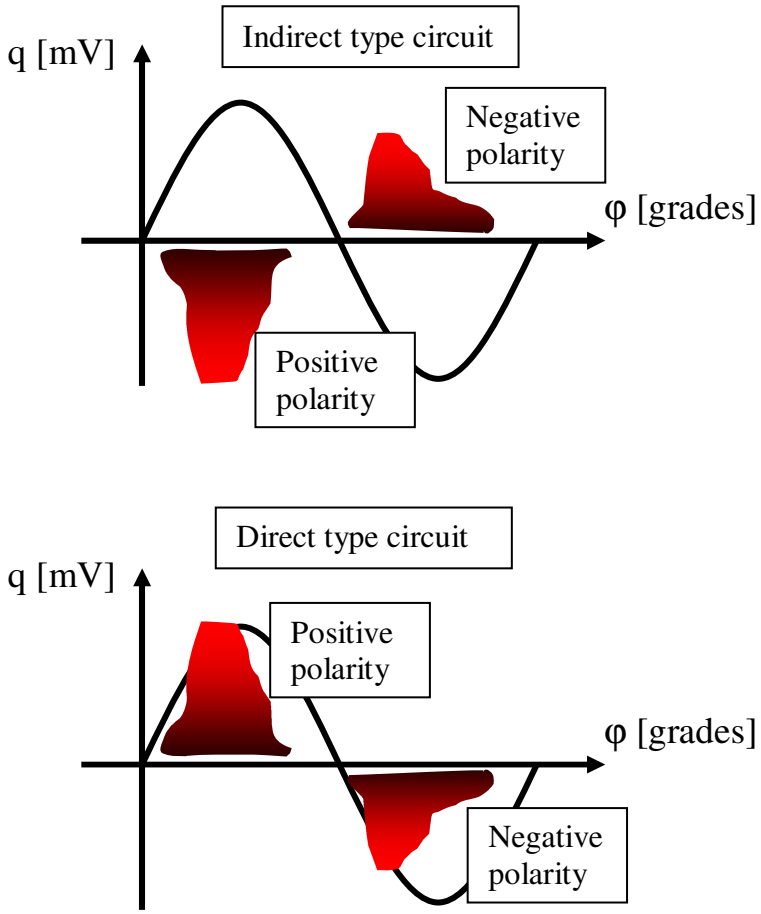


Fig. 2.2: *schematic representation of the discharge polarity assignment, according to circuit type, with respect to amplitude vs. phase diagrams.*

It must be underlined that polarity assignment requires more operative definitions, as well as a

technique to automate the process. However, this topic will be discussed later, in the second section of this thesis.

- Time of occurrence

Time of occurrence allows time sequence analysis of the discharge process and determines the PD repetition rate. The measurement of this quantity may be affected by *superposition error*, i.e. the overlapping of transient output pulse responses when the time interval between input current pulses is less than the duration of a single output response pulse, [1]. The possibility for the measurement system to avoid superposition error depends on its *pulse resolution time*, which is, in general, inversely proportional to the bandwidth. For wide band instruments, the pulse resolution time is of the order of μs.

- Pulse shape

PD pulse shape is made available by digital wide band measuring systems. Clearly, this kind of information is affected by the transfer function that PD signals face while traveling from the discharge site to the coupling device. In general, PD pulses are subjected to *attenuation*, depending on the losses in the signal transmission medium, *distortion*, due to complex capacitive and inductive coupling phenomena, and *reflection*, due to the changes of the traveling wave impedance of the test object along the propagation path. Therefore, it is not expectable that, in the majority of the situations, the measured pulse shape represents the true shape of the transient of charge transfer in the PD site. Indeed, pulse shape is strictly related to the measurement circuit and to the location of the PD source within the test object.

Separation of different PD sources (classification)

The purpose of PD interpretation is to derive information about specific insulation defects, contrarily to other techniques, e.g. loss factor measurements, space charge measurements, etc., which provide a bulk evaluation of the insulation of the object under test. Hence, it is essential that the dataset which constitutes the basis for PD interpretation is pertinent to a single PD phenomenon. In this light, one must face the situation where the measuring system acquires signals pertinent to different sources, because of the presence of either multiple defects discharging simultaneously or noise signals concomitant to PD signals. The separation of acquired signals with respect to their source will be addressed to as *classification process*. In this thesis work, a technique for signal classification based on pulse shape analysis is used, thanks to the adoption of an innovative, wide band measuring system. In fact, pulse shapes are heavily affected by the nature and location of their

source, as well as by the measurement circuit, [2]. Therefore, within a given measurement, it is assumed that different pulse shapes correspond to different sources. On the basis of this assumption, a technique was developed, [4], to separate the acquired signals in clusters which are homogeneous with respect to the pulse shape. Such a technique was not object of this thesis, which is focused on defect identification. However, because classification is a pre-requisite for the effectiveness of the identification process itself, a brief description of this technique will be given in the following.
The information provided by each acquired signal is synthesized by means of two quantities, equivalent time (T) and equivalent bandwidth (W), expressed by equations (1.1) and (1.2), respectively.

$$T^2 = \frac{\sum_{i=1}^{K} (t_i - t_0)^2 \cdot s_i(t_i)^2}{\sum_{i=1}^{K} s_i(t_i)^2} \tag{1.1}$$

In equation (1.1), K is the number of samples, t_i is the time associated to the i-th sample, s_i is the signal amplitude correspondent to the i-th sample (in the time domain) and t_0 is a time baricenter, obtained through equation (1.3).

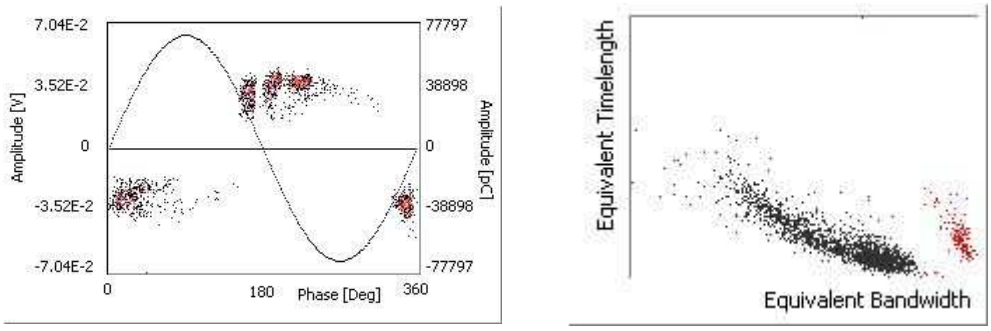
$$W^2 = \frac{\sum_{i=1}^{K} f_i^2 \cdot |X_i(f_i)|^2}{\sum_{i=1}^{K} |X_i(f_i)|^2} \tag{1.2}$$

In equation (1.2) f_i is the i-th sample frequency (in the frequency domain) and $X_i(f_i)$ is the i-th component of the signal, obtained through FFT transformation.

$$t_0 = \frac{\sum_{i=1}^{K} t_i \cdot s_i(t_i)^2}{\sum_{i=1}^{K} s_i(t_i)^2} \tag{1.3}$$

Thus, each pulse can be represented as a point in a two dimensional map (T vs. W), called *classification map*. Therefore, signals pertinent to the same source will correspond to points in the classification map which are close to each other; accordingly, signals pertinent to different sources will produce separate groups in the classification map. Hence, signal separation is carried out on the basis of the classification map, applying fuzzy clustering algorithms, [5].
To clarify the classification process, the result of a test where two PD activities were active at the same time will be shown in the following. The acquired data will be reported in two

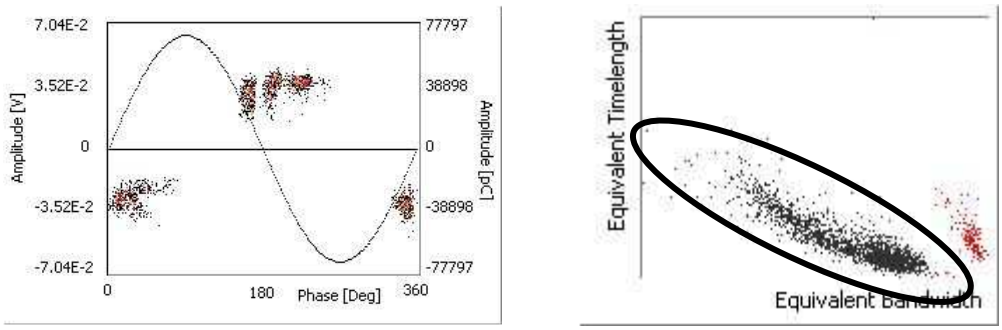
representations: classification map and PRPD pattern. The PRPD (Phase Resolved Partial Discharge) pattern represents the acquired signals in the amplitude vs. phase plane; in addition, each point in this map has a color, which indicates the number of PD occurred with a given amplitude and phase (brighter color corresponds to higher number). Hence, the PRPD pattern is a two-dimensional histogram. Figures 1.3 – 1.7 describe the whole classification process.



(A) (B)

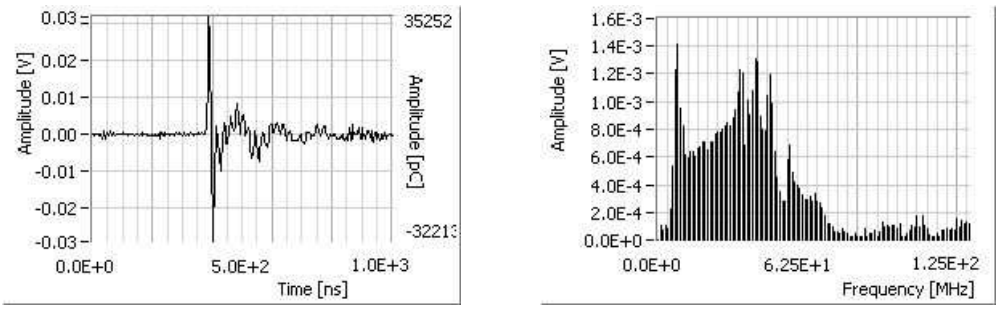
Fig. 1.3: *representation of the entire dataset, (A) PRPD pattern, (B) classification map.*

As can be observed in Fig. 1.3B, the acquired pulses can be grouped in two clouds in the classification map; accordingly, the PRPD pattern will be divided in the two sub-patterns of figures 1.4 and 1.6, correspondent to the pulse shapes of figures 5 and 7, respectively.



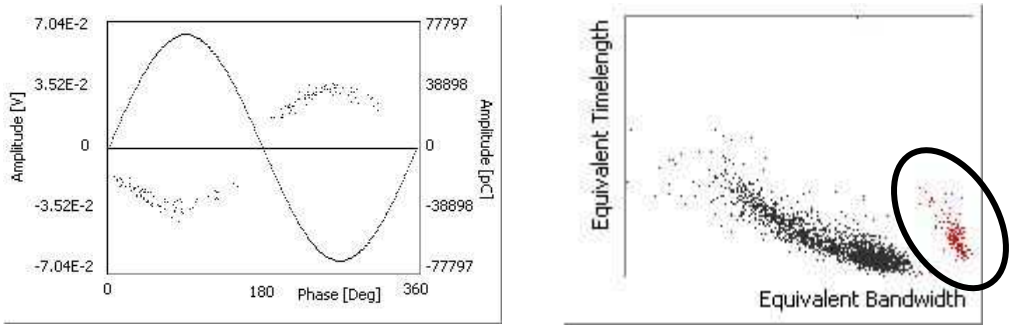
(A) (B)

Fig. 1.4: *sub-pattern (A) correspondent to the cluster in the classification map evidenced in (B).*



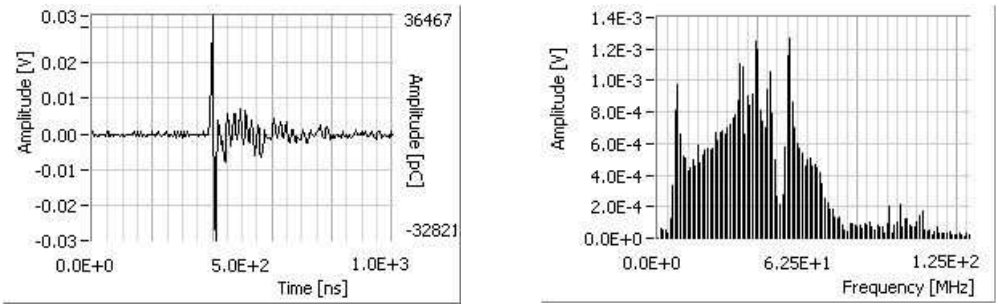
(A) (B)

Fig. 1.5: *typical pulse shape (A) and spectrum (B) relevant to the sub-set of Fig. 1.4.*



(A) (B)

Fig. 1.6: *sub-pattern (A) correspondent to the cluster in the classification map evidenced in (B).*



(A) (B)

Fig. 1.7: *typical pulse shape (A) and spectrum (B) relevant to the sub-set of Fig. 1.6.*

The problem of noise

PD measurements are often affected by noise. Disturbances may be divided in two categories, according to [1]:

- disturbances which occur even if the test circuit is not energized;
- voltage-correlated disturbances which occur when test circuit is energized.

In general, the problem of noise is particularly critical when the following conditions are verified:

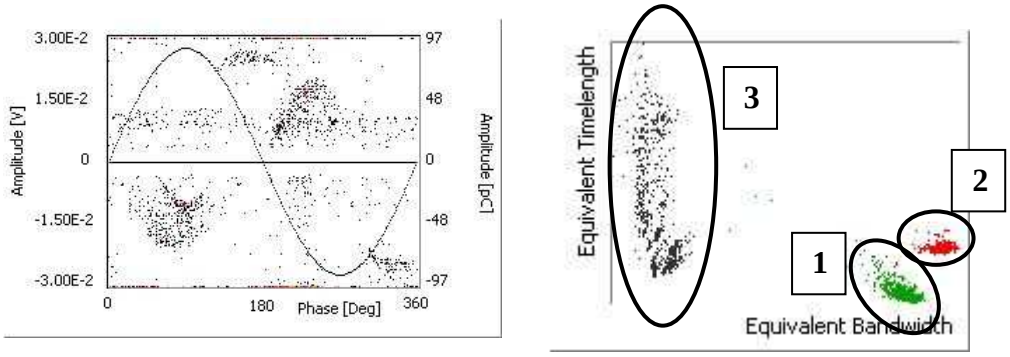
- The repetition rate of noise signals is large with respect to that of PD signals;
- The amplitude of noise signals is large with respect to that of PD signals.

Hence, the importance of noise rejection techniques is strongly dependent on the typology of HV apparatus under test, e.g. cable testing requires a much higher sensitivity with respect to measurement on generators.

Noise rejection may follow two main strategies:

- Avoid the acquisition of noise signals;
- Reject noise in a post-processing phase.

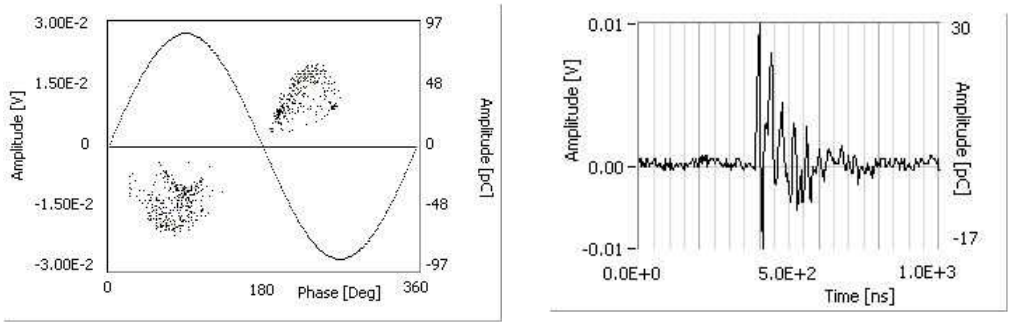
These two strategies may be used at the same time, too. However, some of the techniques which can be used to avoid noise acquisition require narrow band measuring systems, while post-processing rejection requires wide band measuring systems, therefore, some kind of trade off is needed. If a wide band measuring system is used (as in this thesis work), which is endowed with powerful signal processing tools, the only situation which is actually critic occurs when noise signals widely exceed PD signals both in amplitude and in repetition rate. In fact, if noise signals have high repetition rate but are small (usually, it is the case of disturbances which occur even if the test circuit is not energized), the majority of them can be avoided by raising the trigger level (i.e. by setting a high-pass filter on signal amplitude), then, the remaining part of noise signals can be rejected in the post-processing phase. If noise signals are characterized by a low repetition rate, they shall be rejected in post-processing phase, independently on their amplitude (usually, it is the case of voltage-correlated disturbances which occur when the test circuit is energized).
Figures 1.8 – 1.11 illustrates an example of noise rejection, carried out at post-processing phase by means of the classification technique described in the previous paragraph.



(A) (B)

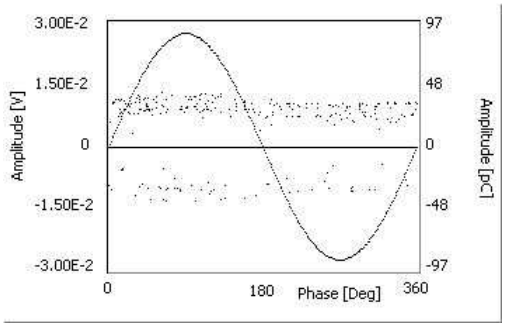
Fig. 1.8: *representation of the entire dataset, (A) PRPD pattern, (B) classification map.*

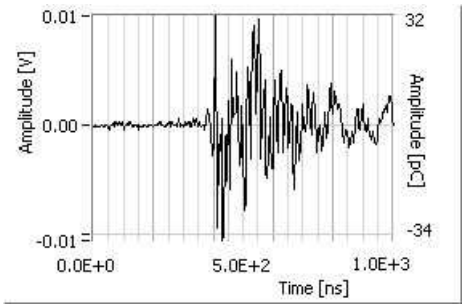
As can be observed in Fig. 1.8A, the acquisition is very noisy. Some disturbances have amplitude larger than the full scale of the instrument, thus obscuring, e.g., the evaluation of the maximum PD amplitude. In the classification map (Fig. 8B) three groups of pulses are singled out. Accordingly, three sub-patterns are obtained, as reported in Figs. 1.9-1.11.



(A) (B)

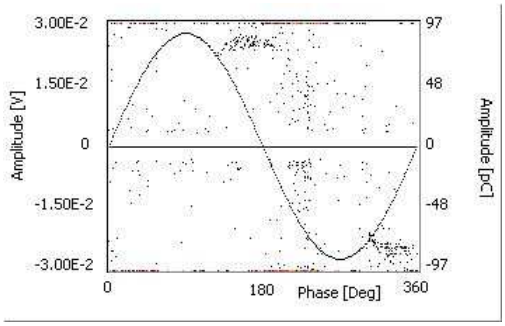
Fig. 1.9: *sub-set labeled 1 in Fig. 1.8B, (A) PRPD pattern, (B) typical pulse shape.*

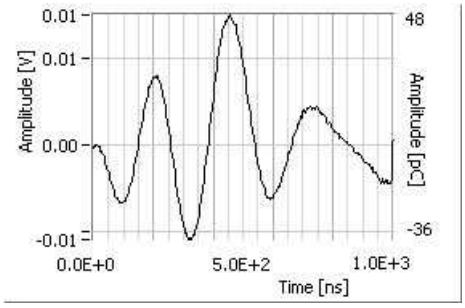




(A) (B)

Fig. 1.10: : *sub-set labeled 2 in Fig. 1.8B, (A) PRPD pattern, (B) typical pulse shape.*





(A) (B)

Fig. 1.11: *sub-set labeled 3 in Fig. 1.8B, (A) PRPD pattern, (B) typical pulse shape.*

It can be noted that the two disturbance phenomena belong to different categories, according to the distinction above. In fact, disturbance phenomenon labeled 2 in Fig. 1.8B is characterized by high frequency content pulses (Fig. 1.10B), it is not correlated with phase (Fig. 1.10A) and is present also when the circuit is not energized. The disturbance phenomenon labeled 3 in Fig. 1.8B is characterized by low frequency content pulses (Fig. 1.11B), it is correlated with phase (Fig. 1.11A) and is present only when the circuit is energized.

It must be underlined that the noise rejection techniques used in post processing phase actually consist of two steps:

- Separation of signals relevant to different sources;
- Identification of disturbances.

In the example above, only the first of these steps is reported. The second step consists in a pattern analysis, in particular of the phase distribution, aimed at the recognition of a given dataset as pertinent to a “PD activity” or a “noise activity”. Hence, such a procedure, described in detail in [6, 7], is part of the identification process. In the example reported here, it is relatively easy to recognize sub-patterns of Figs. 1.10A and 1.11A as pertinent to disturbances; however, sometimes it is more difficult to distinguish noise from PD phenomena and, in any case, it must be possible to run the whole process automatically.

Basic concepts about PD phenomena

According to [9], PD activities may be classified in three categories:

- Discharges in gases and at gas-insulator surfaces;
- Discharges in liquids and at liquid-solid interfaces;
- Discharges in solids (charge injection, electrical trees and water trees).

In this thesis work, only the first and the third of the above reported categories are considered. In particular, the present chapter focuses on the first category. Accordingly, it can be stated that partial discharges occur in air-filled portions of space (defects), without bridging the electrodes of the insulation system. Between the discharges and one or both electrodes a dielectric is present in the shape of a solid, liquid or gaseous insulator. Hence, PD belong to a far greater group of gas discharges. As described in [8], in all these discharges gas molecules are ionized by impact of electrons. The newly formed electrons gain speed in an electric field, ionizing more molecules by impact, so that an avalanche of electrons is formed. The electrons in the avalanche and the ions left behind move toward the electrodes, thus forming a passage of current through the gas, until they are deployed in correspondence of some surface. In this way, a charge separation / transfer takes place in the defect.
Hence, within the PD generating defect two main parts can be distinguished:

- *Discharge volume*: portion of space (usually constituted by air), where the partial discharges occur and which is (at least partially) surrounded by a sound dielectric.
- *Discharge surfaces*: surfaces which delimit the discharge volume in the direction of the electric field; thus they delimit the charge movement associated to the PD.

According to the definition of PD provided by [1], one of the discharge surfaces may coincide with an electrode, but not both.
On the basis of these concepts, two more definitions can be given:

- *Gap*: distance between the discharge surfaces.
- *Local field*: electric field in the discharge volume.

Partial discharges are caused by the *ionization* process induced by the electric field in the defect. The ionization process produces electronic avalanches, which do not necessarily develop into PD. In general, an avalanche develops into a PD if it grows enough to bridge the discharge surfaces, covering at least the gap length. This process can occur only if the local field exceeds a critical value, $E_{critical}$. In other words, if the local field is below $E_{critical}$, electronic avalanches may occur, but they will not develop into PD.
Therefore, two conditions must be verified for a PD to take place in a given defect:

- A starting electron is available to initiate an electronic avalanche;

- The local electric field exceeds the critical value.

The presence of initiatory electrons in the defect can be regarded as a stochastic processes [9-13] and therefore a stochastic *time lag* occurs, Δt_{lag}, after the local field reaches the critical value before an avalanche starts. In practice, the value of the local field at which a PD may be observed under AV voltage is always larger than the critical value; it would be equal to the critical value if the time lag was null.

In the study of PD phenomena an important concept is the *inception* voltage (field) of a PD activity. The partial discharge inception voltage is defined in [1] as the applied voltage at which repetitive PD are first observed in the test object, when the voltage applied to the object is gradually increased from a lower value at which no PD are observed. Clearly, for a given defect of a given test object, any value of applied voltage corresponds to a specific value of applied field, which, in its turn, corresponds to a specific value of the local field.

In this light, the following quantities are defined with regard to PD inception:

- *Observed inception voltage*, V_{inc}: inception voltage calculated according to [1];
- *Theoretical inception voltage*, V^T_{inc}: value of the applied voltage for which the local field assumes the critical value.

Therefore, the following equation can be written.

$$V_{inc} \geq V_{inc}^{T} \quad \left(V_{inc} = V_{inc}^{T} \ \text{ if } \Delta t_{inc} = 0\right) \tag{1.4}$$

Furthermore, a quantity called *overvoltage ratio, ν*, is defined in (1.5), which relates the test voltage, V_t, to the inception voltage.

$$\nu = \frac{V_t}{V_{inc}} \tag{1.5}$$

On the basis of the distinction made between observed and theoretical inception voltages, a quantity called *theoretical overvoltage ratio* is defined by (1.6)

$$\nu^T = \frac{V_t}{V_{inc}^T} \tag{1.6}$$

Clearly, from the definitions (1.5), (1.6) and from the condition expressed by (1.4), equation (1.7) follows.

$$\nu \leq \nu^T \quad \left(\nu = \nu^T \ \text{ if } \Delta t_{inc} = 0\right) \tag{1.7}$$

For the understanding and modeling of PD activities, the "true" quantities V^T_{inc} and ν^T are more important than the correspondent observed ones, [9-14]. Therefore, the measured quantities defined according to [1] are significant as much as the inception time lag plays a negligible role. In many circumstances such a condition is not verified [9, 12, 14].
Thus, the local field and the availability of initial electrons in the defect are two fundamental quantities for PD activity and shall be considered in more detail.
Being E_0 the electric field applied to the test object, the local electric field, E_i, can be expressed by (1.8),

$$E_i = f \cdot E_0 \tag{1.8}$$

where f is a dimensionless average field enhancement factor, which depends, in general, on the relative permittivity of the dielectric which surrounds the defect, the shape of the defect and the geometry of the discharge surfaces.
When a discharge occurs, the local field is subjected to a drop, as expressed by equation (1.9),

$$\Delta E = E_i\left(t^-\right) - E_i\left(t^+\right) \tag{1.9}$$

where t^- and t^+ are the times in correspondence of which the PD begins and ends, respectively.
The PD amplitude, q, is proportional[2] to the field drop, as expressed by equation (1.10).

$$q \propto \Delta E \tag{1.10}$$

As regards the availability of initial electrons in the defect, a quantity called *electron generation rate*, EGR, can be introduced. Such a quantity depends on many factors, [13], which are just listed in the following, since a more detailed discussion of this topic lies outside the object of the present thesis.
Generally speaking, the basic mechanisms of generation of initial electrons are:

- Background radiation;
- Surface emission (photo-ionization, impact of positive ions on the cathode, Schottky effect);
- Detrapping of electrons from traps at the insulator surface by the electric field.

The following quantities are related to the electron generation rate:

- R_0: value of the background radiation (is present even in the absence of other factors);

[2] Quantitative expressions for the calculation of the discharge amplitude under specific conditions can be found, e.g., in [9].

- Φ: effective work function (influences detrapment of electrons by the electric field);
- T: temperature in the defect (influences surface emission);
- p: pressure in the defect (influences surface emission);
- σ: conductivity of discharge surface (influences surface emission).

These initial electron generation mechanisms may be present separately or together, and their relative weight in the discharge process depends on the geometry and on the physical condition of the defect, on the value of the applied field and on the degradation stage. An important role in the physical description of PD phenomena is played by the discharge mechanism itself (e.g. Townsend-like or streamer-like discharges), as shown in [12]. Practical information about the discharge typology could be provided by the pulse shape; however, most of the times distortion phenomena unable this kind of speculation for the purpose of identification.

Connections among measured quantities and quantities which describe the PD activity.

- PD pulse amplitude → local field drop / quantity of transferred charge / discharge length;
- PD phase with respect to the applied voltage → local field / initial electron generation;
- PD time of occurrence → availability of initial electrons / shielding effects;
- PD pulse shape → discharge typology (if distortion effect is negligible).

PD mechanism in insulation voids

Insulation voids consist of cavities embedded in solid insulation. Actually, voids may vary in shape, being flat or narrow with respect to the local field direction, and in location with respect to the electrodes, i.e. they can be attached to one of the two electrodes or embedded in the insulation. However, in this simplified description, focus is made on a cavity of compact shape embedded in the insulation.
This kind of defects (voids) is generally provided with two basic characteristics:

- The local electric field is about homogeneous in the discharge volume;
- The gap (distance between discharge surfaces in the direction of the field) is fixed, i.e. it does not depend on the applied field or environmental conditions.

The most peculiar aspect of PD activity in voids is that the charge transferred by the PD is deployed on the opposite discharge surfaces (see Fig. 12). These charge distributions establish a field (reaction field), E_q, which is opposite to the local field that drove the PD, therefore the local field is reduced to a residual value, E_{res}, which depends on the resistivity of the discharge channel, and the PD ends.
Thus, the local field is given by the algebraic sum of two components, i.e. the applied field reduced

to the defect and the reaction field due to the local space or surface charge that have been left by previous PD activity, as expressed by (1.11).

$$E_i = f \cdot E_0 + E_q \tag{1.11}$$

At the ending instant of a PD event, the local field collapses in the defect and the equilibrium relation for the electric field is that provided by equation (1.12).

$$f \cdot E_0 - E_{res} = E_q \tag{1.12}$$

This mechanism is exemplified by figures 1.12, 1.13 and 1.14.

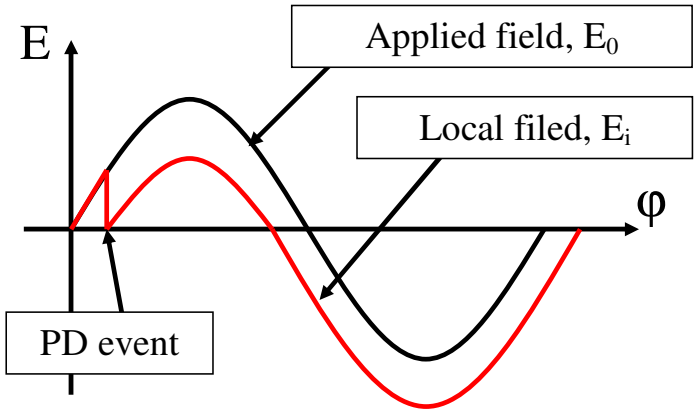


Fig. 1.12: *representation of a PD event in a phase vs. field diagram.*

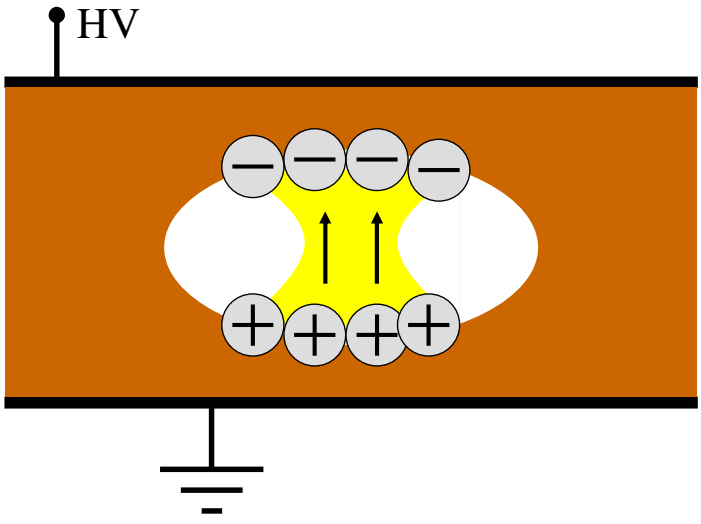


Fig. 1.13: *representation of the charge distributions deployed by the PD of Fig. 12.*

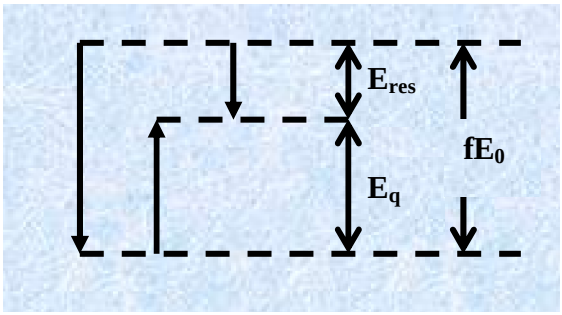


Fig. 1.14: *components of the local field at the time the PD of Fig. 1.12 ends (equilibrium equation).*

When a PD originates on a dielectric surface (cathodic discharge surface), the discharge channel propagates both toward the anode, (gap component of the discharge, in the direction of the field) and radially (surface component of the discharge, in the direction perpendicular to the field). The surface component of the discharge channel capacitively provides the current feed for the gap discharge, [9]. On the contrary, if the discharge originates on a conductive discharge surface, it propagates only toward the anode (gap component only).

As regards the charge distributions deployed by the PD on the (insulating) discharge surfaces, they have a finite lifetime, during which they constitute a memory of the PD event (*memory effect*) in two ways:

- They cause a phase shift and an offset of the local field with respect to the applied one, as expressed by (1.11). Such a phase shift remains as long as the deployed charge distributions are present on the discharge surface. An example of this phenomenon is shown by Fig. 1.12. The maximum phase shift is $|(\Delta\varphi)_{max}| = 90^o$, although this is a theoretical limit, [9].
- They are sources for initial electrons for the successive PD events. In fact, the deployed electrons may be detrapped by the field from potential traps.

The charges deployed by the PD may decay in different ways. In the case of a void, two mechanisms are prevailing, i.e. diffusion in the bulk material and recombination by drift on the void surfaces, as represented in figure 1.15.

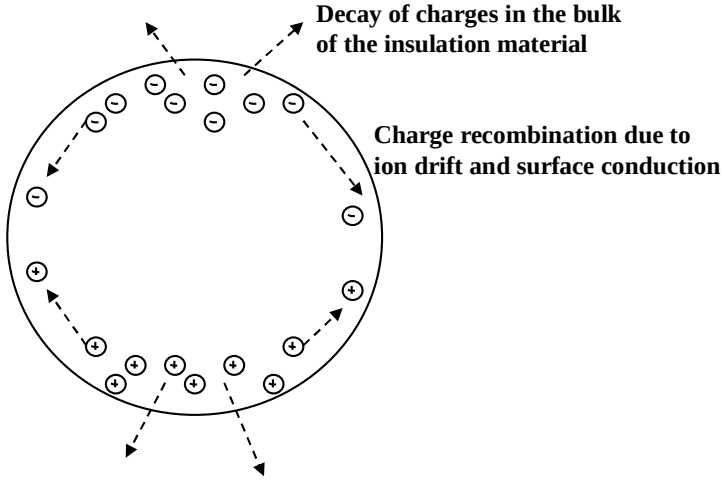


Fig. 1.15: *mechanisms of decay of the charge deployed by PD.*

In general, it can be assumed that the contribution to the local field due to the deployed charge distribution, E_q, varies with time, for effect of decay, with a law of the kind expressed in equation (1.13),

$$E_q(t) = E_q\left(t_{0^+}\right) \cdot \exp\left(-\frac{t}{\tau}\right) \tag{1.13}$$

where t_{o+} represents the instant the PD ends and τ represents a time constant for charge decay.

As regards the charge decay mechanisms, the diffusion in the bulk material is influenced by the nature of the discharge surface, which depends on the degradation stage. Recombination by drift on the void surfaces is influenced by the surface conductivity, which, in turn, depends on the degradation stage. In this light, it is remarkable that the PD activity may cause the formation of conductive paths on the discharge surfaces (due to carbonization). For more details on this topic reference can be made to [15-17].

One important consequence of the local field shift due to the memory effect consists of the possibility for PD events to anticipate the polarity reversal of the applied field (which is used as reference for the evaluation of the discharge phase). In fact, the reaction field E_q is opposite to the applied field E_0 at the time of the PD event, but, when E_0 changes its polarity, the two field contributions have the same sign. Therefore, before external field polarity reverses, there is a phase interval for which E_q and E_0 are opposite and E_q exceeds E_0 in module. In that situation, if $|E_q| - |E_0|$ is larger than the critical field, a leading PD may occur which has direction opposite to the applied field. Discharge anticipation phenomenon is described in figure 1.16, while figure 1.17 shows an example of PRPD pattern with phase anticipation.

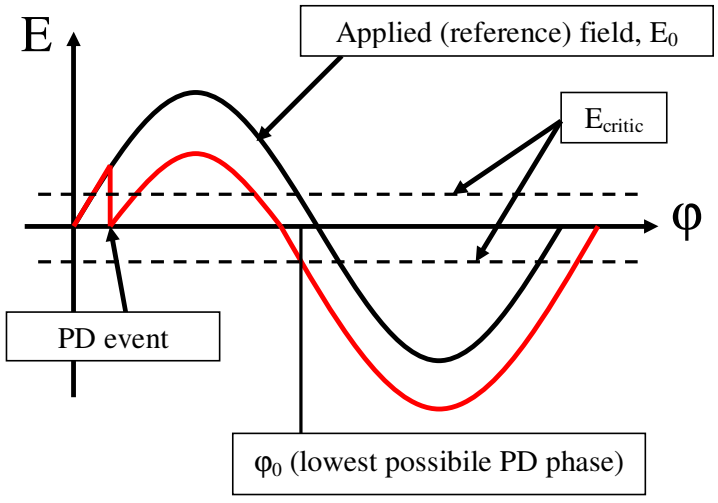


Fig. 1.16: *description of the phase anticipation phenomenon.*

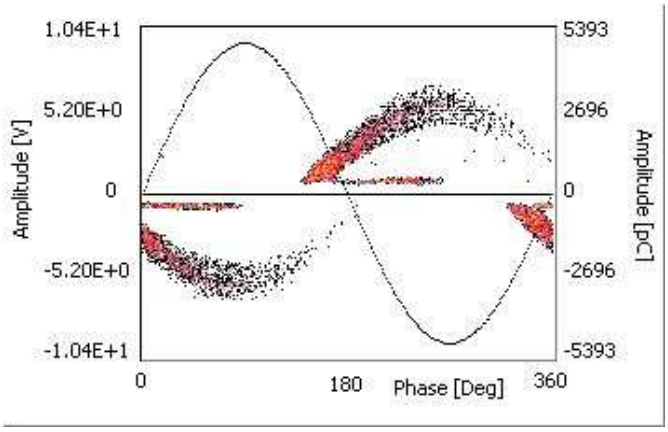


Fig. 1.17: *example of the effect of local field shift on PRPD pattern (phase anticipation).*

Hence, phase anticipation phenomenon is likely observable in the PRPD pattern, and indicates that the defect is enclosed in solid material (with at least one of the two discharge surfaces of insulating material) and that, roughly, the average time for the decay of deployed charges is larger than the half-period of the applied voltage.

Because the gap between the discharge surfaces is constant (does not depend on the applied field), the PD amplitude depends mainly on the development of the discharge channel on the surfaces, which, in turn, depends on the local field drop. Therefore, the amplitude of pulses pertinent to PD activities in voids may vary within a certain range, which tends to be wider for large voids (especially voids that are large in the direction orthogonal to the electric field allow significant variations in the discharge development on surfaces). On the contrary, the range of variation of PD amplitude, in general, is not much influenced by the applied field. In fact, the increment of the applied field, E_0, is usually compensated by an increment of the reaction field, E_q, at any PD event, according to (1.12), as long as the memory effect exists.
As regards the discharge rate, it increases with the applied field, because the smallest intertime is given by equation (1.14).

$$(\Delta t)_{\min} = \frac{E_{critic} - E_{res}}{\frac{dE_i(t)}{dt}} \qquad (1.14)$$

In fact, the smallest intertime occurs if a PD event takes place as soon as the local field reaches the critical value. Therefore, $(\Delta t)_{min}$ is given by the increment that the local field must cover between the two successive PD events divided by the rate at which the local field increases. Such a rate gets higher if the applied field rises (E_0 component), or if the decay time constant, τ, is reduced (E_d component).
In addition, the discharge rate may vary with the pressure and with the temperature of the gas which fills the defect. The consequence of this fact is sometimes a reduction of discharge rate after an

intense PD activity. In the medium period (e.g. hours or days) pressure and temperature variations may cause strong fluctuation of the PD rate.

Summarizing, a PD activity in a void is expected to exhibit the following trend:

- For increasing applied voltage → average amplitude slightly increases; amplitude dispersion is about constant; minimum PD phase decreases (anticipates the polarity reversal); discharge rate increases regularly.
- With time → average amplitude, amplitude dispersion and PD phases are about constant (until heavy degradation occurs); discharge rate fluctuates and tends to decrease (in the long period the PD activity may be extinguished at all).

Other configurations

The considerations reported about the PD activities in voids constitute a basis for the interpretation of the majority of the PD phenomena, by simply adapting the described concepts to the specific situations. However, there are PD activities for which the assumptions relevant to voids cannot be valid and that must be considered separately. These situations (configurations) are characterized by two basic aspects:

- The local field is strongly inhomogeneous in the discharge volume;
- The gap (distance between the discharge surfaces in the direction of the field) may vary dynamically with the applied field or the environmental conditions.

These situations may be exemplified by means of two typical configurations:

- Point to plane configuration;
- Strongly diverging field configuration.

The former configuration well represents, e.g., the so called “corona” PD activities, i.e. those PD phenomena which occur in correspondence of sharp metallic edges, surrounded by gaseous insulation. The latter configuration well represents, e.g., cable terminations, where the grounded shield is interrupted, or, also, stator bars (of generators), where they come out of the grounded iron core.

It is very important to remark that, when dealing with the basic nature of defects (and of PD activities), any classification scheme should not be considered as a strict one. In fact, PD activities and defects in electrical components usually exhibit a behavior (thus have a nature) which represent an intermediate state among those defined by the classification scheme. This consideration, which is in accordance with [8], shall be applied also to the present case, i.e. void, point to plain and strongly diverging field configurations.

To describe the discharge mechanism in a strongly inhomogeneous local field, the simple schematization of figure 18 will be considered.

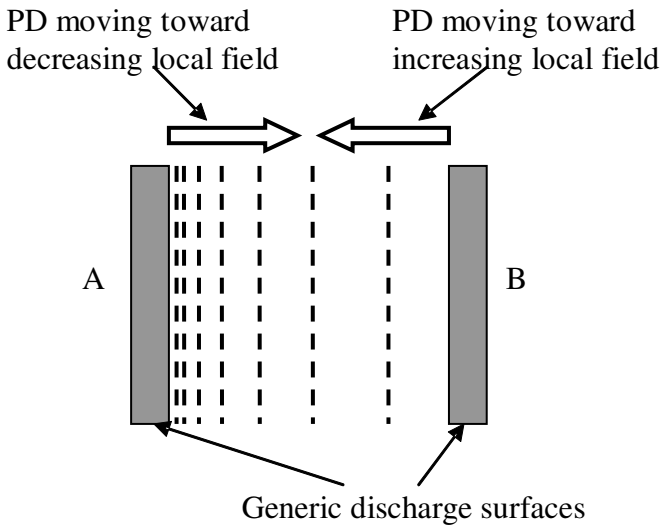


Fig. 18: *qualitative schematization of a defect where the local field is strongly inhomogeneous (broken lines represent equipotential field lines).*

Because the condition for a PD to occur is that an initial electron is present in the discharge volume near the cathode, when surface A (with reference to Fig. 1.18) is a cathode the probability of the PD event is much higher than in the case surface B is a cathode. In fact, especially if the discharge surfaces are conductive, the electric field plays a significant role in the first electron generation.
If the length of the discharge channel is variable, PD which move toward increasing values are most likely to have larger amplitude (because they cover long distances) with respect to those which move toward decreasing field values.
Given these premises, the two configurations mentioned earlier can be considered separately.

Point to plane configuration

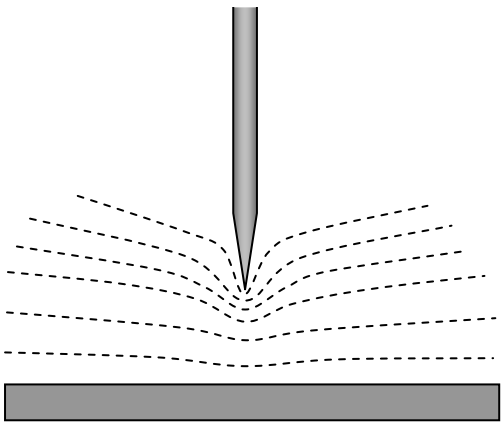


Fig. 1.19: *qualitative schematization of the point-to-plain configuration (broken lines represent equipotential field lines).*

In this configuration PD originate in correspondence of one of the two electrodes, but do not bridge them, because in that case they would constitute a breakdown. Therefore, the insulation is constituted by a certain volume of gas (air) opposite to the electrode that, at the time of the discharge, is a cathode.
In this defect configuration the field concentrates in correspondence of the point (sharp electrode). Therefore, assuming, e.g., that the plane electrode is grounded (and the point is connected to the HV), discharges incept at the second half-period of the applied voltage, i.e. when the point is a cathode. This behavior is due to the fact that the critical field is remarkably different for the two discharge surfaces. There will be a range of applied field values for which the plain electrode does not originate PD. Because the local field concentrates around the point within a very small volume, PD at the point will be all basically of the same length. Because no solid dielectric is present, the memory effect is null, therefore PD will occur in correspondence of the maximum of the applied field, which is proportional to the local field, i.e. when the generation rate of first electrons is maximum (thus the probability of PD events is maximum).
Increasing the applied field, the critic field value for the plane electrode will be reached. In this case discharges occur at both polarities. On the basis of the considerations reported about Fig. 18, PD from the plane electrode will be characterized by higher amplitude and lower repetition rate, with respect to those originated at the point.
These PD activities are characterized by the fact that the charge moved by PD is "deployed" in open air. In particular, when PD occur at the point, electrons are deployed in all directions (corona effect), thus constituting a shielding effect for the point. It must be observed that, when PD occur at the plane electrode, the electrons deployed in air give a significant contribute to the shielding of the point; indeed, an intense PD activity at the plane electrode tends to inhibit the PD activity at the point.

Summarizing, a PD activity in a point to plane configuration (Fig. 1.19) is expected to exhibit the following trend:

- For increasing applied voltage → the PD activity incepts at the point, with very low amplitude dispersion and phase centered at the maximum of the applied field. For higher field values, PD incept also at the plane electrode, with average amplitude much higher and amplitude dispersion slightly higher with respect to the PD at the point; average PD phase may be subjected to a slight decrease; the PD activity at the point may be inhibited by that at the plane.
- With time → average amplitude, amplitude dispersion and PD phases are about constant; discharge rate fluctuates considerably, but, as an average in the long period, does not decrease.

Strongly diverging field configuration

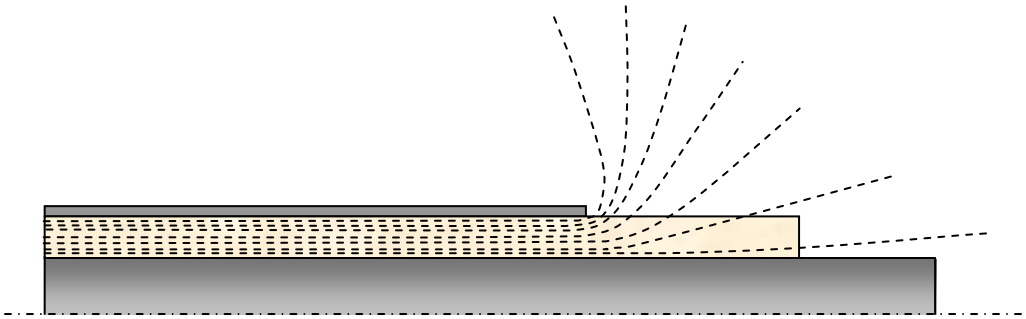


Fig. 1.20: *qualitative schematization of the strongly diverging field configuration (broken lines represent equipotential field lines).*

In this defect configuration one discharge surface is constituted by a conductor (in Fig. 1.20 is the electrode on the top), while the other discharge surface is constituted by insulating material. The characteristic peculiar of this kind of defect, in addition to the inhomogeneous field, is that the two discharge surfaces are aligned (along any given section plane), therefore the gap (i.e. the distance between the discharge surfaces in the direction of the electric field) is very small. The local field concentrates in a small volume, near the conductive discharge surface, but, moving away from that point, it decreases less abruptly than in the case of the point to plane configuration.
This kind of defect is characterized by a relatively low value of the critical field associated to the shortest PD gap. Indeed, a large number of small PD occur, especially when the conductive discharge surface is a cathode. The higher is the applied field, the longer the discharge channel can get, because the gap does not have a definite upper limit. Therefore, especially when the dielectric discharge surface is a cathode, some PD will occur with very large amplitude, while the majority of the PD still have small amplitude. In fact, the probability of occurrence of small-length PD is much higher than that of large-length PD, at any field value. For high field values, the PD activity at the insulating surface tends to inhibit the activity at the conductive surface, because of the shielding effect of large amplitude PD, which deploy electrons at the insulation surface in proximity of the conductive electrode.
Memory effect is possible, but the average time of charge decay is generally lower than the half period, because, since discharge surfaces are aligned, charge recombination for ion drift is very intense. In this light, it can be observed that the mobility of charge carriers in the PD activity may vary considerably with the nature of the materials (e.g. if the electrode in contact with the defect is semi-conductive).

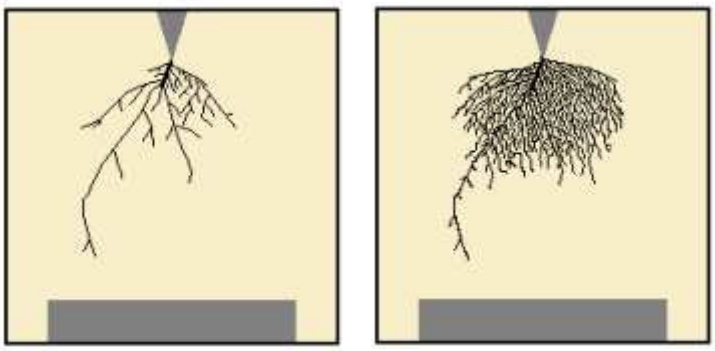
Summarizing, a PD activity in a strongly diverging field configuration (Fig. 1.20) is expected to exhibit the following trend:

- For increasing applied voltage → average amplitude considerably increases; amplitude dispersion increases, too (and, in general, is particularly high); minimum PD phase slightly decreases; discharge rate increases considerably.

- With time → average amplitude tends to decrease, amplitude dispersion and PD phases are about constant (until heavy degradation occurs); discharge rate tends to decrease (sometimes, in the long period the PD activity may be extinguished at all).

Electrical tree growth

The electrical tree is a pre-breakdown channel which develops in solid dielectrics. Electrical treeing occurs in locations where defects are present in the insulation and its formation is usually explained [12-24] as follows. Injection of highly energetic electrons into the dielectric takes place in the locations where high field concentrations are present. These electrons disrupt the bonds in the polymer chains, causing the formation of low-density regions in the material. In these low-density regions local breakdown phenomena occur, which result in partial discharges, giving rise to a tree channel. Hence, charges are moved and the high electric field is transferred to the top of the channel. Successively, degradation by high-energy electrons takes place near the top of the channel, forming new low-density regions. Thus, the tree grows step-wise and, because the formation of channels has a stochastic character, the geometry of the channels themselves recalls that of tree branches. It is possible that an electrical tree is characterized by several branches, which have a common root, in correspondence of the original defect. Examples of electrical tree geometry are reported in figure 1.21.



(A) (B)

Fig. 1.21: *examples of electrical tree geometry, (A) branch-like, (B) bush-like.*

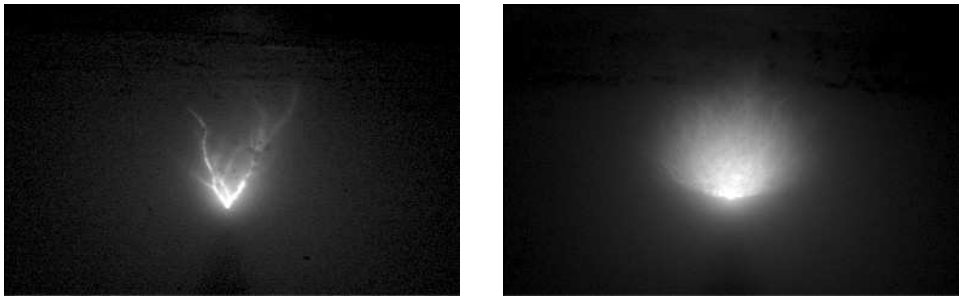
In the following, a list of insulation defects where electrical tree may originate is reported:

- sharp edges, e.g. protrusions or inclusions;
- voids at advanced degradation stage;
- interfaces between dielectrics (especially in the presence of moisture);
- non-conductive inclusions.

Interfaces between dielectrics and non-conductive inclusions are particularly dangerous (also from a

diagnostic point of view), because they usually activate tree formation, together with the correspondent PD activity, after a long time since electrical stress application. Furthermore, once activated, usually they take the insulation to failure in a very short period. This behavior could be explained considering that these defects are associated to large space charge distributions (trapped at insulation interfaces) which cover a large portion of the insulation between electrodes, but are characterized by an almost absolute absence of air (void). Therefore, the time lag for PD inception is very high and the average discharge rate is quite low.

Electrical tree growth may give rive to different geometries, depending on the insulation and defect nature and on the electric field. In particular, two tree geometries are distinguished: branch-like (Fig. 1.21A) and bush-like (Fig. 1.21B). Pictures of the PD activity associated to these electrical trees are reported in figure 18.



(A) (B)

Fig. 1.22: *PD activities associated to trees of different geometry, (A) branch-like, (B) bush-like.*

Bush-like geometry occurs at high values of the applied field. In this case, electronic injection process is very intense, thus providing the formation of a large number of tree channels in all directions (starting form the defect, i.e. the tree root). Hence, a shielding effect is established by the tree itself, which contributes to a reduction of the local field (at the top of tree channels). Indeed, this kind of tree takes more time to cause the final breakdown, with respect to branch-like trees. Branch-like trees usually occur at relatively low field values. They are characterized by a small number of branches, which however receive more constant feed, being the shielding effect less intense. Therefore, branch-like trees usually take the insulation to failure in a time that is relatively small, with respect to bush-like trees. Clearly, any geometry is possible which is intermediate between bush-like and branch-like.

Some characteristics, related to the PD activity, are in common to the majority of electrical trees.

- Multiple discharge sites (within the same defect);
- Charge is moved by steps, along different channels;
- An electrical tree is associable to multiple values for the critical field;
- The tree growth is associated to field shielding effects, therefore the PD activity is likely to be subjected to heavy fluctuations.

References section 1

[1] IEC 60270, Partial Discharge Measurements, 3rd edition, March 2001.
[2] CIGRE Report TF 15.11/33.03.02, Knowledge-based rules for partial discharge diagnosis in service, 2000.
[3] G. C. Stone, “Partial Discharge Part XXV: Calibration of PD measurements for motor and generator windings – why it can’t be done”, IEEE Electrical Insulation Magazine, Vol. 14, n. 1, pp. 9-12, February 1998.
[4] A. Contin, A. Cavallini, G. C. Montanari, G. Pasini, F. Puletti, "Digital Detection and Fuzzy Classification of Partial Discharge Signals”, IEEE Trans. on Dielectrics and Electrical Insulation, Vol.9, N.3, pp.335-348, June 2002.
[5] A. Contin, A. Cavallini, G.C. Montanari, G. Pasini, F. Puletti, "Artificial Intelligence Methodology for Separation and Classification of Partial Discharge Signals", IEEE CEIDP, pp. 522-526, Victoria, Canada, October 2000.
[6] A. Cavallini, A. Contin, G. C. Montanari, F. Puletti, "Advanced PD Inference in On-Field Measurements. Part.1: Noise Rejection", IEEE Trans. on Dielectrics and Electrical Insulation, 2003.
[7] A. Cavallini, M. Conti, G. C. Montanari, “On-Field Partial Discharge Measurements: a Feasible Approach to Noise rejection and Defect Assessment”, INSUCON 2002, pp. 103-108, Berlino, June 2002.
[8] F.H.Kreuger, *Partial Discharge Detection in High-Voltage Equipment*, Butteworth, London 1989.
[9] L. Niemeyer “A Generalized Approach to Partial Discharge Modeling”, IEEE Trans. on Dielectrics and Electrical Insulation, Vol. 2, N° 4, pp. 685-697, August 1995.
[10] G. C. Montanari, A. Cavallini, G. Mazzanti, M. Conti, F. Ciani, S. Serra, "First electron availability and PD generation in insulation cavities", IEEE CEIDP, pp. , Albuquerque, USA, October 2003.
[11] F. Gutlefish and L. Niemeyer “Measurement and Simulation of PD in Epoxy Voids”, IEEE Trans on Dielectrics and Electrical Insulation, Vol. 2, N° 5, , October 1995.
[12] P.H.F Morshuis, *Partial Discharges Mechanisms*, Ed. Delft University Press, Delft, 1993.
[13] C. Heitz, “A general stochastic approach to partial discharges”, IEEE ICSD, pp. 139-144, Vasteras, Sweden, June 22-25, 1998.
[14] G. C. Montanari, A. Cavallini, G. Mazzanti, M. Conti, F. Ciani, S. Serra, "First electron availability and PD generation in insulation cavities", IEEE CEIDP, pp. , Albuquerque, USA, October 2003.
[15] K. Wu, Y. Suzuoky, “Effect of discharge area on PD patterns in voids”, IEEE CEIDP, pp. 227-230, 1999.

[16] T. Mizutani, T. Kondo, K. Nakao, “Change in partial discharge properties of a void in LDPE”, IEEE CEIDP, pp. 257-260, 1999.
[17] A. Cavallini, F. Ciani, M. Conti, P. F. H. Morshuis, G. C. Montanari, "Modeling memory phenomena for partial discharge process in insulation cavities", IEEE CEIDP, pp. , Albuquerque, USA, October 2003.
[18] R. Bartnikas and E.J. McMahon “Engineering Dielectrics Vol. I Corona Measurements and Interpretation”, American Society for Testing Materials, Philadelphia, 1979.
[19] R. Bartnikas “Some Observations on the Character of Corona Discharges in Short Gap Spaces”, IEEE Transactions on Electrical Insulation, Vol. 6, pp 63-75, 1971.
[20] B.A. Fruth and D.W. Gross “Modelling of Steamer Discharges between Insulating and Conducting Surfaces”, Conference Proceedings of the 1995 5th Intl. Conf. On Conduction Breakdown in Solid Dielectrics, Leicester, England, 1995.
[21] Yu.V. Shcherbakov, A.V. Shilova, S. Syssoev “The Near-Surface Evolution of Streamer Discharges” Conference on Electrical Insulation and Dielectric Phenomena, pp 662-665, 1999
[22] L. A. Dissado, “Understanding electrical trees in solids: experiment to theory”, IEEE ICSD, pp. 15-26, Eindhoven, the Nederlends, June, 2001.
[23] L. A. Dissado, J. C. Fothergill, *Electrical degradation and breakdown in polymers*, Ed. Peter Peregrinus Ltd., London, 1992.
[24] J.V. Champion, S.J. Dodds, “Modelling partial discharges in electrical trees”, IEEE ICSD, pp. 291-294, June 1998, Leicester.

Section 2

Section 2 describes methods to process acquired data and defines specific quantities to be used as identification markers. It also describes an identification strategy, and basic concepts about the construction of diagnostic databases. Finally, it covers specific identification issues, providing examples and some interpretation rules based on the selected identification markers.

In this section:

- Discharge distributions
- Filtering techniques
- Derived quantities (identification markers)
- Categories of test objects
- Definition of an identification strategy
- Development of a diagnostic database
- Experimental activity disclosure
- First level – reference PD activities and results
- Second level – reference PD activities and results
- Third level – reference PD activities and results
- References

Discharge distributions

In the first section of this thesis it was underlined that there are three main quantities, associated to a PD acquisition, which are suited for defect identification process, i.e.

- PD amplitude;
- PD phase of occurrence with respect to the applied voltage;
- PD time of occurrence.

In addition, it was affirmed that these quantities are significant only if they are relevant to a set of pulses generated by a single PD source (defect). Therefore, from now on it will be assumed that the datasets analyzed for identification purpose consist of PD pulses relevant to a single source.

In order to identify PD phenomena, it is important (indispensable if the identification shall be automatic) to derive specific quantities from the acquired data, which are indicative of the nature of PD phenomena themselves (diagnostic markers). Furthermore, in the light of automating the whole identification process, these quantities should be derivable by means of procedures which can be performed by the machine in a reliable and robust way. To do this, the above mentioned acquired quantities may be arranged in many ways [1-13], e.g. by means of histograms, scatter plots, multi-dimensional plots, etc. In the following, the distributions which have been adopted in this thesis work to analyze (and visualize) the measured quantities are described. Figures 2.1-2.6 are pertinent to the same PD acquisition.

➔ PD amplitude distribution

PD amplitude distribution is represented by means of histograms, distinguishing between positive and negative discharge polarity[3]. For negative discharges, the absolute value of amplitude is considered.

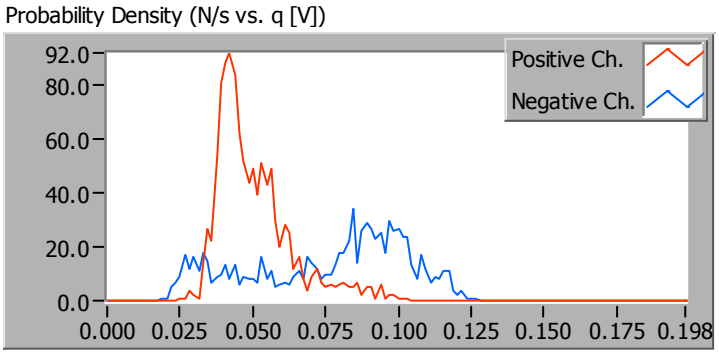


Fig. 2.1: *PD amplitude distribution represented by two histograms, relevant to the different PD polarities.*

[3] As anticipated in the first section, discharge polarity is determined by the sign of the local field, and is evaluated on the basis of PD phase distribution.

➔ PD phase distribution

PD phase distribution is represented by means of histograms, distinguishing between positive and negative discharge polarity.

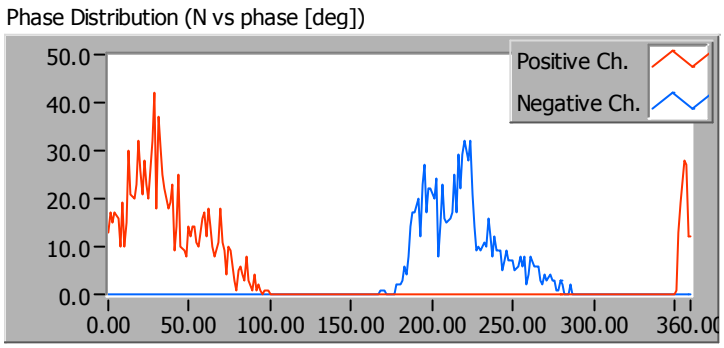


Fig. 2.2: *PD phase distribution represented by two histograms, relevant to the different PD polarities.*

It is important to remark that, for successive calculations, the phase, φ, of negative discharges (thus their distribution) is subjected to a 180 degrees shift, according to equation (2.1).

$$\begin{aligned} \varphi^{+} &= \varphi \quad \text{for positive discharges} \\ \varphi^{-} &= \varphi - 180 \quad \text{for negative discharges} \end{aligned} \tag{2.1}$$

This procedure has the purpose to make the quantities derived from the phase distributions of the two polarities coherent with each other.

➔ PD phase / amplitude distribution

Phase and amplitude distribution are also combined through the PRPD pattern (which was already described in the first section).

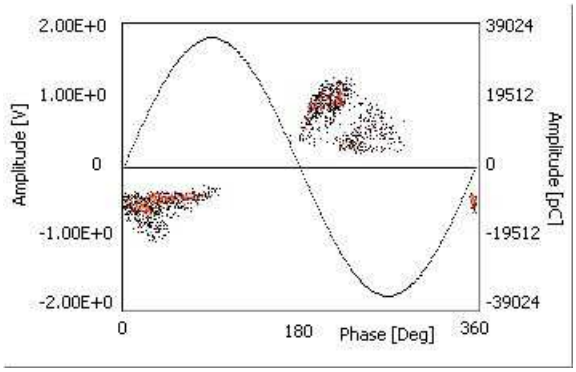


Fig. 2.3: *PRPD pattern, constituting a 2-dimesional histogram, where pulse number is expressed by color.*

➔ PD intertime distribution

Let N be the number of discharges acquired in a certain acquisition. Let the first PD event be associated to a reference time ($t_0 = 0$) and successive PD be associated to the time elapsed since t_0. Thus, a sequence of N-1 intertimes is derived, through equation (2.2).

$$(\Delta t)_k = t_{k+1} - t_k \ , \ \forall k = 1,..., N-1 \qquad (2.2)$$

The PD intertime distribution can be expressed through a single histogram, as shown in figure 2.4.

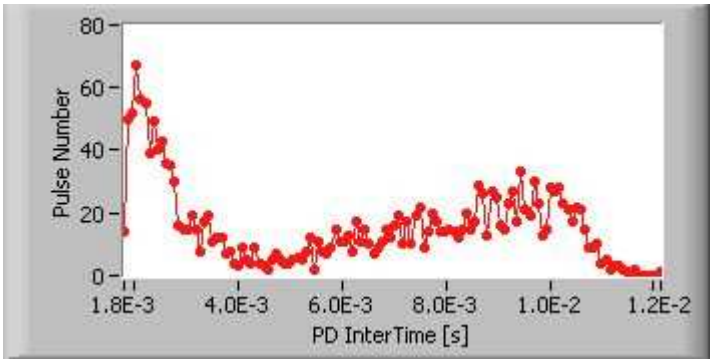


Fig. 2.4: *PD intertime distribution represented by a single histogram.*

In addition, it is useful to split the histogram of Fig. 2.4 in two sub-histograms, which will be addressed to as positive-follows and negative -follows histograms (for the sake of brevity, since it would be more correct to say "positive discharge –follows" and "negative discharge –follows"), through the following criterium: the k-th intertime belongs to the positive-follows histogram if the k+1-th PD has positive polarity, it belongs to the negative-follows histogram if the k+1-th PD has negative polarity. This procedure is illustrated in figure 2.5.

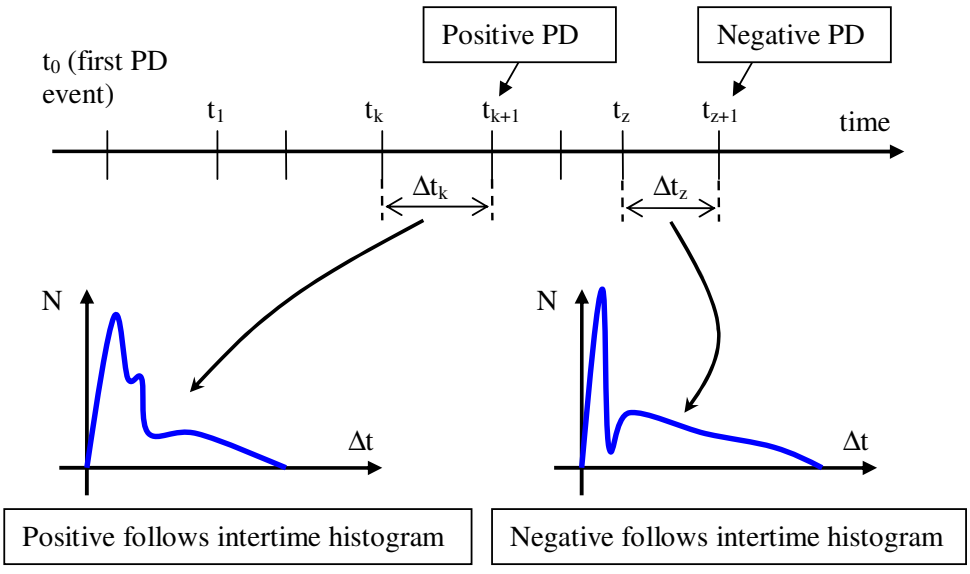


Fig. 2.5: *Schematic description of the calculation of "positive follows" and "negative follows" intertime histograms.*

Hence, these histograms are based on the time sequence distribution, but they also integrate information from the phase distribution, being dependent on discharge polarity. The positive-follows and negative-follows histogram derived from that of Fig. 2.4 are reported in figures 2.6 and 2.7, respectively.

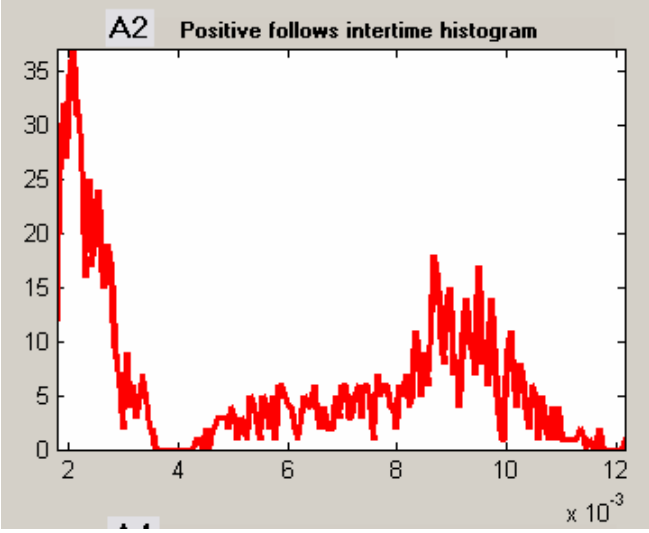


Fig. 2.6: *histogram representing positive-follows intertime distribution.*

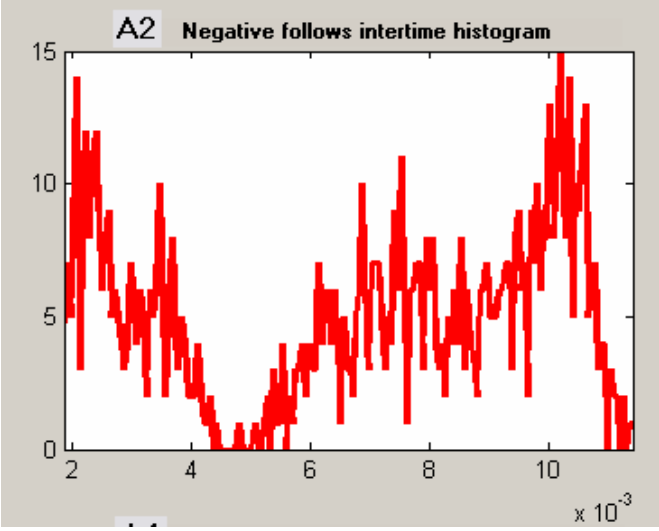


Fig. 2.7: *histogram representing negative-follows intertime distribution.*

Filtering techniques

Each one of the three basic PD distributions (amplitude, phase and intertime) may be affected by

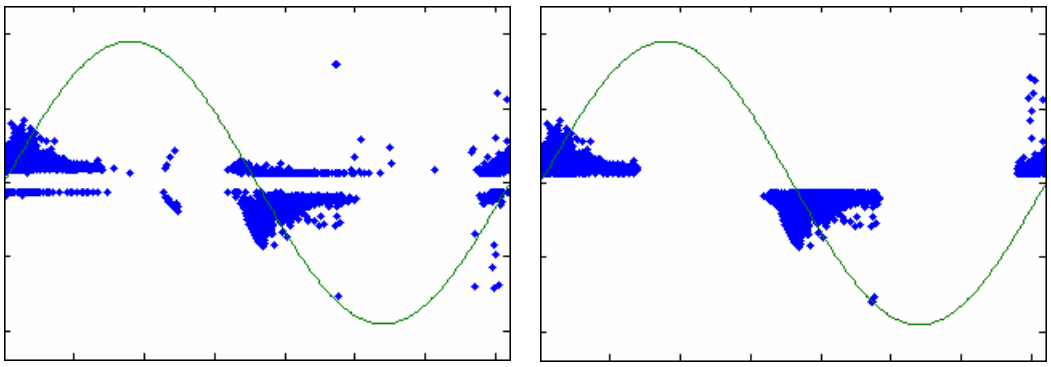
outliers. This circumstance occurs because neither the classification (and noise rejection) process nor the measuring system itself is perfect and because the operators who perform PD tests are likely to make mistakes (e.g. by setting the acquisition parameters, as full scale, trigger level, etc., in a not appropriate way). These "outliers" usually do not constitute a problem for a human expert, when he examines the PD distributions by sight. On the contrary, they do constitute a serious problem in the calculation of derived quantities (diagnostic markers) by the machine. Therefore, in the light of automating the identification process, techniques aimed at filtering the PD distributions play an essential role.

It is worth to underline that, for the application of these techniques, it is useful to handle the acquired data in the form of vectors (e.g. $\mathbf{q} = (q_1,\ldots, q_N)$ would be the PD amplitude vector, with N number of acquired PD) rather than in the form of grouped data (resultant from histogram construction).

Phase distribution

Phase distribution filtering is the most important and difficult among the filtering process, because the majority of the outliers occur as phase displacement of pulses and because this filtering process must account for the typology of measurement circuit and the phase anticipation[4] of the PD with respect to the zero crossing of the applied voltage (due to the memory effect).

As example, figures 2.8 and 2.9 can be considered. Fig. 2.8 is relevant to an acquisition performed with a measurement circuit of direct type, where phase anticipation is present.



A B

Fig. 2.8: *example of amplitude vs. phase diagram, (A) before and (B) after the filtering process – measurement circuit of direct type.*

Fig. 2.9 is relevant to an acquisition performed with a measurement circuit of indirect type, where phase anticipation is not present.

[4] As discussed in the first section of this thesis, phase anticipation occurs when positive (negative) PD take place before the zero-crossing of the positive (negative) half-wave of the applied voltage.

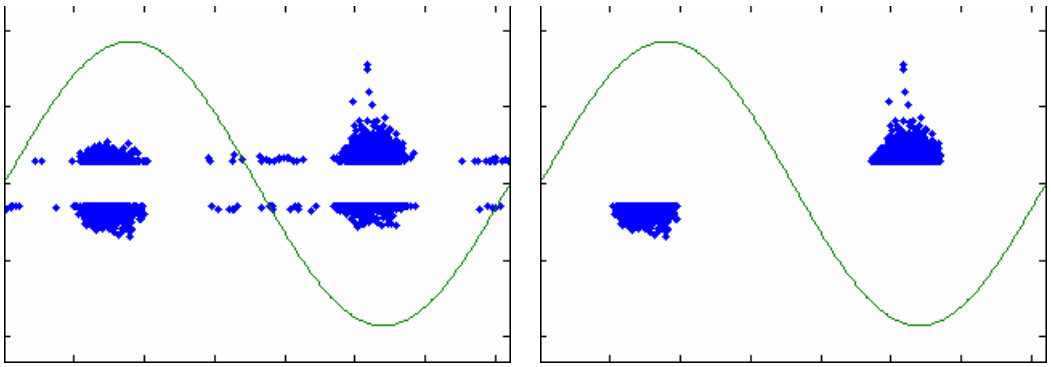


Fig. 2.9: *example of amplitude vs. phase diagram, (A) before and (B) after the filtering process – measurement circuit of indirect type.*

The technique used to filter the phase distribution can be qualitatively described as follows.
Because of memory effect (section 1, Figs. 1.16 and 1.17), it is possible to observe discharges driven by a positive local field from 90 degrees before to 90 degrees after the phases 0 and 180, respectively (anticipation and posticipation effect of the PD phases with respect to the applied voltage). Likewise for PD driven by the negative local field. Therefore, a PD distribution could be spread in the whole 360 degrees phase spectrum. Let the phase interval be divided in four sections, i.e. 0-90, 90-180, 180-270, 270-360, to make this description simpler. As regards the positive distribution, PD which fall in section 270-360 actually precede those which fall in section 0-90. In this light, phase interval sections can be numbered according to a precede-to-follow order, distinguishing the discharge polarity. To take into account the PD sign, distinction must be made between direct and indirect measurement circuits.
Figures 2.10 and 2.11 divide the phase-amplitude plane in eight sections, with phase ranging in [0, 360] and amplitude ranging in [- Fs (full scale), + Fs], and assign to each an indication to both PD polarity and phase ordering.

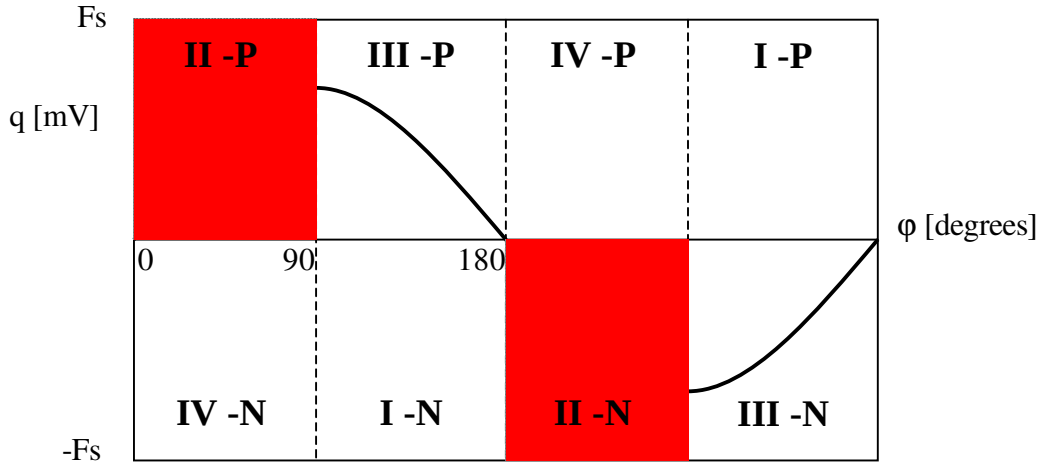


Fig. 2.10: *exemplification of polarity assignment and phase ordering in the φ-q plane, for a measurement circuit of direct type (P stands for positive, N for negative).*

In Fig. 2.10 those sections are put in evidence with red color where it is considered most probable to observe discharges of a given polarity (for measurement circuits of direct type).

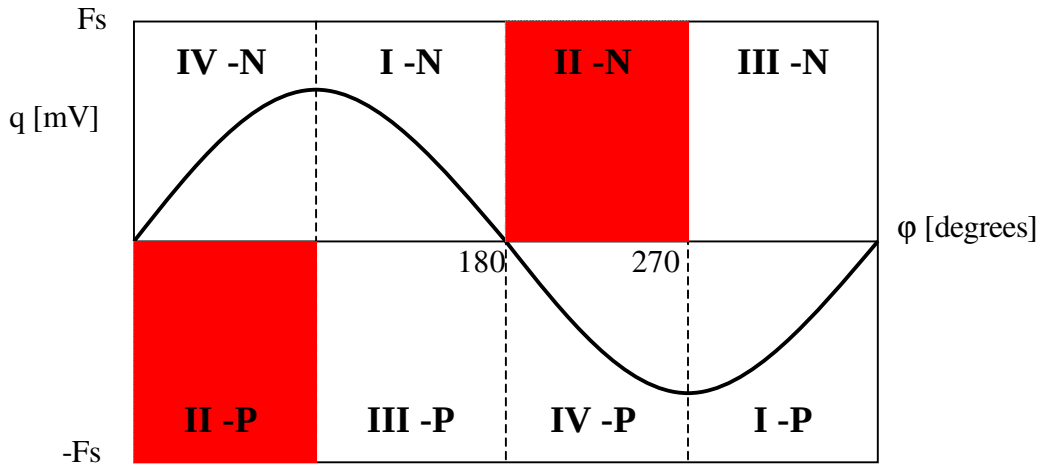


Fig. 2.11: *exemplification of polarity assignment and phase ordering in the φ-q plane, for a measurement circuit of direct type (P stands for positive, N for negative).*

Fig. 2.11 is analog to Fig. 2.10, but it refers to measurement circuits of indirect type.
Hence, it is assumed that positive PD are most likely to occur around a phase of 45 degrees, while negative PD are most likely to occur around a phase of 225 degrees. Thus, 45 and 225 are the reference phase values for positive and negative PD distributions, respectively. This is obviously an approximation, but, in the opinion of the author, it is acceptable in the majority of the situations.
Let $\boldsymbol{\varphi} = (\varphi_1,\ldots, \varphi_N)$ be the vector of PD phases (before filtering). On the basis of the considerations reported about Figs. 2.10 and 2.11, from vector $\boldsymbol{\varphi}$ a new vector, $\boldsymbol{\varphi}_{\mathbf{adapted}}$, is derived, according to the precede-follows ordering and according to the sign of the discharges and the circuit type. For example, if, for the k-th PD, $\mathbf{q}(k)$ is negative, $\boldsymbol{\varphi}(k)$ is 340 and the circuit is indirect, $\boldsymbol{\varphi}_{\mathbf{adapted}}(k)$ will be equal to 360 - $\boldsymbol{\varphi}(k)$, i.e. – 20 degrees (in Fig. 2.11 the PD would fall in the section labeled I-P).

On the basis of the assumption above (phase references) and of the "adapted" phase vectors, the filtering technique consists of grouping PD phases in two histograms (one for PD of positive sign, the other for PD of negative sign) characterized by non-constant channel (bin) width. In particular, given a minimum channel width, the width of the histogram channels will be as larger as closer the channel is to the reference phase. Successively, the filter operates by setting a threshold value on the frequency of occurrences in the histogram channels. Then, phase channels which pass this check are divided in groups and the mean squared distance from each group to the phase reference value is calculated. Finally, the group is selected which is "closest" to the reference.
In this way, for each polarity two phase values are calculated, i.e. the minimum and the maximum phase of the selected group, thus deriving a phase interval for each polarity. Usually, it is useful to discard the PD pulses which are outside the phase interval correspondent to their sign (according to

the circuit type) and to keep, but with opposite sign, those PD which fall in the selected phase interval but have wrong sign. This procedure is illustrated in Fig. 2.12.

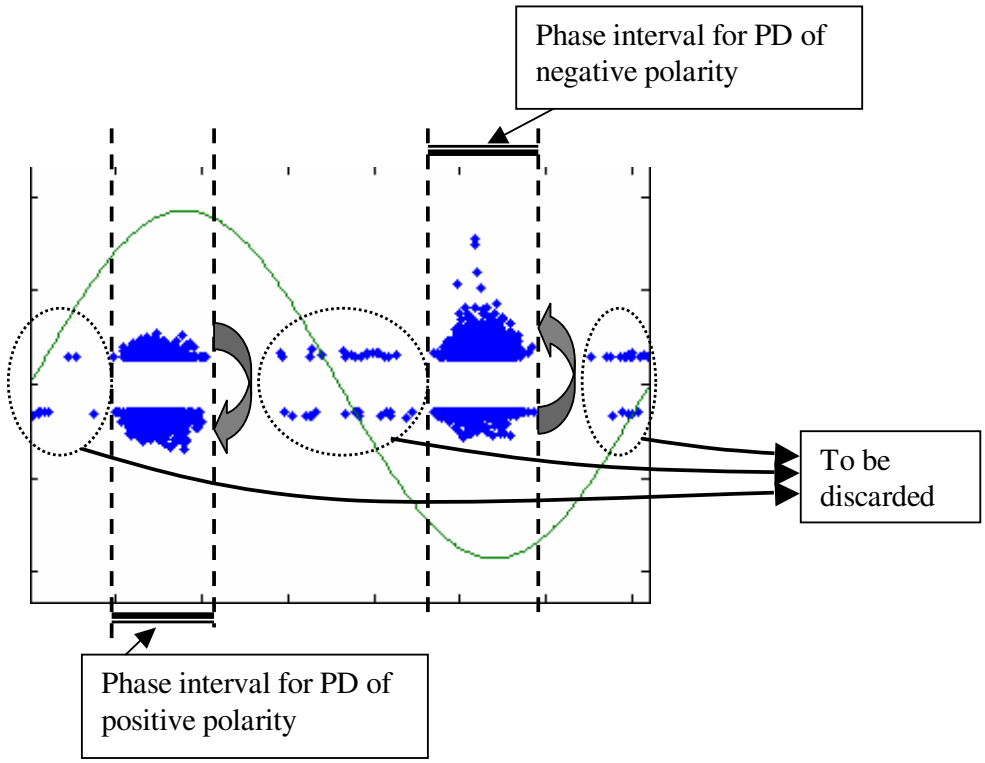


Fig. 2.12: *example of the action taken by the filter in the case of Fig. 2.9 (indirect circuit type).*

Amplitude distribution

The technique used to filter the amplitude distribution is very simple. Let **q** = (q_1,…, q_N) be the vector of pulse amplitudes. The filter operates on the basis of equation (2.3), and requires setting of just one constant, k.

$$q_{filtered} = \{q(i) \mid q(i) - \mu < k \cdot \sigma \quad i = 1,..., N\} \tag{2.3}$$

In (2.3), μ and σ represent the mean value and the variance of the amplitude distribution, respectively.

The result of the application of this filtering process to the PD pattern of Fig. 2.8B is reported in figure 2.13.

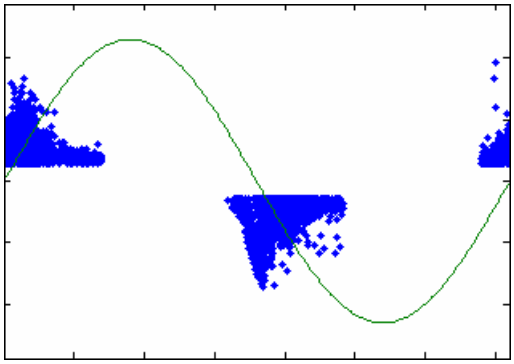


Fig. 2.13: *example of the result of the amplitude filter, in the case of Fig. 8B.*

It is very important to observe that these filtering processes must be run is series. In other words, the action taken by one filtering process on a specific PD quantity distribution must be applied to all of the other distributions as well. For example, if the phase filtering discards the k-th and the h-th elements of the phase vector, the correspondent elements of the vectors of amplitude and time of occurrence (and others, if available, as equivalent time and bandwidth) must be discarded, too.

Intertime distribution

Intertime distribution is filtered with the same technique as PD amplitude distribution. Such a filtering shall be run on the whole intertime vector. As regards positive-follows and negative-follows distributions, they must be derived after filtering process on phase amplitude is performed, because they require each PD to be provided with an information about its polarity.

Derived quantities (identification markers)

In this paragraph a list of quantities will be reported; the majority of them have been used in this thesis work as identification markers, others are reported for the sake of completeness. These quantities were derived from the distributions previously described and constitute the link between the measured data and the discharge mechanism, being aimed at the characterization of PD phenomena. Indeed, the selection of the identification markers should be strictly related to the description that the human expert is able to provide for PD phenomena. This is probably the reason why much effort was spent in this field, with different results [14-22]. The quantities reported in the following can be classified according to two different criteria.

Classification of the identification markers with respect to the type of PD distribution from which they are derived:

- Amplitude distribution-based quantities;
- Phase distribution-based quantities;

- Intertime distribution-based quantities.

Because each PD distribution can be split in two distributions with respect to polarity, there are quantities which are calculated on the basis of the whole distribution (global quantities), others on the basis of a single polarity distribution (single-polarity quantities) and other quantities which put in relation the two distributions (of different polarity) of the same typology (combined quantities); finally, there are quantities which combine information coming from distributions of different types (complex quantities).
Classification of the identification markers with respect to the kind of interaction between distributions of different polarity in their calculation:

- Global quantities;
- Single-polarity quantities;
- Combined quantities;
- Complex quantities.

List of identification markers.

1)	FiminPos	phase distribution-based;	single-polarity
2)	FiminNeg	phase distribution-based;	single-polarity
3)	FimeanPos	phase distribution-based;	single-polarity
4)	FimeanNeg	phase distribution-based;	single-polarity
5)	DFiPos	phase distribution-based;	single-polarity
6)	DFiNeg	phase distribution-based;	single-polarity
7)	Sk_ampPos	amplitude distribution-based;	single-polarity
8)	Sk_ampNeg	amplitude distribution-based;	single-polarity
9)	betaPos	amplitude distribution-based;	single-polarity
10)	betaNeg	amplitude distribution-based;	single-polarity
11)	alfaPos	amplitude distribution-based;	single-polarity
12)	alfaNeg	amplitude distribution-based;	single-polarity
13)	LdsPos	amplitude distribution-based;	single-polarity
14)	LdsNeg	amplitude distribution-based;	single-polarity
15)	Sk_phPos	phase distribution-based;	single-polarity
16)	Sk_phNeg	phase distribution-based;	single-polarity
17)	NPos	based on any distribution;	single-polarity
18)	NNeg	based on any distribution;	single-polarity
19)	NQNPos	amplitude distribution-based;	single-polarity
20)	NQNNeg	amplitude distribution-based;	single-polarity
21)	QmPos	amplitude distribution-based;	single-polarity
22)	QmNeg	amplitude distribution-based;	single-polarity

23)	FimeanDPos	phase distribution-based;	complex
24)	FimeanDNeg	phase distribution-based;	complex
25)	QminPos	amplitude distribution-based;	single-polarity
26)	QminNeg	amplitude distribution-based;	single-polarity
27)	QmaxPos	amplitude distribution-based;	single-polarity
28)	QmaxNeg	amplitude distribution-based;	single-polarity
29)	QmeanPos	amplitude distribution-based;	single-polarity
30)	QmeanNeg	amplitude distribution-based;	single-polarity
31)	ND	based on any distribution;	relational
32)	Nw	intertime distribution-based;	global
33)	P1	intertime distribution-based;	global
34)	P2	intertime distribution-based;	global
35)	P3	intertime distribution-based;	global
36)	P4	intertime distribution-based;	global
37)	QD	amplitude distribution-based;	combined
38)	FD	phase distribution-based;	combined
39)	stdQD	amplitude distribution-based;	combined
40)	ND2	based on any distribution;	combined
41)	ITmeanPfollows	intertime distribution-based;	single-polarity
42)	ITmeanNfollows	intertime distribution-based;	single-polarity
43)	ITD	intertime distribution-based;	combined
44)	P5	intertime distribution-based;	complex
45)	pF1Pos	phase distribution-based;	single-polarity
46)	pF1Neg	phase distribution-based;	single-polarity
47)	pF2Pos	phase distribution-based;	single-polarity
48)	pF2Neg	phase distribution-based;	single-polarity
49)	AgPos	amplitude distribution-based;	single-polarity
50)	AgNeg	amplitude distribution-based;	single-polarity
51)	FgPos	phase distribution-based;	single-polarity
52)	FgNeg	phase distribution-based;	single-polarity

In the following, the definitions of these quantities will be reported, together with a short account of the methods to calculate them.

Primarily, some general operators will be provided, which will be useful for quantities calculation.

Let $\mathbf{v}$ be a vector of N elements, $\mathbf{v} = (v_1, \ldots, v_N)$.

- *mean*

$$mean(v) = \frac{\sum_{k=1}^{N} v(k)}{N} \tag{2.4}$$

– *std* (standard deviation)

$$std(v) = \sqrt{\frac{\sum_{k=1}^{N}(v(k) - mean(v))^2}{N-1}} \tag{2.5}$$

– *skewness*

$$skewness(v) = \frac{\sum_{k=1}^{N}(v(k) - mean(v))^3}{(std(v))^3} \tag{2.6}$$

Definition – *normalizedratio*

Let x_1 and x_2 be two real positive numbers.

$$normalizedratio(x_1, x_2) = s \cdot \left(1 - \left(\frac{x_1}{x_2}\right)^{-s}\right) \tag{2.7}$$

where *s* is given by (2.8).

$$s = sign\left(\log\left(\frac{x_1}{x_2}\right)\right) \tag{2.8}$$

1 – 2) **FiminPos, FiminNeg** [degrees] Range of variation: [-90, 180]

Minimum value for the positive and negative PD phase distributions, respectively.
A graphical representation of these quantities is provided in figure 2.14.

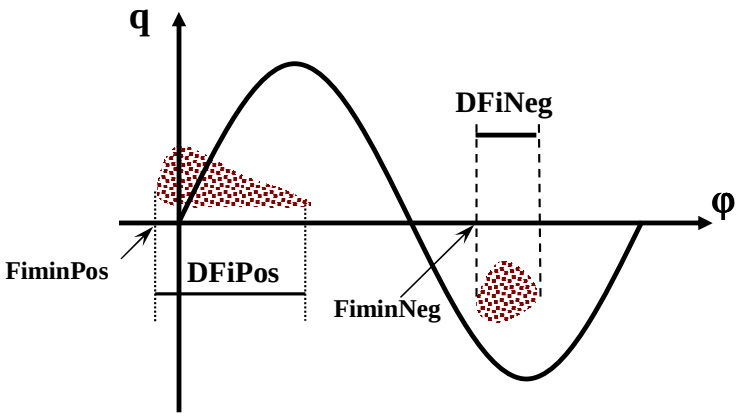


Fig. 2.14: *representation of quantities Fimin and Dfi*

3 – 4) **FimeanPos, FimeanNeg** [degrees] Range of variation: [0, 180]

Mean value for the positive and negative PD phase distributions, respectively. It is worth to observe that, in practical cases, it is useful to extend the range of variation to approximately [-20, 180].

5 – 6) **DFiPos, DFiNeg** [degrees] Range of variation: [0, 360]

Phase interval (i.e. difference between maximum and minimum phase) for the positive and negative PD phase distributions, respectively (Fig. 14).

7 – 8) **Sk_ampPos, Sk_ampNeg** (dimensionless) Range of variation:]-∞, +∞ [

Skewness value for the positive and negative PD amplitude distributions, respectively.

9 – 10) **betaPos, betaNeg** (dimensionless) Range of variation:]0, +∞[

Shape factor of the Weibull function which fits the positive and negative PD amplitude distributions, respectively.

11 – 12) **alfaPos, alfaNeg** [mV] or [pC] Range of variation:]0, +∞[

Scale factor of the Weibull function which fits the positive and negative PD amplitude distributions, respectively.

13 – 14) **LdsPos, LdsNeg** (dimensionless) Range of variation: [0, 0.5]

Significance level, which estimates the degree of fitness of the Weibull function with respect to the positive and negative PD amplitude distributions, respectively.

15 – 16) **Sk_phPos, Sk_phNeg** (dimensionless) Range of variation:]-∞, +∞ [

Skewness value for the positive and negative PD phase distributions, respectively.

17 -18) **NPos, NNeg** (dimensionless) Range of variation: [0, +∞[

Number of acquired positive and negative PD pulses.

19 – 20) **NQNPos, NQNNeg** [mV] or [pC] Range of variation:]0, +∞[

This quantity is defined in [23]. Its meaning is illustrated in figure 15, and it can be calculated as

indicated in (2.9),

$$NQN = \Delta q \cdot \sum N_i \qquad (2.9)$$

where Δq is the width of the bins (supposed all of the same width) used for the PD amplitude histogram (of a given polarity) and N_i is given by (2.10),

$$N_i = \log 10\left(\frac{n_i}{T}\right) \qquad (2.10)$$

where n_i is the counter of pulses in the i-th amplitude channel and T is the duration of the PD acquisition (acquisition time).

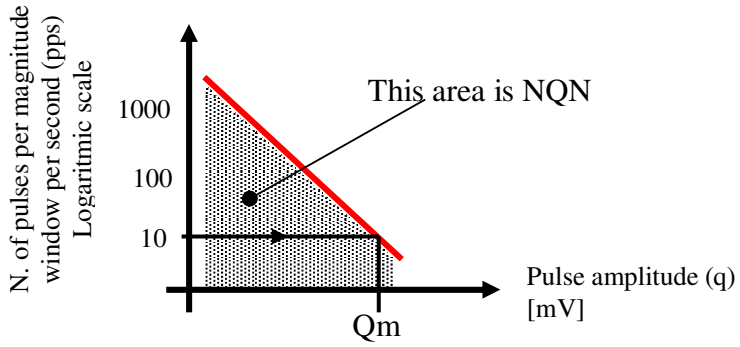


Fig. 2.15: *representation of quantities NQN and Qm*

21 – 22) **QmPos, QmNeg** [mV] or [pC] Range of variation:]0, +∞[

Discharge amplitude that corresponds to value 10 discharges per second in the histogram of Fig. 2.15.

23 – 24) **FimeanDPos, FimeanDNeg** [degrees] Range of variation: [-180, 180]

Let Fi_L and Fi_H be, for each PD polarity distribution, the mean phase value of "high discharges" and "low discharges", respectively, as indicated in figure 16. Let FiLPos and FiHPos be the quantities described above (Fi_L and Fi_H) for the positive polarity distribution and FiLNeg and FiHNeg the ones relevant to the negative polarity distribution. FimeanDPos and FimeanDNeg result from equations (2.11) and (2.12), respectively.

$$\text{FimeanDPos} = \text{FiLPos} - \text{FiHPos} \qquad (2.11)$$

$$\text{FimeanDNeg} = \text{FiLNeg} - \text{FiHNeg} \qquad (2.12)$$

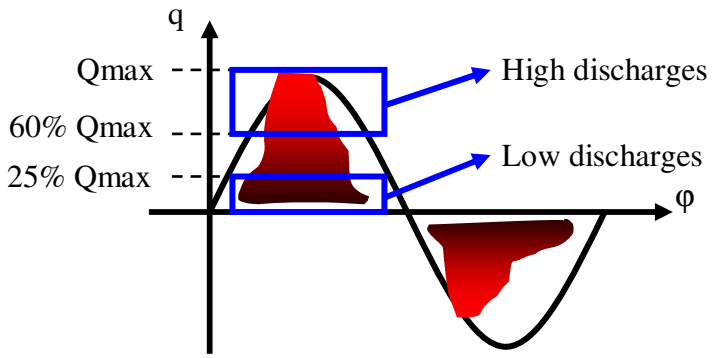


Fig. 2.16: *description of quantities FimeanDPos and FimeanDNeg.*

25 – 26) **QminPos, QminNeg** [mV] or [pC] Range of variation:]0, +∞[

Minimum value for the positive and negative PD amplitude distributions, respectively.

27 – 28) **QmaxPos, QmaxNeg** [mV] or [pC] Range of variation:]0, +∞[

Maximum value for the positive and negative PD amplitude distributions, respectively.

29 – 30) **QmeanPos, QmeanNeg** [mV] or [pC] Range of variation:]0, +∞[

Mean value for the positive and negative PD amplitude distributions, respectively.

31) **ND** (dimensionless) Range of variation: [0, 1]

Ratio between NPos and NNeg, normalized to [0, 1], according to (2.13)

$$ND = 1 - \frac{1}{1 + \frac{NPos}{NNeg}} \tag{2.13}$$

32) **Nw** (dimensionless) Range of variation: [0, +∞[

Number of discharges per period of the applied voltage.

33) **P1** [s] Range of variation:]0, +∞[

10% percentile of the intertime distribution.

34) **P2** [s] Range of variation:]0, +∞[

90% percentile of the intertime distribution.

35) **P3** (dimensionless) Range of variation: [0, 1]

With respect to the intertime distribution (histogram), it is the ratio between the maximum probability and the probability of the first histogram channel.

36) **P4** (dimensionless) Range of variation: [0, 1]

With respect to the intertime distribution, let *rT* be the ratio between the half period of the applied voltage (10 ms at 50 Hz) and the midpoint of the shortest time interval, $[t_1, t_2]$, that satisfies (2.14).

$$\int_{t_1}^{t_2} f(t)\,dt \geq 0.8 \tag{2.14}$$

P4 if then given by equation (2.15).

$$P4 = \min(rT, 1) \tag{2.15}$$

37) **QD** (dimensionless) Range of variation: [-1, 1]

QD is given by equation (2.16),

$$QD = \max\begin{pmatrix} abs\left(normalizedratio\left(\text{QmaxPos}, \text{QmaxNeg}\right)\right), \\ abs\left(normalizedratio\left(\text{QmeanPos}, \text{QmeanNeg}\right)\right) \end{pmatrix} \cdot S \tag{2.16}$$

where *S* is the sign of the "normalizedratio" which has the largest module.

38) **FD** [degrees] Range of variation: [-180, +180]

FD is given by equation (2.17).

$$\text{FD} = \text{FiminPos} - \text{FiminNeg} \tag{2.17}$$

39) **stdQD** (dimensionless) Range of variation: [-1, 1]

stdQD is given by equation (2.18).

$$stdQD = normalizedratio\left(stdQPos, stdQNeg\right) \tag{2.18}$$

where *stdQPos* and *stdQNeg* are the values of the standard deviation for positive and negative PD amplitude distribution, repectively.

40) **ND2** (dimensionless) Range of variation: [-1, 1]

ND2 is given by equation (2.19).

$$ND2 = normalizedratio\left(NPos, NNeg\right) \tag{2.19}$$

41 - 42) **ITmeanPfollows, ITmeanNfollows** [s] Range of variation:]0, +∞[

Mean value of the positive-follows intertime distribution and negative-follows intertime distribution, respectively.

43) **ITD** (dimensionless) Range of variation: [-1, 1]

ITD is given by equation (2.20).

$$ITD = normalizedratio\left(ITmeanPfollows, ITmeanNfollows\right) \tag{2.20}$$

44) **P5** (dimensionless) Range of variation: [-1, 1]

P5 is given by equation (2.21).

$$P5 = ITD \cdot ND2 \tag{2.21}$$

45 – 46) **pF1Pos, pF1Neg** (dimensionless) Range of variation: [0, 1]

Let FimodePos be the value of the PD phase distribution of positive polarity which corresponds to the highest probability (frequency of occurrences). pF1Pos is given by equation (2.22).

$$pF1Pos = \frac{\text{FimodePos-FiminPos}}{\text{DFiPos}} \tag{2.22}$$

Likewise for pF1Neg.

47 – 48) **pF2Pos, pF2Neg** (dimensionless) Range of variation: [0, 1]

Ratio between the mean probability (frequency of occurrences) and the maximum probability, in the PD phase distribution of positive and negative polarity, respectively.

49 – 50) **AgPos, AgNeg** (dimensionless) Range of variation: [0, 1]

Quantity Ag is aimed at revealing the presence of humps in the PD amplitude distribution.
Let the cumulative function, F, of amplitude distribution (for a given discharge polarity) be determined through a histogram of M channels (let M be 90). Then, values of the cumulative probability which correspond to 1/3 and 2/3 of M are considered. In this way, probability intervals indicated in figure 2.17 as a, b and c are evaluated.

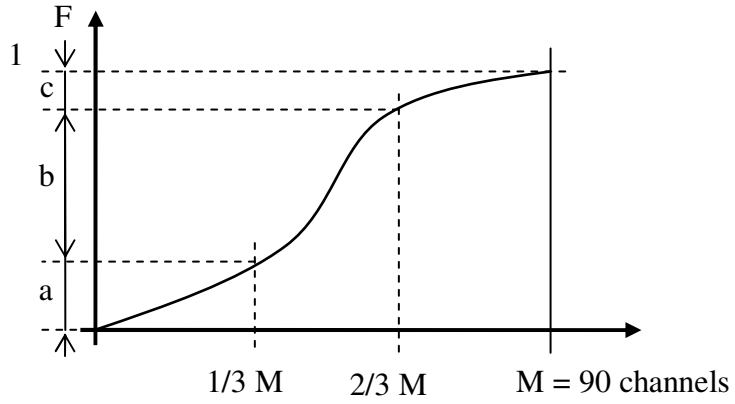


Fig. 2.17: *schematic description of the procedure to determine quantity Ag.*

Being r1 the ratio of b and a and r2 the ratio of c and b, two quantities, f(r1) and f(r2) are calculated, where f(r) is the piece-wise linear function described in figure 2.18 ($f(r_a) = 0$, $f(r_b) = 1$,.$r_a = 0.4$, $r_b = 1$).

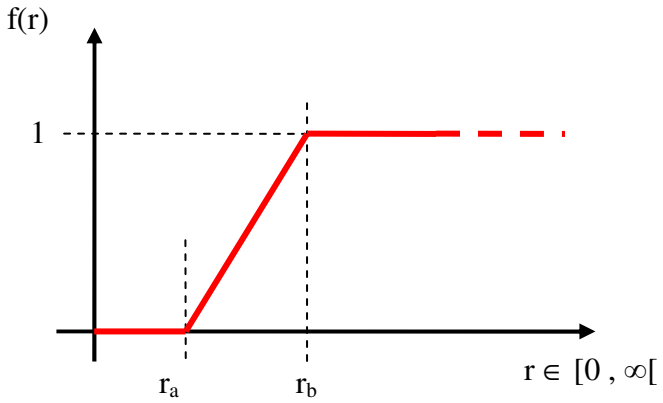


Fig. 2.18: *non linear function used for the calculation of quantity Ag.*

Ag is then calculated as the maximum value between f(r1) and f(r2).

51 – 52) **FgPos, FgNeg** (dimensionless) Range of variation: [1, +∞[, with Fg natural

Number of peaks of the PD phase distribution for positive and negative polarity, respectively. It is calculated through a polynomial fitting procedure applied to the cumulative distribution of the PD phases (distinguishing discharge polarity).

It is not expectable that any of these identification markers is independent from either time or applied voltage. Indeed, PD activities are subjected to modifications and identification markers shall change accordingly, as long as they reflect and characterize the PD activity[5]. Hence, if the result of the identification has to be invariant with respect to time and applied voltage, the following circumstances must be verified.

- The trend of the identification markers with respect to the applied voltage is somehow predicted, in such a way that the identification algorithm accounts for those modifications.
- The trend of the identification markers with respect to time in the short period, i.e. the duration of the PD testing, is either negligible, or anticipated by the identification algorithm.
- The trend of the identification markers with respect to time in the long period, i.e. a period which involves actual degradation or, even, change of the defect nature, determines an appropriate change in the identification result.

When the identification algorithms are described, it will specified whether these circumstances are likely to be verified or not.

Categories of test objects

A large variety of measurement objects have been tested during this thesis work. Some measurements were gathered from test sessions performed at LIMAT (HV laboratory which belongs to the University of Bologna), other from the testing experience of TechImp company (which operates in the field of HV apparatus diagnosis). Indeed, it is very important to perform PD measurements both in laboratory and on the field. In fact, in order to learn how to interpret PD data automatically, known examples (models) are required, both by the human operator and by the machine. At the same time, to make the learned interpretation criteria actually applicable to HV equipments, testing examples are required, which are truly representative of the conditions encountered during on-line and off-line testing.
In this light, three kinds of measurement objects are distinguished.

[5] Clearly, depending on the specific situation considered, some quantities will result more significant and/or invariant with respect to others.

- Laboratory models. Specimens realized with the aim of providing specific insulation defects in known conditions. Measurements on laboratory models are likely to provide training examples to derive general concepts and to investigate the relation between measured data and physical phenomena. On the other side, they may be scarcely representative of HV apparatus on field, in general.
- Industrial models. They are constituted by HV equipments where artificial defects are realized, with the aim of reproducing the most common defects for a given equipment typology. Measurement on industrial models constitute a trade off with respect to the need to have representative data and, at the same time, known situations. Therefore, these data can be used for both training and testing, although with a remarkable degree of uncertainty.
- Industrial objects. This group is relevant to defects in industrial objects. They shall constitute the majority of the data overall available, but only in rare circumstances can be used for training, because, in general, the actual nature and harmfulness of these defects is unknown. However, they are essential to check the applicability of the results obtained on the basis the measurements performed on the other object typologies.

Definition of an identification strategy

PD interpretation is aimed at constituting a guide for operators in decision making, e.g. for condition based maintenance (CBM), quality control or on line monitoring. In this light, the information provided by PD analysis, to be useful, must be strictly related to the specific typology of apparatus under test. In fact, for the majority of the HV apparatus typologies, knowledge is available about what kind of insulation defects occur most frequently and which are the most harmful for that specific apparatus. Therefore, the ultimate goal of PD interpretation is to address a PD activity to a category which is specific of a given HV apparatus, thus being meaningful. Clearly, these categories are defined with respect to the whole apparatus (e.g., for a large generator, a possible specific defect category is constituted by the detachment of a stator bar from the iron core within the slot). In this light, two main problems arise.

The first problem consists of the variety of HV apparatus typologies, which would determine a huge number of specific categories of insulation defects.

The second problem consists of the fact that acquired PD pulses depend on the local conditions of the insulation defect (e.g. local field, nature of the discharge surfaces, etc.), which are not directly linked to the definitions of insulation defects proper of specific categories. In this light, general categories, i.e. categories defined with respect to the local conditions of the defect (e.g. defect for which the local field is perpendicular to the discharge surfaces), would be much more directly linked to the acquired PD data and could be applicable to different typologies of HV apparatus.

A valid identification strategy should be aimed at solving these two major problems. The

identification strategy proposed in this thesis is based on a multiple categorization of insulation defects, organized in a multilevel structure, being these levels relevant to the concepts of generality and specificity discussed above. Hence, the identification process is preliminarily aimed at providing general information about PD activity, in order to have an insight of the basic nature of the defect, independently on the kind of apparatus under test. Successively, more specific information is provided, although still independent on the kind of apparatus under test, in order to allow a link to specific situations for any apparatus typology (given that the various apparatus typologies cannot be all equally and sufficiently known). Finally, fully specific information are provided for those apparatus typologies for which an adequate knowledge is available.

The fact that the identification process is split into a variety of issues (identification tasks), with respect to generality and specificity of defect categories, is very useful also in consideration of the different kinds of measurement objects available. According to the measurement object classification described in the previous paragraph, it is remarkable that laboratory models are particularly suited for training when dealing with general defect categories. On the opposite side, industrial objects are particularly suited for training when dealing with apparatus-specific defect categories (as long as the expected identification result is known). A further advantage of this identification strategy is that the result of the general identification tasks can be used as input information for those identification tasks which are more specific.

Thus, identification is a complex process, consisting of steps and sub-steps, organized in a tree-like structure, where sub-steps can be seen as tree nodes, each one covering a specific task. In this way, since a machine shall execute the whole identification process automatically, several artificial intelligence techniques have been considered; in fact, the optimum solution could be a combination of different techniques. Moreover, the whole tree structure is handled by a fuzzy logic based system, which connects the different tree nodes. In such a way, each node (identification step) can be constructed using the technique that results most suitable for that specific task.

Figure 2.19 provides a graphical representation of the whole identification process, which is divided into three levels.

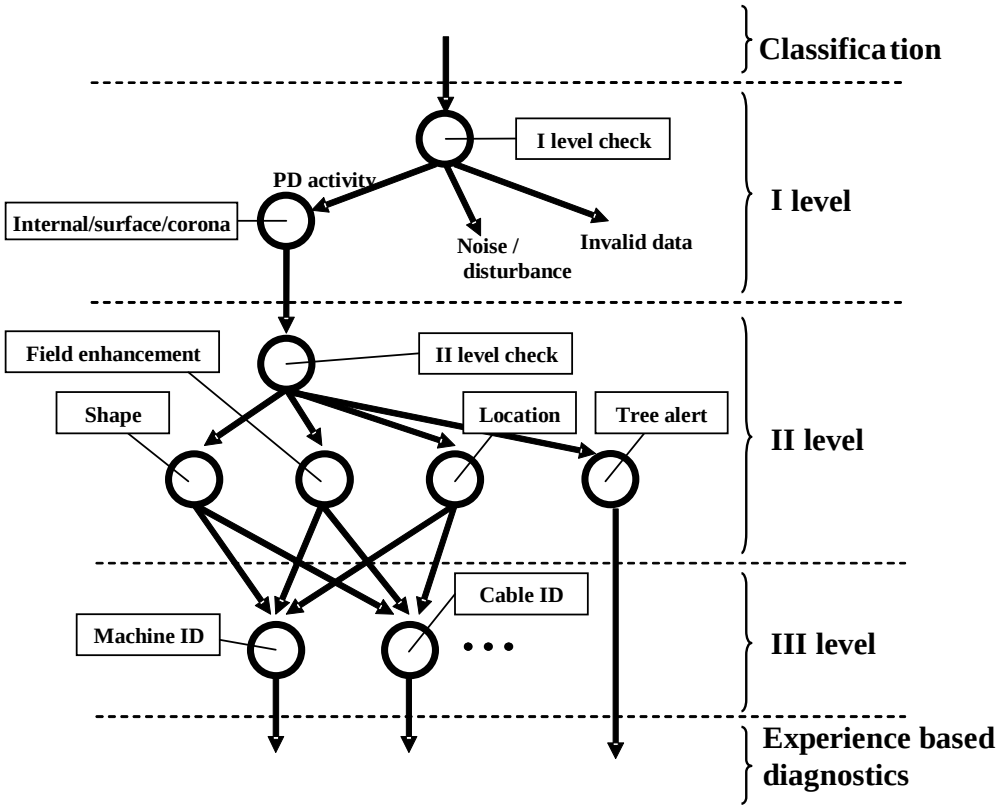


Fig. 2.19: *representation of the identification strategy.*

The first level of identification is general, because its output categories do not depend on the kind of electrical apparatus under test. It provides an indication about the nature of the defect with respect to five broad categories. These categories are: internal, surface or corona PD, noise and invalid data.

- Internal PD are those pulses which occur in air gaps delimited by dielectric surfaces, or solid dielectric and metallic electrodes, involving significant components of electric field orthogonal to defect surfaces (fixed gap).
- Surface PD are defined as discharges that develop on surfaces of solid insulating materials, involving significant field component tangential to the discharge surface (variable gap).
- Corona discharges constitute the well-known phenomenon of PD generated in open air (gas) from a sharp edge.
- Noise is sub-classified in background noise and external disturbances, and it is recognized by statistical tests.
- Invalid data collects all PD data that has not physical meaning or have not been successfully identified as noise, although being recognized as not suited to PD identification.

The output categories “internal, surface and corona” correspond to the three basic typologies to which a defect may belong and they refer directly to the local characteristics of the defect. Hence, a combination of these fuzzy outputs identifies the basic nature of the defect. It is noteworthy that a defect may constitute an intermediate situation, with respect to those basic typologies. Also important for understanding the first level identification results is the concept of likelihood. The likelihood is the sum of the output memberships and, ordinarily, is equal to one. As an example, let

us assume that the output of the classification tool is CORONA=0.0, INTERNAL=0.5, SURFACE=0.5. This means that the defect nature is intermediate between surface and internal. The total likelihood is 1.0, meaning that defect features are part of the knowledge base of the classification tool. On the contrary, if the output of the classification tool is CORONA=0.0, INTERNAL=0.1, SURFACE=0.1, the defect nature results intermediate between internal and surface, as in the previous case, but the identification likelihood is 0.2, meaning that defect features are partially new to the knowledge base of the classification tool.

The second level of identification provides a more detailed description of the defect, focusing, in particular, on location (with respect to the electrodes), shape (with respect to the electric field direction), field distribution inside the defect and presence of electrical treeing. Thus, at the second level each output category is pertinent to a specific characteristic of the defect. As for the first level, output categories do not depend on the kind of equipment under test. The larger is the membership for a given category, the more likely the defect will posses the corresponding characteristic.

The third level provides a detailed description of the defect for a specific class of equipment. Therefore, the third level output is not generally valid for any insulation system of electrical apparatus. For rotating machines, which constitute the only typology apparatus specifically discussed in this thesis, the output categories are “distributed microvoids”, “embedded delamination”, “conductor-side delamination”, “slot discharges” and “stress-grading discharges”. The larger is the membership for a given category, the more likely the defect will belong to that category.

With regard to the identification strategy, a brief discussion is needed about trend analysis. It is quite evident that the evolution of PD activity with time and (especially) applied voltage can return precious information. However, trend analysis poses three tough problems, especially if it has to be automated.

- It is complicated, as concerns the identification algorithm, because the amount of input information to be handled is generally very large.
- It is not always possible (e.g. applied voltage cannot be varied in the majority of the on-line tests).
- It involves a dependability decrease, because a large number of possible mistakes is added in the testing phase (it must be considered that test operators cannot be always PD experts).

For these reasons, in the adopted strategy, the identification process does not resort to trend analysis. Hence, each identification can be seen as a photograph of the PD activity; thus, a single identification result is pertinent to a specific time and applied voltage. Nevertheless, trend analysis can be applied, when appropriate data is available, on the basis of the single-shot-identification results. In this way, the diagnostic relevance of trend analysis will be endowed with more robustness.

Development of a diagnostic database

The importance of gathering measurement data in well organized structures, i.e. databases, is central to any issue which has to do with diagnostics. In particular, PD interpretation is a matter of learning from examples, and these examples are provided by measurement data. Therefore, it is essential to be able to retrieve specific information or analyze a large amount of data in a simple and fast way.

Databases pertinent to PD based diagnostics have two main purposes.

- Store all possible measurement data, so that they can be retrieved for future analysis and comparison.
- Derive sets of known examples, to derive concepts and train the machine for PD interpretation. Known examples are useful also to test the performance (efficiency) of existing identification procedures.

The database which collects and organizes all possible measurement data will be addressed to as "general database", while databases of known examples used for training or testing will be addressed to as "training database" or "testing database", respectively. Training and testing databases share the same scructure (they only differ for the way they are used), therefore no distinction will be made between them in the following.

A general database consists of a certain number of records, each record corresponding to a PD acquisition. The fields of the general database include identification markers (derived quantities), as well as information aimed at retrieving records in future search, e.g. acquisition date, operator, etc. Moreover, a general database may store the acquired quantities, besides the derived ones, in order to allow to process measurement data again at any time.

Figure 2.20 provides a schematic representation of the general database, being N_{tot} the number of its fields.

Record	**Attribute 1**	**…**	**Attribute N_{tot}**
1			
2			
…			

Fig. 2.20: *representation of a diagnostic database (general database).*

Training (and testing) databases constitute a sub-set of the general database; indeed, usually they do

contain few records, since the number of known examples is generally small. The extraction of a training database from the general database presupposes the definition of a certain number of categories (to which the records of the database must be associated), thus it depends on the identification strategy. For example, considering the first level of identification, let focus be made on three categories of PD activities: internal, surface and corona. The training database relevant to these categories will include all those PD activities (records) of the general database for which it is known whether they belong to the internal, surface or corona category. The information about the membership to the searched categories is associated to a specific field, which is called "target field".
Figure 2.21 provides a schematic representation of the training database, being N the number of fields which are extracted from the general database, apart from the target field.

Record	**Attribute 1**	**…**	**Attribute N**	**Target**
1				
2				
…				

Fig. 2.21: *representation of a diagnostic database (training database).*

Thus, the general database stores the experience of PD measurements. Such an experience provides prior knowledge when an identification issue is given, which determines a set of categories, and a set of examples (records) can be found for which the membership to those categories is known.
In general, a restricted number of records are available for the development of training databases and the majority of them are relevant to measurements performed on laboratory or industrial models. Hence, the training database could be seen as a sample, with respect to the population constituted by the general database.
In this light, the search for an identification algorithm can be seen as the search for a query (or set of queries) for each category, such that the records of the general database which satisfy this query are all those and only those belonging to the correspondent category. Clearly, to determine these queries one can rely only upon the training database, which, as a sample, is only partially representative of the general database. Therefore, when artificial intelligence (AI) algorithms are discussed in the next section, aimed at the analysis of training databases, efforts will be spent in order to develop techniques able to estimate the confidence of the learned algorithm, accounting for the fact that training databases are only partially representative of the general one.
Appendix A of this thesis discusses the effect of AI and fuzzy logic algorithm on the diagnostic databases.

Experimental activity disclosure

Experimental activity constitutes the basis of this thesis work. In fact, experimental data made available by PD measurements provide essential information to derive information, which may promote PD interpretation criteria. Interpretation criteria may be derived either by the human expert or automatically by the machine; in any case, the set up of a database gathering and organizing experimental data represents a fundamental issue. The effectiveness of such a database, in turn, depends on the typology of tested objects and on the testing procedure. In this paragraph, the typologies of measurement objects tested during this thesis work will be reported, together with a short account of the testing procedure. As regards the measurement objects, distinction will be made among laboratory models, industrial models and industrial objects, in accordance with the categorization dealt with in previous paragraphs.

Laboratory models

- Spherical and ellipsoidal cavities embedded in epoxy resin, characterized by different size and location with respect to the electrodes.
- Cavities obtained by placing different insulation layers between metal electrodes, being one of these layers provided with a cavity. Different specimens have been set up, in order to obtain the void to be either in contact with one electrode or embedded in the insulation. An example of these specimens is depicted in figure 2.22.
- Cavities obtained embedding in epoxy resin insulation boxes of cylindrical shape. Different specimens have been set up, in order to obtain voids of different shapes, i.e. flat (the gap was less then one third of the diameter of the insulation surfaces, as reported, e.g., in figure 2.23), compact (the gap was of comparable length with respect to the diameter of the insulation surfaces, as reported, e.g., in figure 2.24) and narrow (the gap was more than three times the diameter of the insulation surfaces). Furthermore, configurations were set up with not parallel electrodes, to obtain inhomogeneous field, as reported, e.g., in figure 2.25.
- Specimen “Cigre 1”, consisting of a cavity in contact with a spherical electrode and insulated with respect to the other, flat, electrode by epoxy resin. An example of this specimen is provided by figure 2.26.
- Specimens of the type “Cigre 2”, consisting of slabs of cross-linked XLPE where a needle electrode was inserted (and, in some cases, partially extracted to obtain a cavity in front of needle tip). An example of this specimen is provided by figure 2.27.
- Point to plane specimens, with different values of gap and of point sharpness. Furthermore, in some of these specimens an insulation layer (silicon rubber) was placed in contact with either the plane or the point electrode.
- Specimens where a conductor is laid on an insulation layer, to obtain discharges on the insulation surface.

- Specimens consisting of two slabs of either XLPE or EPR in contact with each other along their smooth surfaces, with spherical electrodes inserted in these slabs in order to obtain a component of the electric field parallel to the interface defined by the said insulation surfaces. An example of this specimen is provided by figure 2.28.

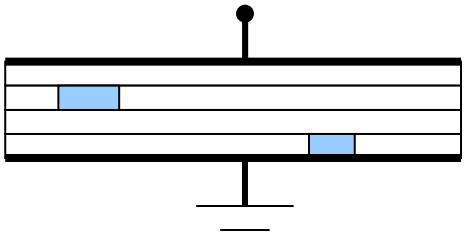


Fig. 2.22: *schematic representation of a multilayer specimen with voids at different locations.*

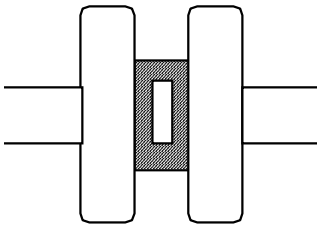


Fig. 2.23: *schematic representation of a specimen with flat (cylindrical) defect.*

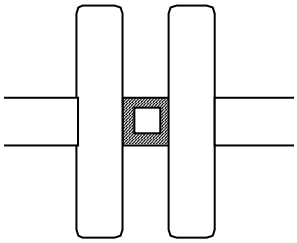


Fig. 2.24: *schematic representation of a specimen with compact (cylindrical) defect.*

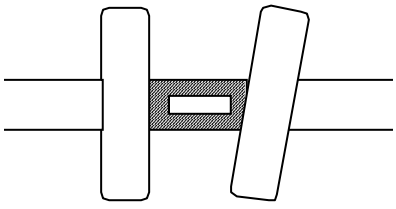


Fig. 2.25: *schematic representation of a specimen with narrow (cylindrical) defect, with inhomogeneous field.*

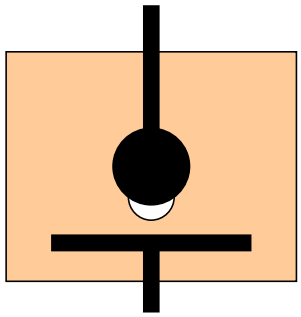


Fig. 2.26: *schematic representation of a specimen Cigre 1.*

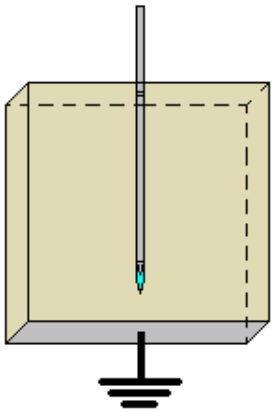


Fig. 2.27: *schematic representation of a specimen Cigre 2.*

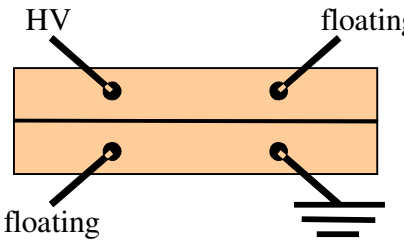


Fig. 2.28: *schematic representation of a specimen where the defect consists of an insulator-insulator interface.*

Industrial models[6]

- Stator bars with damaged, or even absent stress grading system.
- Stator bars with delaminations between the bar insulation and the slot (i.e. the iron core).
- MV cable with either flat or compact cavities inserted in a joint, being these cavities located close either to the grounded shield or to the HV electrode.
- Cable terminals of different typologies (e.g. field deflectors, field grading), with various kinds of artificial defects, e.g. wrong assembling or inclusions.

[6] In the following list, only those models will be reported, for the sake of brevity, on which systematic and repeated test sessions were carried out.

Industrial objects[3]

- MV and HV cables, with either XLPE or EPR insulation (measurements carried out both on line and off line).
- HF (high frequency) transformers for radar applications, insulated in epoxy resin (measurements carried out off line; more than 100 objects tested).
- Large generators (nominal voltage ranging from 6 kV to 15 kV). Measurements carried out both on line and off line.

Test procedure

As regards on line testing, measurements were carried out performing acquisitions at prefixed time intervals and varying the settings of the measuring system.

As regards off line testing, the following procedure was adopted.

- The test object is subjected to conditioning, applying a voltage lower than V_n (V_n is the rated voltage or a generic reference voltage).
- The applied voltage is incremented until inception (V_{inc} is recorded).
- The applied voltage is kept at the V_{inc} level for a time T (which depends on the test object) and acquisitions are performed at intervals T/c (c is a constant larger than 1, which shall be set according to the variability of the PD activity) → inception step of acquisition
- The applied voltage is raised to a voltage $k_1 V_{inc}$ (k_1 is a constant equal to, e.g. 1.1) and the first step of acquisition is performed, as for the inception step.
- The applied voltage is raised to a voltage $k_2 V_{inc}$ ($k_2 > k_1$) and the second acquisition step is performed, and so on, until a given voltage, V_{max}, is reached. V_{max} and k_i values depend on the object under test.
- The applied voltage is reduced with a ramp, and the extinction voltage is recorded.

On the basis of the experimental activity carried out during this thesis work, some criteria for the interpretation of PD activities were derived, which will be disclosed in the next paragraphs.

Such an experimental activity leaded also to various scientific works, [24-45] (to which one can refer to get further information), where the interpretation criteria described in the following have been applied to both laboratory specimens and industrial objects. In these publications, proof is given of the validity of the derived interpretation criteria, which have been implemented in an automatic diagnostic system resorting to those fuzzy logic and artificial intelligence techniques, which are described in detail in the third section of this thesis.

First level – reference PD activities and results

In this paragraph focus is made on the discrimination among three categories of PD generating defects, namely internal, surface and corona. For the discrimination of noise one can refer to [46, 47].
The output of the first-level identification algorithm consists of a vector of three elements; each one of these elements ranges in [0, 1] and specifies the membership to a specific output category.
Because the definition of the first-level categories is based mainly on the direction of the local field with respect to the discharge surfaces, the first level of identification has been found to be robust, i.e. it is scarcely influenced by voltage and time variations.
The identification markers which are selected for a specific identification task will be addressed in the following also as "input parameters", indicating that they constitute the input for the identification algorithm relevant to a specific identification task.

Input parameters for the first-level identification:

- FiminPos, FiminNeg
- FimeanPos, FimeanNeg
- DFiPos, DFiNeg
- Sk_ampPos, Sk_ampNeg
- betaPos, betaNeg
- ND

Thus, the identification algorithm of first level requires eleven input parameters. However, it must be observed that the first-level algorithm, being aimed at recognizing the basic nature of a defect, is supposed to return coherent results also in the case it is run separately on the two PD polarity distributions. Indeed, the algorithm is run twice, each time taking the six input parameters pertinent to a single polarity (ND is a combined quantity, therefore it does not change from one polarity to the other), thus simplifying remarkably the identification process. This procedure is subjected to a check on parameter ND. In fact, if ND equals zero, only the negative PD distribution is present, while if ND equals one, only the positive distribution is present. In case both polarities are present, the results obtained for the positive and negative polarities are multiplied per ND and (1-ND), respectively, then summed.
Of the selected quantities, three are relevant to the phase distribution (Fimin, Fimean and DFi), two to the amplitude distribution (Sk_amp and beta) and one (ND) is global, being based on the number of detected PD. It is noteworthy that the two identification markers pertinent to the amplitude distribution are shape factors, i.e. they are not associated to absolute values (mV or pC), which are dependent on the size of the defect and on its location with respect to the coupling device and thus are not relevant to the basic nature of the defect. The fact that quantities related to the intertime distribution have not been selected is determined by the need to focus on the local field direction

with respect to the discharge surfaces (according to the definition of internal, surface and corona categories), while the quantities derived from the intertime distribution are particularly related to the generation rate of initial electrons. As regards the parameters pertinent to the phase distribution, it can be observed that those three quantities provide, if combined together, information about the skewness of the phase distribution as well.

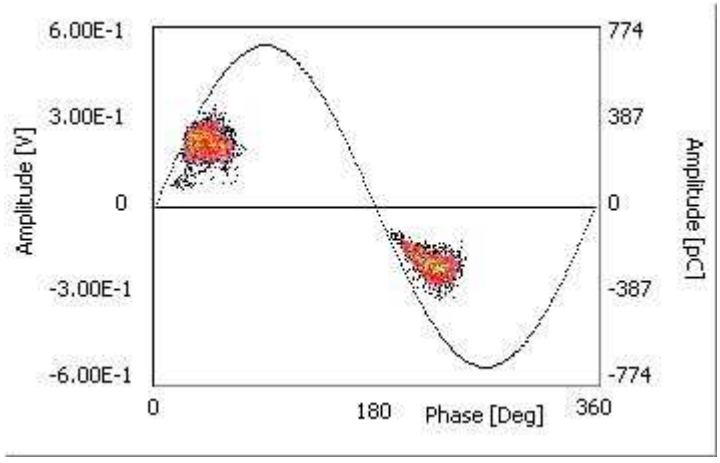
As regards "Invalid data" category, it is relevant to PD activities (the evaluation of "noise" category is performed first, therefore it is already excluded at the time the category "invalid data" is evaluated) which are either absurd (no physical meaning) or not significant (not enough information for statistical processing), according to the following criteria.

- Absurd data: (Fimean < Fimin) OR (Fimean > Fimin+DFi) OR any quantity is out of range.
- Not significant data: (NPos < 100) AND (NNeg < 100).

In the following, for each one of the output categories, the recognition principles will be reported together with some examples.

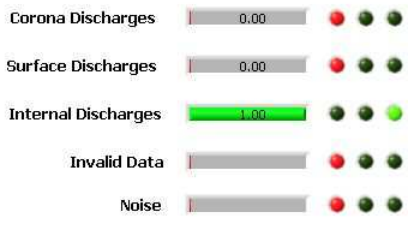
Important remark about examples. It is not possible, for need of brevity, to report a number of examples which is representative of the complex variety of the situations that can be faced in PD measurements.

Internal PD



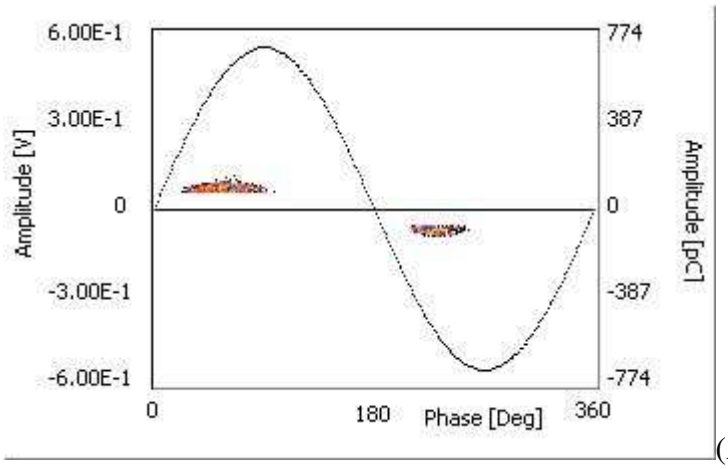
(A)

(B)	Fimin [degrees]	Fimean [degrees]	DFi [degrees]	Sk_amp	beta	N	ND
Positive Polarity	12	42	62	-0.40	6.2	3014	0.49
Negative Polarity	13	45	62	0.16	5.9	3078	



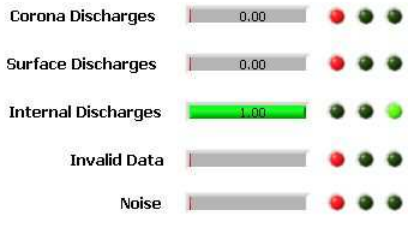
(C)

Fig. 2.29: *example of internal PD activity on a HF transformer (void embedded in epoxy resin), (A) PRPD pattern, (B) table of parameters, (C) identification result.*



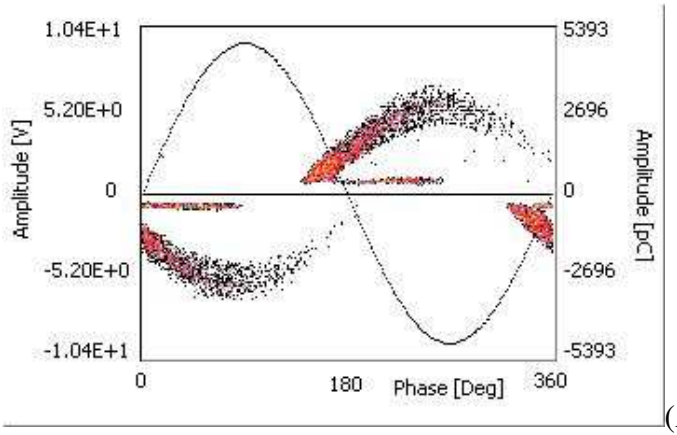
(A)

(B)	Fimin [degrees]	Fimean [degrees]	DFi [degrees]	Sk_amp	beta	N	ND
Positive Polarity	22	59	77	0.15	8.3	1210	0.64
Negative Polarity	28	47	49	0.34	8.7	692	



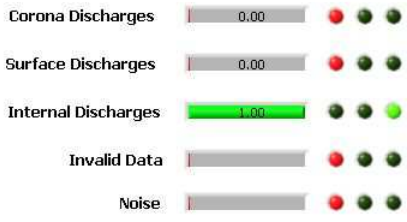
(C)

Fig. 2.30: *example of internal PD activity on a narrow cylindrical artificial defect, (A) PRPD pattern, (B) table of parameters, (C) identification result.*



(A)

(B)	Fimin [degrees]	Fimean [degrees]	DFi [degrees]	Sk_amp	beta	N	ND
Positive Polarity	-40	19	216	0.64	1.5	4196	0.53
Negative Polarity	-40	18	232	0.48	1.8	3744	



(C)

Fig. 2.31: *example of internal PD activity on a void embedded in epoxy resin (artificial defect), (A) PRPD pattern, (B) table of parameters, (C) identification result.*

Interpretation criteria for internal PD.

PD activities of the internal typology are very common and may be relevant to a wide range of possible defects. Indeed, no specific characteristic can be attributed to internal activities as a whole, which, in turn, have to be identified as the conjunction of a large variety of situations. Moreover, defects whose basic nature is mainly surface or corona are likely to exhibit some characteristics which are typical of internal defects. In fact, in the majority of the situations, the local field has a non negligible component orthogonal to the discharge surfaces, thus determining a certain contribute to internal nature.

In order to derive interpretation criteria for internal PD activities, one can focus on PD mechanism in cavities, which is described in the first section of this thesis. Recalling some basic concepts, on one side it can be observed that three characteristics can be associated to internal defects: constant gap, local field homogeneous in the discharge volume, memory effect. On the other side, nothing can be said about neither PD generation rate, nor PD magnitude (as regards absolute values), since no assumption was made about discharge surfaces nor defect size (and location with respect to the

coupling device) and, most of all, the overvoltage ratio is, in general, unknown.

When memory effect is evident, involving significant acticipation and posticipation effect, internal activities exhibit a low value of *Fimin* (i.e. lower than zero) and a high value of *Dfi* (e.g. higher than 180 degrees). In case the memory effect is not particularly evident, usually it is still true that the local field varies with time with an approximately sinusoidal law (the decay time of deployed charge may be comparable with the voltage period, but not negligible), therefore PD events are more probable when the applied voltage increases with higher rate, i.e. for low phase values. Hence, *Fimean* is usually observed around 50÷60 degrees (close to the inception) and will decrease as the overvoltage ratio rises.

Two basic behaviors can be distinguished for self-sustained internal PD activities, with respect to PD time of occurrence:

- PD occur grouped in subsequent events characterized by high repetition rate, i.e. groups of PD occurring within a half period (of the applied voltage), being these groups separated by a time lag either comparable with the half period (in case the successive group occurs as soon as the applied voltage changes polarity) or significantly larger than the half period.
- As an average, one PD event is observed for each half-period.

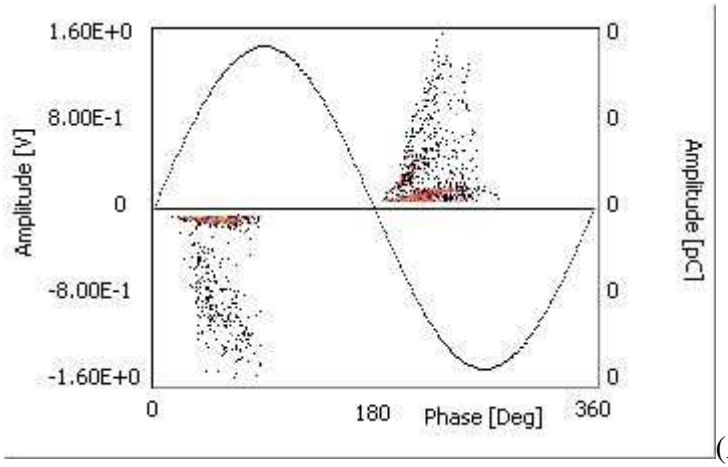
In the former case, the amplitude dispersion may be remarkable ("rabbit ear" like pattern) and *beta* and *Sk_amp* will be low and high, respectively. On the other side, *Fimin* is usually low and *DFi* high, because of memory effect (Fig. 2.31).

In the latter case, *Fimin* is usually low but not below zero, *DFi* is fairly small (or, even, very small) and the amplitude dispersion is low (*beta* high and *Sk_amp* low) (Figs. 2.29 and 2.30).

A characteristic which is quite typical of internal defects consisting of voids which have a rather compact, regular shape, is that the contour of the PRPD pattern is regular, i.e. it resembles a sinusoidal profile (Figs. 2.30 and 2.31).

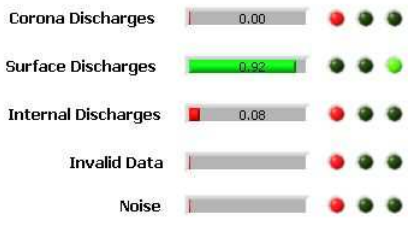
In the majority of the situations, the PD activity is fairly symmetric (symmetry evaluation consists of a comparison between the PD activities of the two polarities) with respect to the number of discharges, therefore *ND* assumes average values (i.e. relatively close to 0.5).

Surface PD



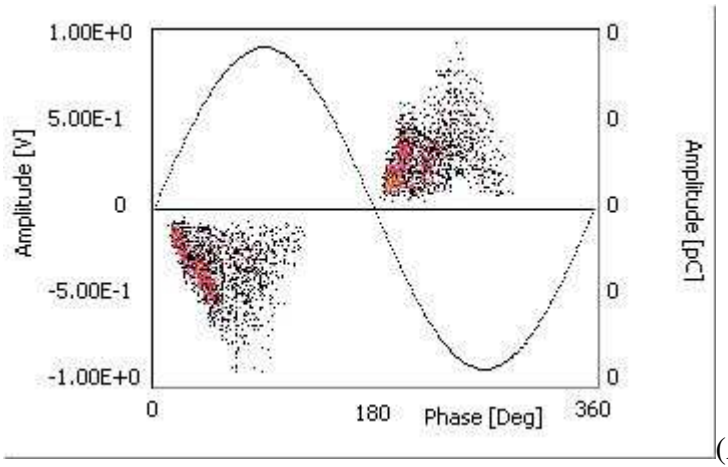
(A)

(B)	Fimin [degrees]	Fimean [degrees]	DFi [degrees]	Sk_amp	beta	N	ND
Positive Polarity	15	52	73	1.49	0.9	864	0.44
Negative Polarity	6	42	96	2.00	1.1	1098	



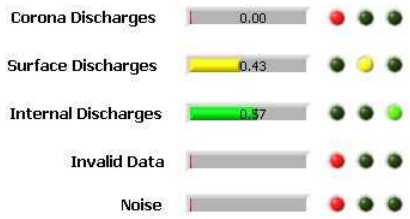
(C)

Fig. 2.32: *example of surface activity on the stress grading system of a MV cable termination of stress grading type (PD develop in air), (A) PRPD pattern, (B) table of parameters, (C) identification result.*



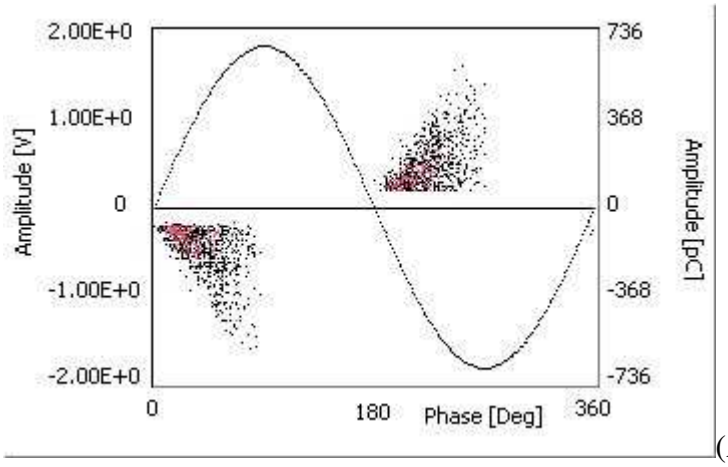
(A)

(B)	Fimin [degrees]	Fimean [degrees]	DFi [degrees]	Sk_amp	beta	N	ND
Positive Polarity	11	50	113	0.69	2.3	1866	0.47
Negative Polarity	4	40	108	1.03	2.1	2131	



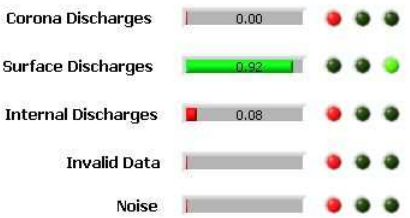
(C)

Fig. 2.33: *example of mixed internal-surface activity on a field deflector MV cable termination (PD develop between termination and cable insulation surfaces), (A) PRPD pattern, (B) table of parameters, (C) identification result.*



(A)

(B)	Fimin [degrees]	Fimean [degrees]	DFi [degrees]	Sk_amp	beta	N	ND
Positive Polarity	-3	28	92	0.76	1.8	2154	0.54
Negative Polarity	2	32	90	0.98	1.9	1843	



(C)

Fig. 2.34: *example of surface activity on a stator bar stress grading, (A) PRPD pattern, (B) table of parameters, (C) identification result.*

Interpretation criteria for surface PD.

PD of the surface typology are likely to be external with respect to the object under test. Nevertheless, they may constitute dangerous situations and, even, lead to insulation breakdown. The

heavy degradation of the insulation surface caused by surface PD, which may lead to breakdown, is generally referred to as tracking process.

It is quite usual that a defect has a nature which is intermediate between internal and surface. This circumstance occurs when the defect is a flat cavity (i.e. one of its dimensions is smaller than the others, e.g. of one order of magnitude) and when the local field has components both parallel and orthogonal to the surfaces defined by the defect. In these cases the defect is actually enclosed in solid material, which may or may not comprise a metallic surface. These situations may be associated to delaminations between insulation layers, or between an insulation layer and a conductor.

It must be observed that, in case an interface between two layers is subjected to a local field which has a non negligible component parallel to the interface plane, but no void is present, PD may still take place, but they will exhibit an internal nature, with significant anticipation effect. Furthermore, in this case PD repetition rate is likely to be very law and the PD activity will be subjected to fluctuations, since the defect may remain inactive for even long times (up to weeks).

Remark. To link the following considerations to the description of PD mechanisms, one can refer to the first section of this thesis, in particular to the part which deals with strongly diverging field configurations.

The main characteristic of surface PD activities consists of a considerable dispersion of the amplitude distribution, which is associated to values of *beta* and *Sk_amp* low and high, respectively. Small discharges constitute the majority of the PD events, but qute large discharges may appear, although with low probability. In addition, the memory effect is not as remarkable as for internal defect, therefore *Fimin* cannot be significantly lower than zero. At the same time, the critical field is usually very low, hence *Fimin* is usually lower than 40 degrees. Furthermore, surface PD activities are usually characterized by groups of discharges occurring within a half period of the applied voltage with very high repletion rate, each group taking place regularly at every polarity reversal. Hence, the PD repetition rate is likely to be high, as absolute value.

With reference to the PRPD pattern, it is interesting to observe the PD distribution (of a single polarity) with respect to its contour.

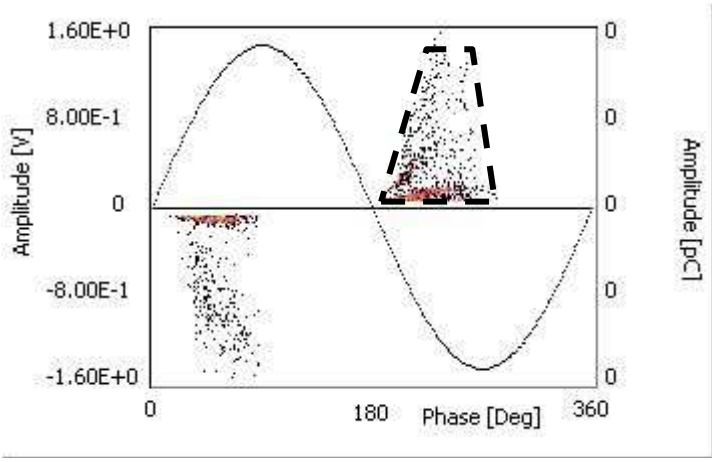
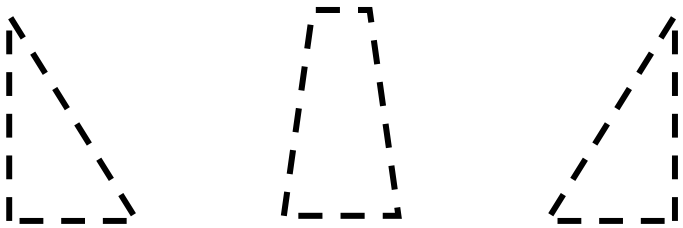


Fig. 2.35: *example PRPD pattern PD contour evaluated ictu oculi.*

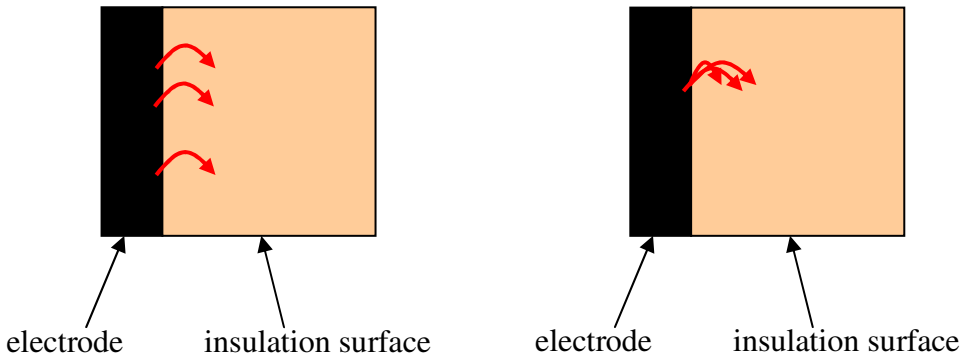


(A) (B) (C)

Fig. 2.36: *schematization of possible shapes of the contour of a surface PD activity, namely descendant (A), constant (B) and ascendant (C).*

Three basic contour profiles can be distinguished: descendant (Fig. 2.36A), constant (Fig. 2.36B) and ascendant (Fig. 2.36C). A descendant profile suggests the presence of a non negligible memory effect, and is often associated to defects which have a nature intermediate between surface and internal. An ascendant profile suggests that the average statistical time lag progressively increases, during the half period of the applied voltage. This is imputable to a shielding effect of the discharges of a single group, which is negligible in the case of constant profile. The shielding effect occurs when electrons deployed on the insulation surface in proximity of the electrode surface reduce the electrical field.

Furthermore, it can be observed that sometimes PD distributions of surface PD activities are characterized by a maximum phase (*Fimin + DFi*) which is not larger than 90 degrees (Figs. 2.32, 2.34). Considering the geometry of surface defects (strongly inhomogeneous field), a possible explanation of such behavior is the following. Subsequent PD may consist of either the transfer of different charge distributions (taking place "in parallel" in different sites), Fig. 2.37A, or subsequent charge transfer in the same site, Fig. 2.37B. If the discharges which form a group (i.e. those PD occurring within the same half-period of the applied voltage) correspond to subsequent charge transfers in the same site, thus involving memory effect, they occur only as long as the applied voltage keeps increasing. Hence, surface PD distributions which occur for phase values larger than 90 degrees likely involve a plurality of discharge sites, within the same defect (Fig. 2.35).



(A) (B)

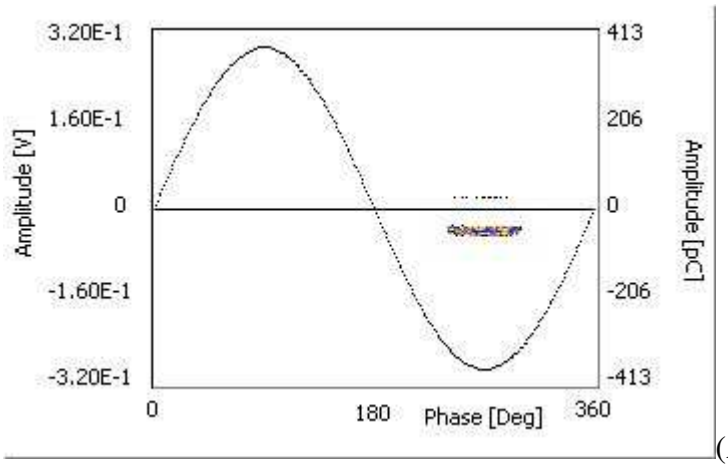
Fig. 2.37: *schematic top view of the discharge surfaces in a surface defect, distinguishing two possible PD mechanisms: (A) multiple sites, (B) single site.*

In addition, *DFi* may be influenced by the shielding effect associated to the electrons deployed by the PD in the portion of insulation surface in proximity of the electrode. Values of *DFi* of about 100 degrees indicate low shielding effect (Figs. 2.33 and 2.34), while values of about 60 degrees indicate high shielding effect (Fig. 2.32).

Fimean is determined mainly by the overvoltage ratio and the memory effect. In the majority of the situations *Fimean* is fairly close to 45 degrees.

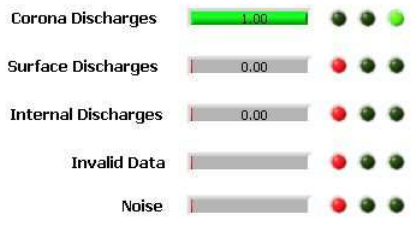
As regards the symmetry between the two discharge polarities, with respect to the number of PD events, it was observed that surface PD activities are mainly symmetric (ND close to 0.5) if the discharge surface which is an electrode is conductive, while is not symmetric if the discharge surface which is an electrode is semi-conductive.

Corona PD



(A)

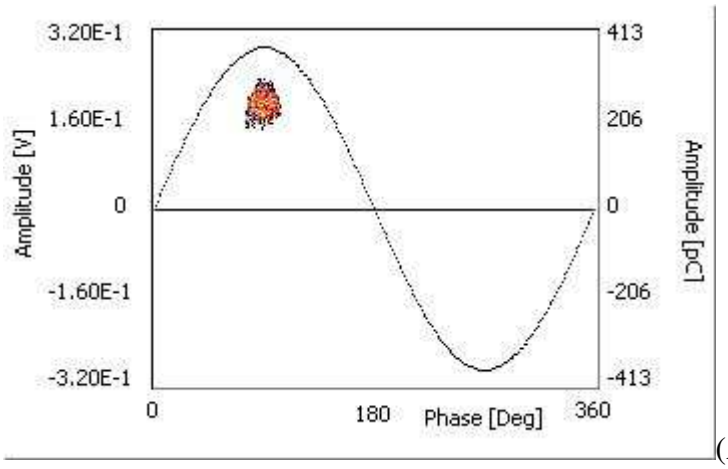
(B)	Fimin [degrees]	Fimean [degrees]	DFi [degrees]	Sk_amp	beta	N	ND
Positive Polarity	-	-	-	-	-	0	0
Negative Polarity	59	90	60	0.00	11	1753	



(C)

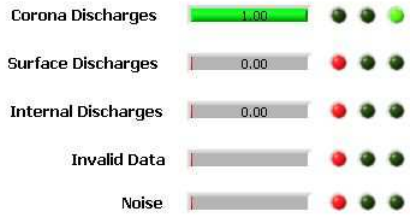
Fig. 2.38: *example of corona PD activity on a point to plane specimen, with plane grounded (PD in*

correspondence of the point), (A) PRPD pattern, (B) table of parameters, (C) identification result.



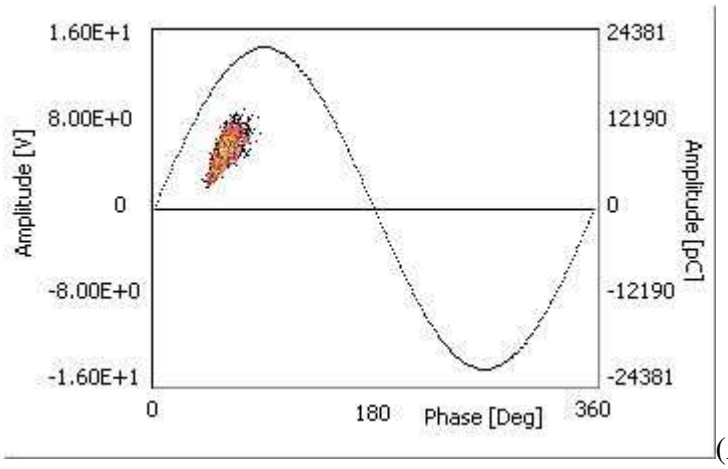
(A)

(B)	Fimin [degrees]	Fimean [degrees]	DFi [degrees]	Sk_amp	beta	N	ND
Positive Polarity	74	87	29	0.23	12.1	1557	1
Negative Polarity	-	-	-	-	-	0	



(C)

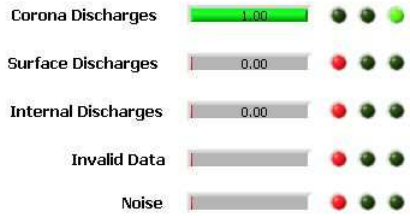
Fig. 2.39: *example of corona PD activity on a point to plane specimen, with point grounded (PD in correspondence of the plane), (A) PRPD pattern, (B) table of parameters, (C) identification result.*



(A)

(B)	Fimin [degrees]	Fimean [degrees]	DFi [degrees]	Sk_amp	beta	N	ND

Positive Polarity	39	59	49	-0.04	4.6	2000	1
Negative Polarity	-	-	-	-	-	0	



(C)

Fig. 2.40: *example of corona PD activity on a point to plane specimen, with plane connected to HV and insulated with silicon rubber(PD in correspondence of the plane), (A) PRPD pattern, (B) table of parameters, (C) identification result.*

Interpretation criteria for corona PD.

PD activities of the corona typology can be associated to the point to plane configuration, dealt with in the first section of this thesis. In this light, the most standard configuration relevant to corona activities occurs when the role played by the plane electrode is negligible and the overvoltage ratio is relatively low (i.e. the applied voltage is close to the inception voltage). In this case, corona activities are easily recognizable by observing the PRPD pattern. In fact, only one of the two polarity distributions is present, in particular, the positive one, if the point electrode is grounded (*ND* equals 1, as in Figs. 2.38 and 2.39), the negative one, if the point electrode is connected to HV (*ND* equals 0, as in Fig. 2.37). In addition, the PD distribution has very low amplitude dispersion (*beta* is high, e.g. higher than 7, and *Sk_amp* is close to 0) and occurs in correspondence of the maximum value of the applied voltage (*Fimean* is high, i.e. close to 90 degrees, *DFi* is small, e.g. 50 degrees, and, consequently, *Fimin* is high). However, when the overvoltage ratio is larger, the situation changes. First of all, PD occur at both polarities (thus *ND* may have values close to 0.5, typical of symmetric activities). The activity relevant to the point, i.e. those PD occurred when the point electrode is a cathode, is characterized by an increment of *DFi*, and a decrement of *Fimin*, while *Fimean* is almost constant and also the amplitude dispersion is still very low. The activity relevant to the plane is characterized, with respect to the activity relevant to the point, by PD of remarkably larger amplitude, by a lower value of *DFi* and by a relatively larger amplitude dispersion. In case an insulation layer is interposed between the two electrodes, in contact with the plane electrode, the PD activity relevant to the point shifts toward lower phase values (*Fimean* decreases) and *Fimin* decreases remarkably, since memory effect is present (Fig. 2.39). Also, the amplitude dispersion increases.

Second level – reference PD activities and results

The second level of identification is aimed at inferring three characteristics of insulation defects, namely location, shape and local field enhancement, and at recognizing whether the PD activity is associated to electrical tree phenomena. Therefore, the second level of identification provides much more specific information with respect to the first level; however, it is less robust than the first level, i.e. it may be remarkably influenced by voltage and time variations, as well as by the degradation stage.

Defect location with respect to the electrodes, defect shape and the local field distribution inside the defect constitute factors which interact and whose effects superpose in the PD distributions (i.e. amplitude, phase and intertime distribution). The purpose of this paragraph is to describe interpretation criteria in order to decuple, as much as possible, these effects, and to suggest those identification markers which are considered most suited for each identification issue, considered singularly.

Contrarily to the first level of identification, for location and field enhancement issues, the comparison between the two polarity distributions is fundamental. Indeed, the evaluation of PD distribution dissymmetry is the clue for this kind of identification. In this work, the following typologies of dissymmetry are considered:

- Amplitude → reference identification marker: *QD*
- Amplitude dispersion → reference identification marker: *stdQD*
- Phase → reference identification marker: *FD*
- PD occurrences → reference identification marker: *ND2 (or ND)*
- Intertime → reference identification marker: *ITD*

In the evaluation of the dissymmetry of PD activity, one must consider separately the case of unipolar activities (from that of bipolar activities), i.e. the case where only one of the two polarity distributions is present. In fact, in this case the combined quantities (e.g. *QD*, *stdQD*, etc.) cannot be calculated. Therefore, a preliminary check is performed on *ND2*, which is equal to –1 if only the negative distribution is present, to 0 if both distributions are present and they have the same number of PD and to +1 if only the positive distribution is present. For each identification issue, three distinct algorithms follow this check, being associated to low, medium and high values of *ND2*. It is worth observing that the translation into mathematical terms of concepts as “low”, “medium” and “high” was performed resorting to a fuzzy logic, as will be described in details in the third section of this thesis.

As regards defect shape and tree growth, the dissymmetry evaluation is not as important as for location and field enhancement.

Defect location

Output categories:

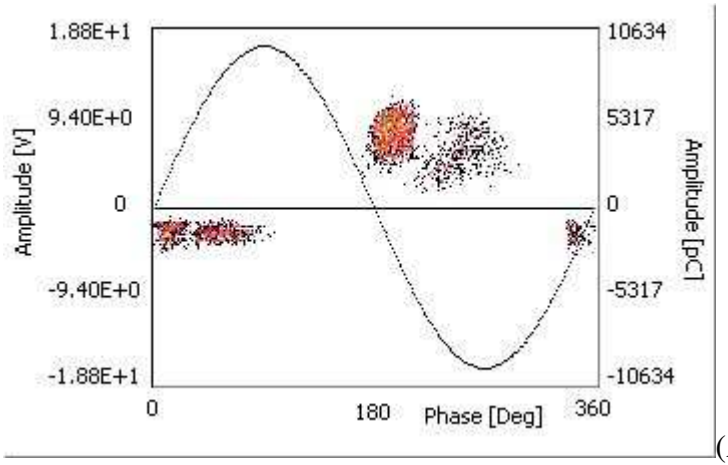
- LV side → indicates that one of the discharge surfaces coincides with the ground electrode, while the other discharge surface is an insulating one.
- HV side → indicates that one of the discharge surfaces coincides with the electrode connected to the power supply, while the other discharge surface is an insulating one.
- Embedded → indicates that both discharge surfaces are insulating ones.

Input parameters for the second-level identification / location:

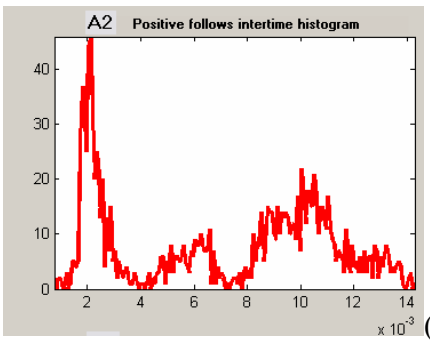
- ITD
- P5
- FD
- stdQD
- betaPos, betaNeg
- betaD

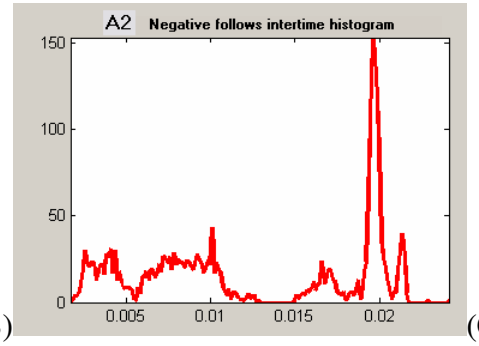
All of the above reported parameters were described in previous paragraph, except for *betaD*, which is simply derived from *betaPos* and *betaNeg*, combined through the *normalizedratio* operator (2.7). In the following, some examples are reported which are particularly significant for defect location; then, interpretation criteria will be described.

<u>HV side</u>



(A)

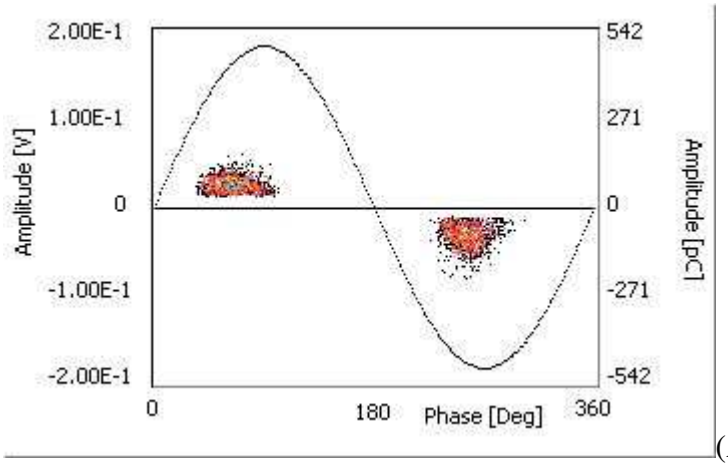


(B)


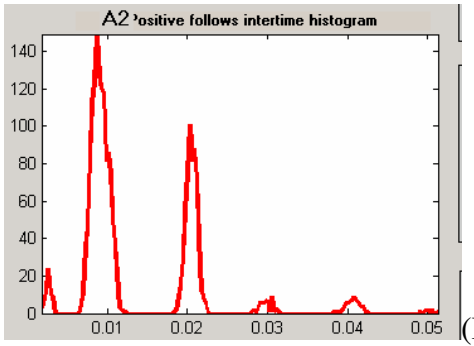
(C)

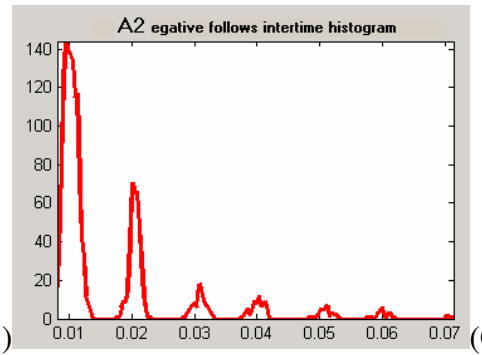
(D)	ND2	ITD	P5	stdQD	FD	beta	betaD	pF1	pF2
Positive Polarity	-0.48	-0.44	0.21	-0.42	-12	2.3	-0.56	0.31	0.28
Negative Polarity						5.2		0.17	0.23

Fig. 2.41: *example of HV side PD activity on a Cigre 1 specimen, with plane electrode grounded, (A) PRPD pattern, (B) Positive-follows intertime distribution, (C) Negative-follows intertime distribution, (D) table of parameters.*



(A)



(B)


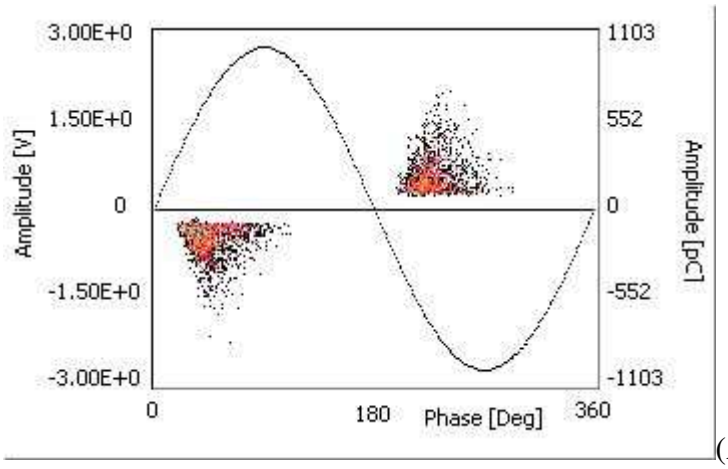
(C)

(D)	ND2	ITD	P5	stdQD	FD	beta	betaD	pF1	pF2
Positive Polarity	0.12	-0.15	-0.02	-0-38	-19	3.8	0.21	0.53	0.33
Negative Polarity						3.0		0.32	0.35

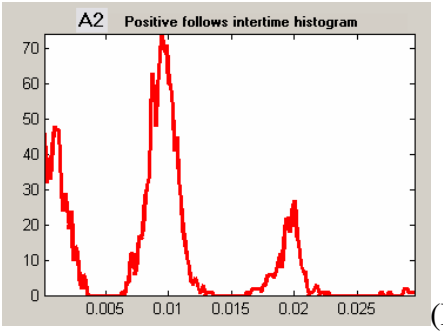
Fig. 2.42: *example of HV side PD activity on a cable joint with artificial defect (compact void at HV side),*

(A) PRPD pattern, (B) Positive-follows intertime distribution, (C) Negative-follows intertime distribution, (D) table of parameters.

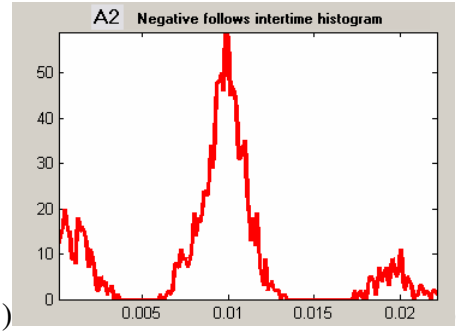
Embedded



(A)



(B)

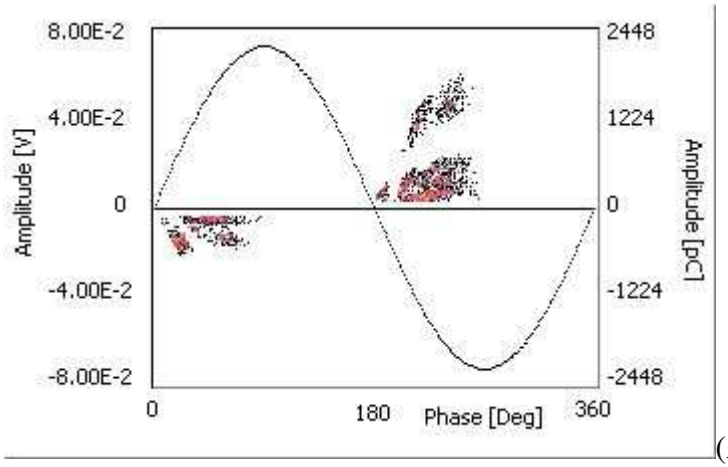


(C)

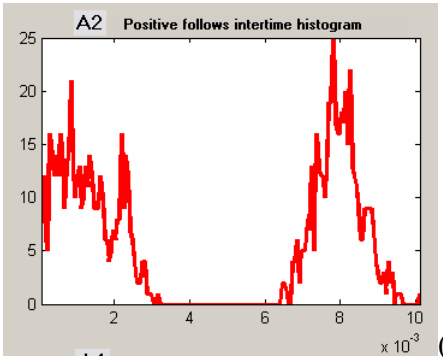
(D)	ND2	ITD	P5	stdQD	FD	beta	betaD	pF1	pF2
Positive Polarity	0.25	-0.02	-0.01	-0.13	2	2.1	0.0	0.18	0.25
Negative Polarity						2.1		0.44	0.34

Fig. 2.43: *example of embedded PD activity on tthe stator bar of a generator, with internal delamination between mica foils, (A) PRPD pattern, (B) Positive-follows intertime distribution, (C) Negative-follows intertime distribution, (D) table of parameters.*

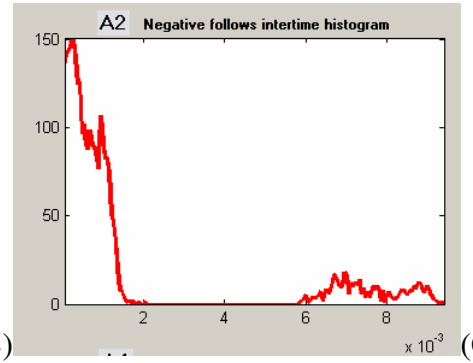
LV side



(A)



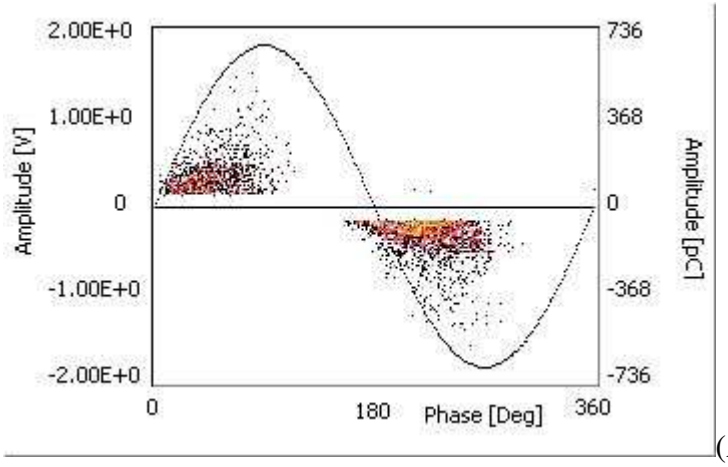
(B)



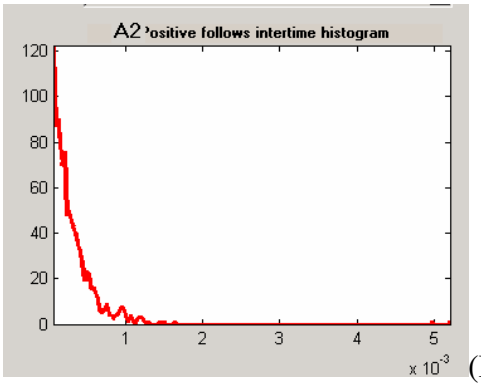
(C)

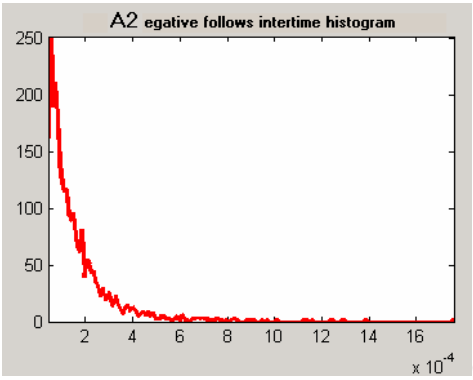
(D)	ND2	ITD	P5	stdQD	FD	beta	betaD	pF1	pF2
Positive Polarity	-0.68	0.63	-0.43	-0.63	5	2.7	0.52	0.17	0.30
Negative Polarity						1.3		0.43	0.33

Fig. 2.44: *example of LV side PD activity on a MV cable, with damaged semiconductive shield, (A) PRPD pattern, (B) Positive-follows intertime distribution, (C) Negative-follows intertime distribution, (D) table of parameters.*



(A)



(B)


(C)

(D)	ND2	ITD	P5	stdQD	FD	beta	betaD	pF1	pF2
Positive Polarity	-0.60	0.56	-0.34	0.05	26	2.0	0.05	0.43	0.42
Negative Polarity						1.9		0.61	0.50

Fig. 2.45: *example of LV side PD activity on the stator bar of a generator (PD occur in correspondence of the stress grading), (A) PRPD pattern, (B) Positive-follows intertime distribution, (C) Negative-follows intertime distribution, (D) table of parameters.*

Interpretation criteria for defect location

If voids embedded in solid insulation are considered, it seems reasonable to assume that the defect location does not influence significantly the local field. In fact, the electric field in the void should be about the same, independently of the fact that the void is close to one electrode or is located somewhere between the electrodes. Therefore, the factor which differentiates voids at different locations should be constituted mainly by the nature of the discharge surfaces. In fact, if the defect is embedded both discharge surfaces are insulating; on the contrary, when the defect is either LV side or HV side, one of the discharge surfaces is metallic (in the opinion of the author, the case of semi-conductive surfaces should be studied separately). In case the discharge surfaces have different nature, one would expect to observe some kind of dissymmetry, related to the starting electron generation rate; on the contrary, a dissymmetry of such a kind is not expected for embedded defects. Indeed, it is important to single out what specific kind of dissymmetry can be associated solely (or principally) to defect location, since there can be many different kinds of dissymmetry besides defect location (e.g. field enhancement or a generic difference in the nature of the discharge surfaces). In this light, it is important to focus on which discharge surface is a cathode in which polarity, as it is done, for example, in figure 2.46.

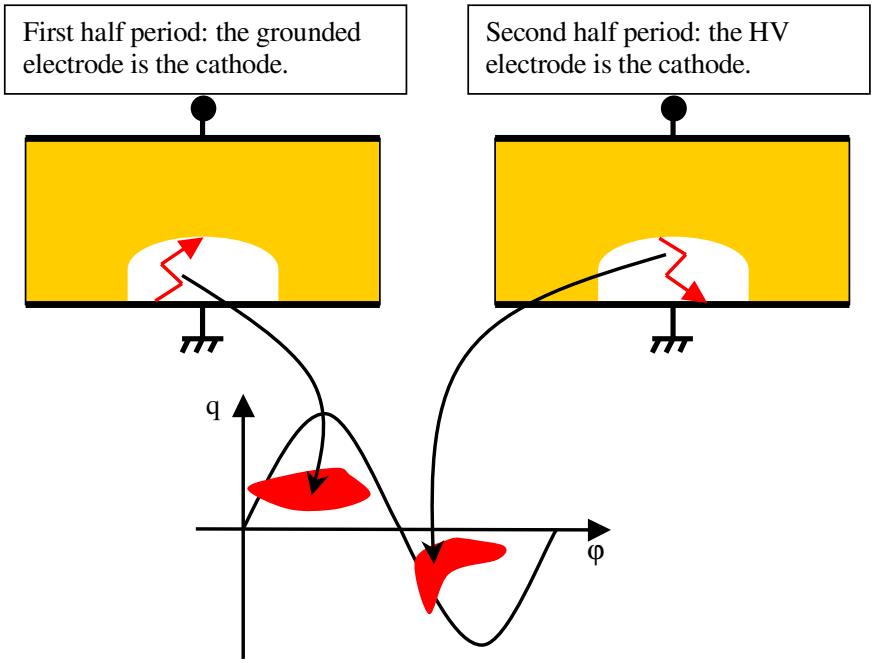


Fig. 2.46: *Schematization of the discharge process with respect to PD polarity, for a defect localised close to the LV electrode*

It was observed, [48-50], that dielectric surfaces are more efficient than metallic surfaces in the emission of first electrons (i.e. electrons able to initiate electric avalanches that may develop in partial discharges, if the field inside the defect exceeds a critical value), at least for values of the applied voltage which are not too far from the overvoltage ration (e.g. for values of the overvoltage ratio up to 1.4). The fact that dielectric surfaces are more efficient than metallic surfaces in the emission of first electrons may be explained considering that charge carriers deployed on a highly conductive surface are immediately depleted, while charge carriers deployed on a dielectric surface are stored in voltage traps and can trigger more easily new PD events.

This speculation was confirmed by experimental results [50], which showed that the average intertime is smaller when the PD that follows is initiated by an insulating surface. Therefore, the identification marker *ITD*, which is the ratio between the average intertime of PD originating at the two discharge surfaces, is particularly suited for this kind of identification issue. In fact, a large value of ITD (close to +1) means that, as an average, intertimes where the second PD is positive (i.e. occurred in the first half period of the applied voltage) are larger than intertimes where the second PD is negative. This means that the PD statistical time lag is, as an average, higher for the defect surface at the ground side (that is, the discharge surface that is a cathode in the first half period of the applied voltage), indicating that such a surface has a lower efficiency in electron emission, compared to the other surface (the one on the side of the HV electrode, which is an anode in the first half period). Thus, a large value of *ITD* would suggests that the surface on the LV side is metallic and the one on the opposite side (HV side) is dielectric (being the former surface less efficient in electron emission compared to the latter), indicating that the defect is in contact with (or very close to) the LV electrode. Obviously, a similar speculation can be done for small values of

ITD. Moreover, this kind of dissymmetry affects PD phase as well. In fact, the discharge polarity for which the average intertime is lower is likely characterized by smaller PD phases.
However, for large values of the overvoltage ratio, the metallic discharge surface may become more efficient than the insulating one in generating first electrons; as a consequence, the number of PD which occur at the metallic surface is larger than that of PD which occur at the insulating surface. For this reason, quantity P5 was defined, which combines two dissymmetry typologies, namely intertime and PD occurrence dissymmetry. Furthermore, it can be observed that PD which initiate at the insulating surface are characterized by larger dispersion, because the streamer channel, in that case, may develop on the insulating surface (as was mentioned in the first section).
Hence, the criteria to infer the location of a defect associated to a bipolar PD activity can be summarized as follows:

- ITD low ; P5 high ; FD low ; stdQD high => HV side
- ITD high ; P5 low ; FD high ; stdQD low => HV side
- ITD medium ; P5 medium ; FD medium ; stdQD medium => Embedded

Thus, four conditions are evaluated; it is important to underline that these conditions should be evaluated with a priority ordering, i.e. *ITD* (first), then *P5*, *FD* and, finally *stdQD*. In other words, if *ITD* has an average value (i.e. around 0), indicating the absence of intertime dissymmetry, parameter *P5* is evaluated (and so on). On the contrary, if *ITD* is either low or high, the other parameters are not taken into account (or are taken into account with a low weight), because they may even provide contradictory information.

When dealing with unipolar activities (those activities where only one of the two PD polarity distributions is present, or, more in general, where *ND2* is either very low or very high), the evaluation of combined parameters (*ITD*, *P5*, FD and *stdQD* in this case) may lead to deceptive and insignificant results, because of the lack of PD events for one polarity distribution. Indeed, these situations indicate, by themselves, a strong dissymmetry. In this light, and according to experimental observation, it was found that defect location can be inferred resorting to parameters *betaPos*, *betaNeg*, *betaD*, and the values of the output of the first level of identification, in particular regarding the corona category, namely *corPos* and *corNeg*.
The criteria to infer defect location in case of unipolar activities can be summarized resorting to the following lists.

Negative unipolar PD activity.

- (*betaNeg* high and *betaD* low) or *corNeg* => HV side
- *betaNeg* low => LV side
- otherwise embedded

Positive unipolar PD activity.

- (*betaPos* high and *betaD* high) or *corPos* => HV side

- *betaPos* low => LV side
- otherwise embedded

Defect shape

This characteristic is proper of cavities internal to solid insulation, which may or may not be adjacent to a conductor.

Output categories:

- Flat → indicates that the gap (i.e. the distance between the discharge surfaces) is remarkably smaller then the smallest of the dimensions of the discharge surfaces.
- Compact → indicates that the gap is comparable with the smallest of the dimensions of the discharge surfaces.
- (narrow → indicates that the gap is remarkably larger then the smallest of the dimensions of the discharge surfaces).

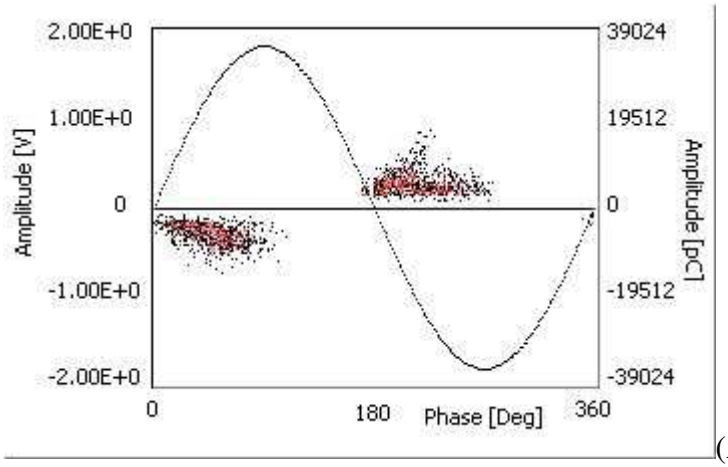
Flat defects can be thought as sort of coins which lay on a plane orthogonal to the applied field. Actually, narrow defects are not considered, because they are not likely to occur in practice. In fact, if the gap is remarkably larger then the smallest of the dimensions of the discharge surfaces, the defect would be, practical speaking, an interface between two layers, with the field parallel to the interface (this case was briefly discussed in the paragraph, dealing with the first level of identification).

Input parameters for the second-level identification / shape:

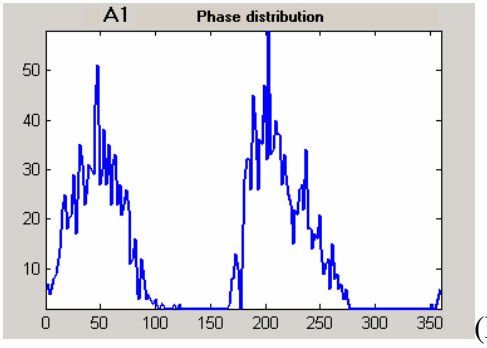
- pF1Pos, pF1Neg
- pF2Pos, pF2Neg
- betaP, betaN

Many other identification markers were taken into account, also resorting to artificial intelligence techniques, and the above mentioned quantities turned out to be the most significant to determine defect shape.

Compact



(A)

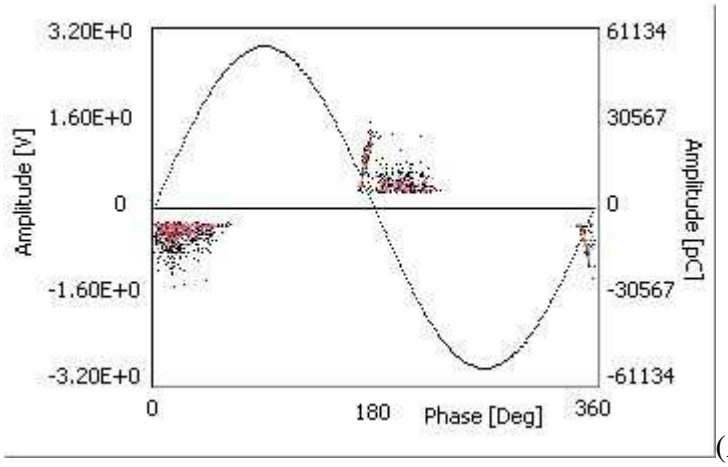


(B)

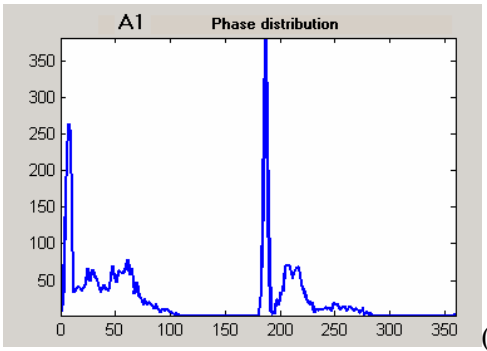
(C)	ND2	ITD	P5	stdQD	FD	beta	betaD	pF1	pF2
Positive Polarity	-0.15	0.04	-0.01	-0.40	12	2.4	0.21	0.48	0.37
Negative Polarity						2.3		0.31	0.36

Fig. 2.47: *example of PD activity in a compact (cylindrical) void, (A) PRPD pattern, (B) phase histogram, (C) table of parameters.*

Flat



(A)



(B)

(C)	ND2	ITD	P5	stdQD	FD	beta	betaD	pF1	pF2
Positive Polarity	0.30	-0.04	-0.01	0-34	-1	1.8	-0.14	0.08	0.16
Negative Polarity						2.1		0.06	0.09

Fig. 2.48: *example of PD activity in a flat defect, consisting of a flat void inside a MV cable joint, (A) PRPD pattern, (B) phase histogram, (C) table of parameters.*

Interpretation criteria for defect shape

Four of the identification markers selected for the identification of defect shape are single-polarity quantities. Indeed, the interpretation criteria here described can be applied to single polarity distributions. Of these quantities, *pF1* and *pF2* belong to the PD phase distribution, while *beta* belongs to the PD amplitude distribution (and accounts for PD amplitude dispersion).
If a defect is flat, it is expectable that different PD sites activate within the defect itself. Of these, one (or more) will be characterized by an inception field which is lower with respect to the others. Hence, in correspondence of these PD sites the probability for a PD event to occur will be higher. This fact has two main consequences on the PD activity:

- A large number of PD events occurs at low phase values (close to *Fimin*) => *pF1* and *pF2* have low values.
- A relatively large dispersion is observed for the PD amplitude distribution => *beta* is low (relatively to values typical of internal PD activities).

Thus, the following criteria can be derived.

- *pF1* low and *pF2* low => flat
- otherwise compact

Field enhancement

In the schematic defect configurations described in the first section, insulation voids were associated to homogeneous local fields, contrarily to point to plane or strongly divergent field configurations. Actually, in the majority of the situations defects characterized by a point to plane

or strongly divergent field configuration exhibit a corona or surface nature, respectively, at the first level of identification. However, these defects may also exhibit an internal behavior, depending on circumstances (as, e.g., overvoltage ratio, degradation stage, etc.). Furthermore, insulation voids themselves may be characterized by an inhomogeneous local field, maybe because of the presence of a metallic protrusion inside the void. Therefore, some criteria to infer local field enhancement must be derived, independently on the nature that the defect exhibits at first level.

Output categories:

- Enhanced LV side → the local field enhances more at the HV side than at the LV side.
- Enhanced HV side → the local field enhances more at the LV side than at the HV side.
- Homogeneous → the local field is homogeneous in the defect.

Input parameters for the second-level identification / field enhabcement:

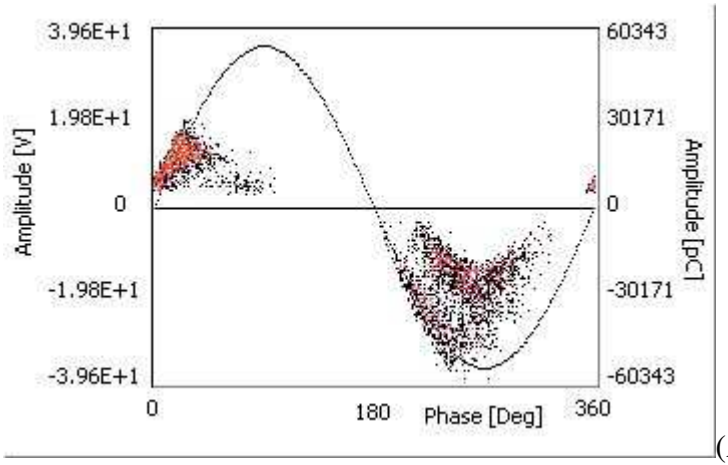
- stdQD
- FD
- betaPos, betaNeg
- betaD

It is worth to recall that parameter *ND2* is involved in the preliminary (second level) check, aimed at recognizing whether the PD activity is bipolar or unipolar (positive or negative).

Homogeneous

As an example of PD activities with homogeneous local field, one can refer to those of Figs. 2.29, 2.30 or 2.32.

Enhanced HV side



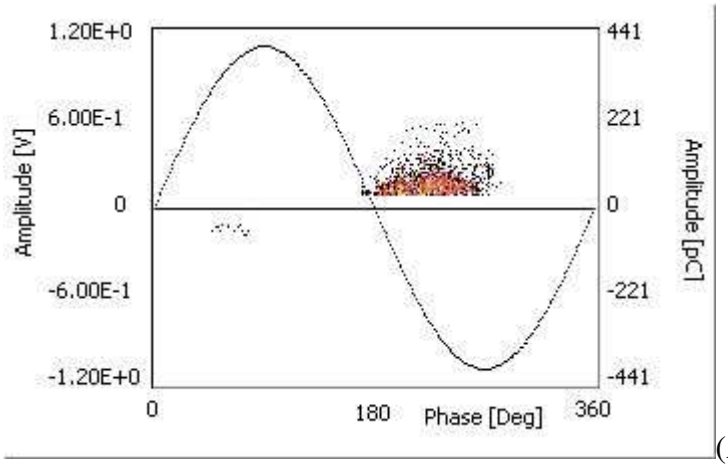
(A)

(B)	ND2	ITD	P5	stdQD	FD	beta	betaD	pF1	pF2

Positive Polarity	-0.22	-0.86	0.19	-0.51	-14	3.7	0.22	0.27	0.27
Negative Polarity						2.9		0.35	0.41

Fig. 2.49: *example of PD activity with field enhancement at HV side, in a rubber-insulated- point to plane specimen, (A) PRPD pattern, (B) table of parameters.*

Enhanced LV side



(A)

(B)	ND2	ITD	P5	stdQD	FD	beta	betaD	pF1	pF2
Positive Polarity	-1	-	-	-	-	-	-	-	-
Negative Polarity						2.2		0.45	0.41

Fig. 2.50: *example of PD activity with field enhancement at LV side, in a generator stator bar with damaged stress grading system, (A) PRPD pattern, (B) table of parameters.*

Interpretation criteria for field enhancement

The fact that the local field is larger in correspondence of one discharge surface with respect to the other has two basic consequences.

- The PD amplitude dispersion is lower where the local field is larger, because discharges move in the direction of decreasing field (and the average time lag is smaller).
- The minimum PD phase is lower for the polarity which is associated to larger field enhancement, since the inception field is lower.

Hence, the following criteria can be derived, for bipolar PD activities, to infer local field enhancement:

- *stdQD* low / *FD* low => field larger at HV side
- *stdQD* high / *FD* high => field larger at LV side

- otherwise (i.e. for mainly symmetric activities) homogeneous

In case of unipolar activities, local field enhancement can be inferred resorting to parameters *betaPos*, *betaNeg*, *betaD*, *corPos* and *corNeg*.

Negative unipolar PD activity.

- (*betaNeg* high or *corNeg*) => field larger at HV side
- *betaNeg* not high and *betaD* high => field larger at LV side
- otherwise homogeneous

Positive unipolar PD activity.

- (*betaPos* high and *betaD* high) or *corPos* => field larger at HV side
- *betaPos* not high and *betaD* low => field larger at LV side
- otherwise homogeneous

Tree growth

Electrical tree formation and growth constitute the final stage of life of an electrical insulation. A treed region has high probability to cause insulation breakdown, in a time that depends on several factors, e.g. the nature of material (thermoplastic/thermosetting/compound, electro-mechanical properties, gas permeability, etc.), service stresses (electrical, thermal, mechanical, polluting agents), random events (overvoltages). The presence of tree growth phenomena indicates, in general, that degradation rate is high, thus a PD activity that can be associated to tree formation should be considered as a weakening phenomenon that, independently of insulation system, will take to breakdown with high probability.
Even if breakdown can approach very fast in most cases, early detection of electrical trees in solid insulation may prevent from catastrophic events, e.g. failure of a high-voltage transmission cable, of a large transformer or rotating machine. Technical-economical optimised design of maintenance operations can, hence, be achieved if the presence of an electrical tree is diagnosed in due time.
For further details please refer to [13, 38, 41], which describe artificial intelligence tools, applied to an innovative system for PD detection, that allows electrical tree identification. In the following, basic concepts about tree identification are discussed.

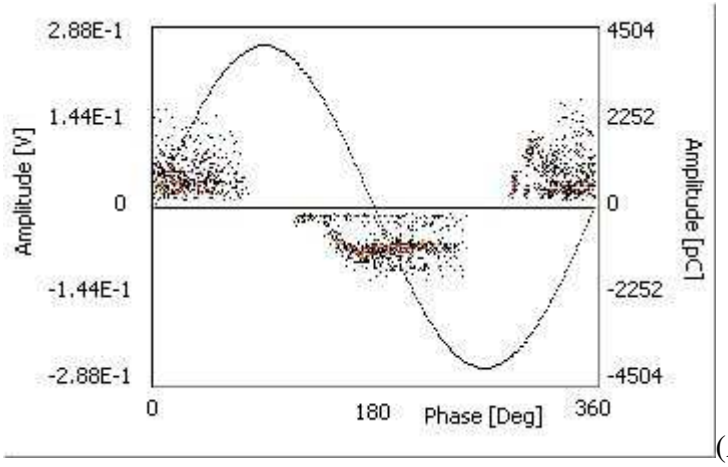
Output categories:

- Tree (branch like / bush like typology; void present / no void present).
- No tree.

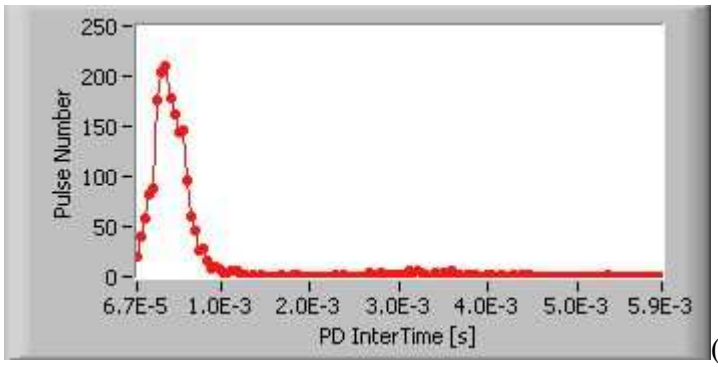
Input parameters for the second-level identification / field enhabcement:

- P1
- P2
- P3
- P4
- AgPos, AgNeg
- FgPos, FgNeg

In Fig. 2.51, a PD activity is shown, which is associated to tree growth in a Cigre 2 specimen, with void in front of needle tip.



(A)



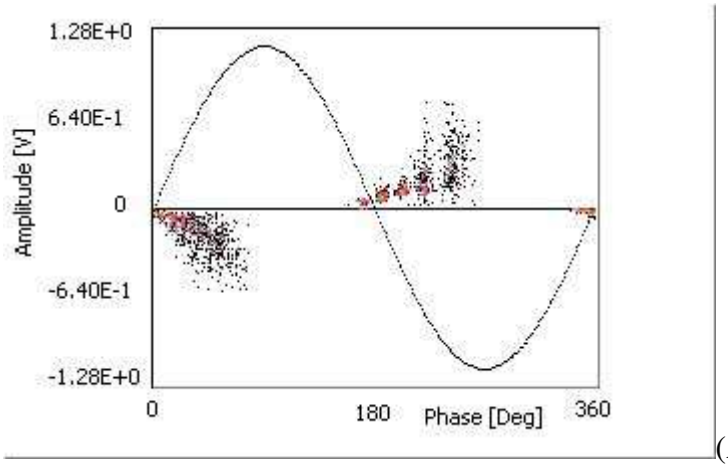
(B)

(C)	P1 [s]	P2 [s]	P3	P4	Ag	Fg
Positive Polarity	0.00013	0.00057	0.03	0.03	0.3	0
Negative Polarity					1	2

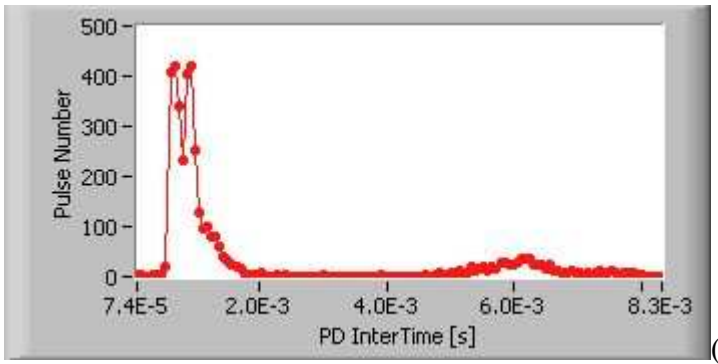
Fig. 2.51: *example of PD activity associated to tree growth, in a Cigre 2 specimen with void in front of needle tip, (A) PRPD pattern, (B) intertime histogram, (C) table of parameters.*

On the contrary, in Fig. 2.52, a PD activity is shown, which is associated to tree growth in a Cigre 2 specimen, where the presence of void in front of needle tip was avoided. Such an electrical tree was

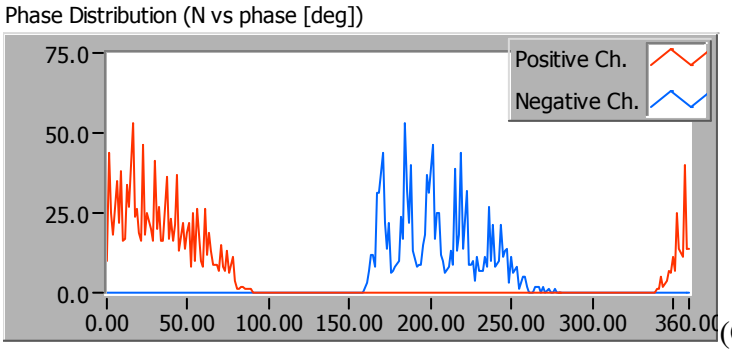
of the branch-like typology, as was verified by means of optical detection equipment.



(A)



(B)

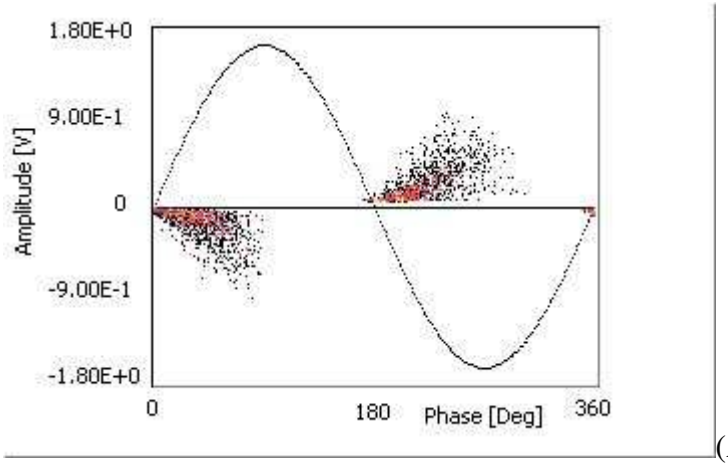


(C)

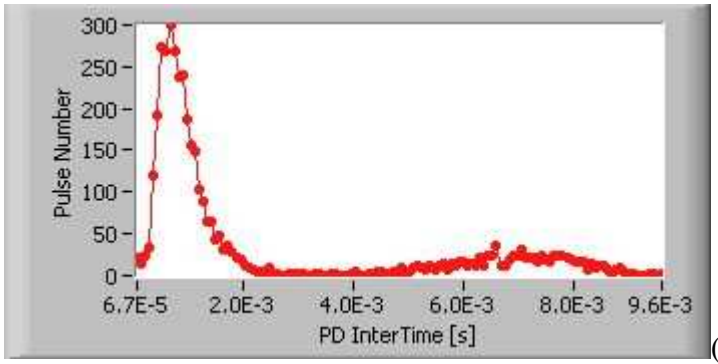
(D)	P1 [s]	P2 [s]	P3	P4	Ag	Fg
Positive Polarity	0.00059	0.00619	0.01	0.10	0	0
Negative Polarity					0	5

Fig. 2.52: *example of PD activity associated to tree growth, in a Cigre 2 specimen without void in front of needle tip (branch-like tree), (A) PRPD pattern, (B) intertime histogram, (C) phase histogram, (D) table of parameters.*

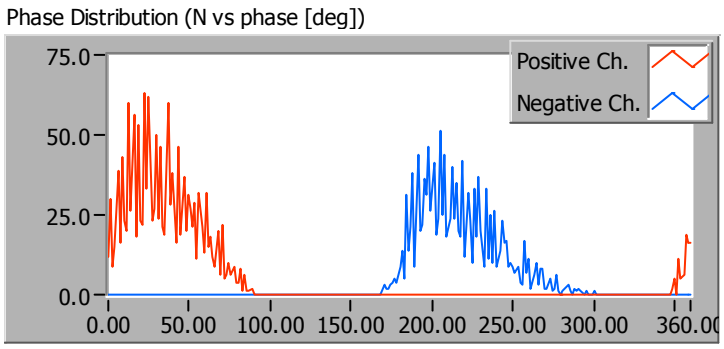
In Fig. 2.53, a PD activity is shown, which is associated to bush-like tree in a Cigre 2 specimen, where the presence of void in front of needle tip was avoided. The bush-like nature of the tree was verified by means of optical detection equipment.



(A)



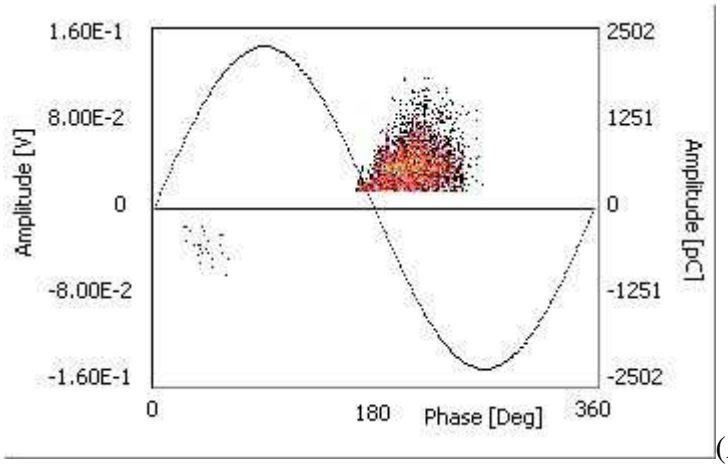
(B)



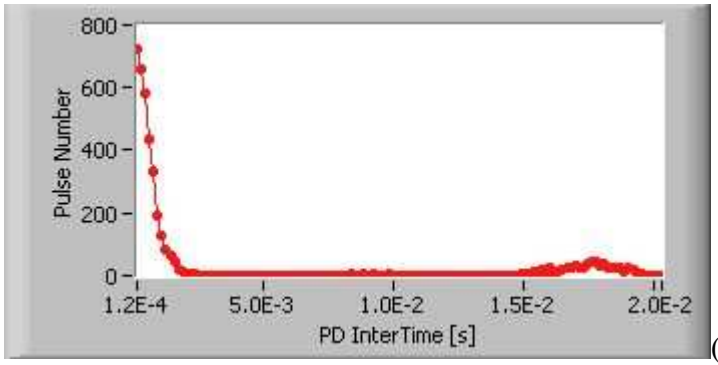
(D)	P1 [s]	P2 [s]	P3	P4	Ag	Fg
Positive Polarity	0.00054	0.00760	0.02	0.35	0	0
Negative Polarity					0	2

Fig. 2.53: *example of PD activity associated to tree growth, in a Cigre 2 specimen without void in front of needle tip (bush-like tree), (A) PRPD pattern, (B) intertime histogram, (C) phase histogram, (D) table of parameters.*

Finally, Fig. 2.54 shows a PD activity in a Cigre 2 specimen, with void in front of needle tip, is shown, which is not associated to tree growth.



(A)



(B)

(C)	P1 [s]	P2 [s]	P3	P4	Ag	Fg
Positive Polarity	0.00026	0.01702	1	0.03	0	0
Negative Polarity					0	2

Fig. 2.54: *example of internal PD activity before the inception of tree growth, in a Cigre 2 specimen with void in front of needle tip, (A) PRPD pattern, (B) intertime histogram, (C) table of parameters.*

Interpretation criteria for tree growth

PD activities associated to electrical trees are internal, at the first level of identification. They can be distinguished from activities in voids resorting to the evaluation of the intertime distribution. As can be seen from the comparison of Figs. 2.51, 2.52 and 2.53 on one side, and Fig. 2.54 on the other side, when discharges occur in a tree channel, the intertime between consecutive discharges has a distribution that starts from smaller values with respect to the distribution obtained before tree inception (the average intertime after tree inception is significantly lower than the semi-period of the supply voltage waveform). In addition, the smallest intertime values are not those associated with the highest probability. Such experimental evidence fits the physical phenomena involved in tree growth [51-53]. In fact, during tree growth a certain amount of charge is moved by the applied field along the ionised path (that corresponds to a volume in the solid insulation material characterized by low density). Such a charge transfer takes place by steps, as long as charge moves along a branched channel, and at the edge of each branch it stops until the electric field, in that

point, enhances enough for a further step. Hence, it can be expected that a sequence of PD pulses is generated at each charge shift along a branched channel. This phenomenon must take place while the applied electric field has the same sign, that is, within one half period; after field polarity reversal a similar phenomenon occurs, but in the opposite direction. Thus, it is reasonable that the average intertime is lower than the half period. However, high repetition rate discharges may occur also in internal defects (cavities), characterized by a high overvoltage ratio and a large number of free electrons. In this case, the smallest intertime occurs when a PD takes place as soon as the electric field exceeds the inception field (hence, it depends on the values of the inception field and of the residual field, that can be considered constant for one acquisition). If free electrons are abundant, the shortest intertime tends to be the most probable. On the contrary, during tree growth, the smallest PD intertime depends on the tree geometry and the amount of transferred charge; hence, there is no reason why the shortest intertime should be the most likely.
Hence, a PD activity associated to tree growth is characterized by low values of parameters P1, P2, P3 and P4.

As regards the discrimination between tree typologies, namely branched trees and bushed trees, it can be observed that the main difference between these two typologies was found in the phase distribution histogram, which, actually, is the projection of the PRPD pattern in the "phase - number of discharges" plane. In fact, it can be noted that the distribution of negative PD for branch-like trees is characterized by well distinct peaks (Fig. 2.52C), recognizable also in the PRPD pattern as humps (Fig. 2.52A), which are not observable for bush-like trees (Figs. 2.53).
It is worth to underline that in the tests illustrated in Figs. 2.52, 2.53 (specimens without void in front of needle tip) the needle electrode was connected to the ground. Therefore, negative PD (roughly, PD occurred in the second half period of the applied voltage) originate in the tree channels and move toward the needle. In this light, the presence of peaks in the negative phase distribution may be associated to the presence of a limited number of distinct sources for PD originating in the tree channels, each one characterized by a specific inception field. In addition, different sources might affect each other, perhaps causing a further delay in the inception of the successive activity. In this light, it is reasonable that PD activities in bush-like trees do not manifest such peaks. In fact, bush-like trees, presenting a large number of branches, are characterized by many PD sources, which are activated with very short delay one from the other.
Thus, quantity Af, which counts the number of peaks in the phase distribution histogram, seems to be able to identify properly the tree typology. According to experimental evidence, a value of Af larger than 4, for at least on of the polarity distributions, is typical of branched trees; value smaller than 2 are typical of bushed trees.

As regards the discrimination between tree activities with void and without void, as for the identification of the tree typology, the time of occurrence histogram does not provide significant information. As regards the first level of identification, specimens with void are characterized by a

higher membership to the internal category, than specimens without void (although this category is predominant also for the latter ones). This behavior is reasonable, because the void provides a further contribution to internal nature, but it does not constitute, itself, a criterion to distinguish the two situations. The second difference regards the amplitude distribution. In fact, in specimens with void PD amplitude values gather in two groups, forming two humps. This is probably caused by the fact that all discharges in the void have a similar amplitude, which is small with respect to the average amplitude of PD in tree channels. On the contrary, in specimens without void the amplitude distribution is homogeneous, i.e. without humps.
Hence, quantity Ag may be used to investigate the presence of void in correspondence of a an electrical tree. Quantity Ag ranges in [0 , 1]; values close to 1 indicate the presence of humps in the amplitude distribution. According to experimental evidence, as an average, half of the acquisitions on specimens with void are characterized by high values of Ag; on the contrary, Ag is 0 for all the acquisitions of specimens without void.

Third level – reference PD activities and results

Third level of identification is tailored for a specific typology of industrial apparatus. In this thesis, focus was made on rotating machines, in particular large generators.
Seven output categories were defined. So far, only the first six were provided with automatic identification criteria. These criteria were developed according to previous experiences [55-57], as well as to a large experimental activity, which enabled to set up a diagnostic database tailored for this identification issue. The developed identification criteria do not necessarily have a direct link to physical aspects of PD activities in generator defects, which are not even easily reproducible with laboratory models. Therefore, such criteria will be reported in the following as short accounts.

Distributed microvoids
Defects internal to the insulation, consisting of small voids. This kind of defect is present in any machine due to unavoidable imperfections in the impregnation process. During off line tests, the discharges associated to this kind of defects usually appear at the lowest voltage and, at higher voltage, may be masked by other PD activities.

Embedded delamination
Detachments between mica foils, internal to the insulation. These are flat voids (the size orthogonal to the electric field is much larger than that parallel to the electric field), embedded in the insulation. These defect can be particularly dangerous since the air interface increases the thermal conductivity of the insulation system, possibly leading to overheating and hot spots.

Conductor side delamination

Detachments of the insulation from HV electrode (copper part of the bar). These defects consist of flat voids placed between HV electrode and insulation. Also in this case, overheating (hot spots) can happen.

Slot discharges

Discharges between the bar/coil insulation and the stator iron core. These discharges firstly erode the semiconductive coating, then the insulation.

Semicon/grading discharges (corona)

Discharges occurring at the junction of the semi-conductive and stress grading parts in the overhang, in the presence of pollution / degradation. Machines in good condition can withstand for long times these discharges, which are tangential to the insulation surface.

Significant deterioration

Generic internal defects which may be not recognized as any of the above mentioned typologies, but provide significant PD activity (above a pre-selected threshold).

Bar to bar

These discharges occur in the air gap between bars of different phases. PD are orthogonal to insulation surface and may deteriorate the insulation system faster than corona discharges. These discharges usually form PD pattern composed of several "clouds" of discharge having similar amplitude. At present, no automatic recognition is available for this discharge category.

Input parameters for the third-level identification / rotating machines:

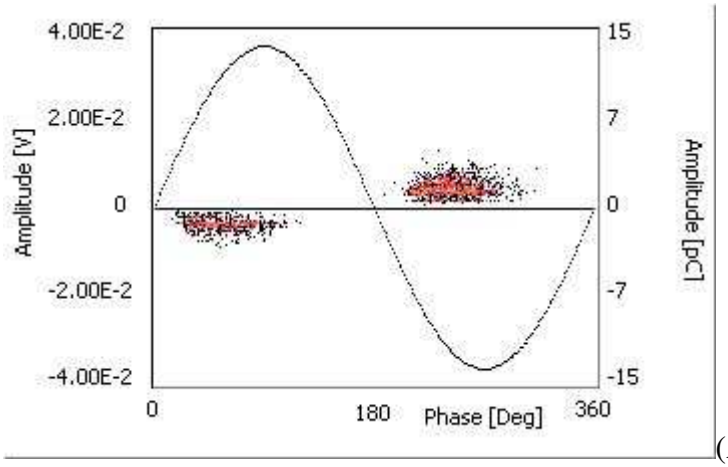
- ND
- QD
- FD
- stdQD
- SIPos, SINeg
- betaPos, betaNeg
- FimeanDPos, FimeanDNeg

All of these quantities have been previously defined, except for SIPos and SINeg. Let "IntPos" and "SurPos" be the first level identification outputs relevant to internal and surface phenomena, respectively, for positive polarity PD. Likewise, IntNeg and SurNeg can be considered for the distribution of negative PD. SIPos and SINeg result from equations (2.23) and (2.24), respectively.

$$SIPos = \frac{SurPos}{IntPos + SurPos} \tag{2.23}$$

$$SINeg = \frac{SurNeg}{IntNeg + SurNeg} \tag{2.24}$$

Distributed microvoids



(A)

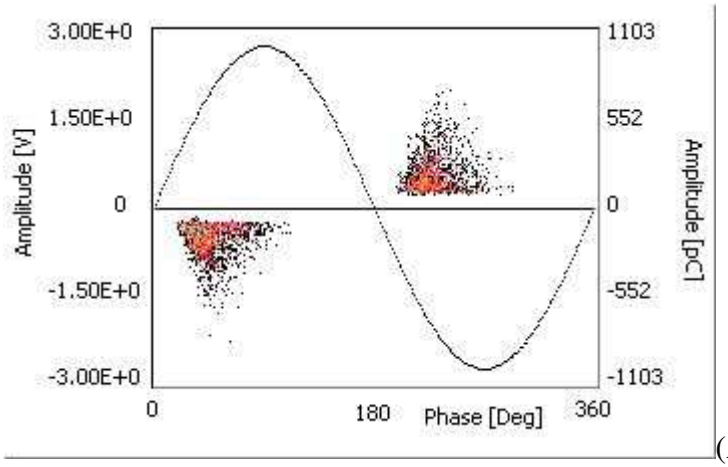
(B)	ND	QD	FD	stdQD	Fimin	SI	beta	FimeanD
Positive Polarity	0.42	-0.13	-5	-0.21	18	0.00	2.9	-26
Negative Polarity					24	0.24	2.7	-11

Fig. 2.55: *example of PD activity relevant to distributed microvoids, (A) PRPD pattern, (B) table of parameters.*

Qualitative description of the PD activity related to category distributed microvoids.

- Positive and negative distributions are symmetric.
- The dispersion of PD height is low.
- Phase intervals of PD distributions are centered around 45 and 225 degrees.
- Inception phase does not take place before the zero crossing of the applied voltage.

Embedded delamination



(A)

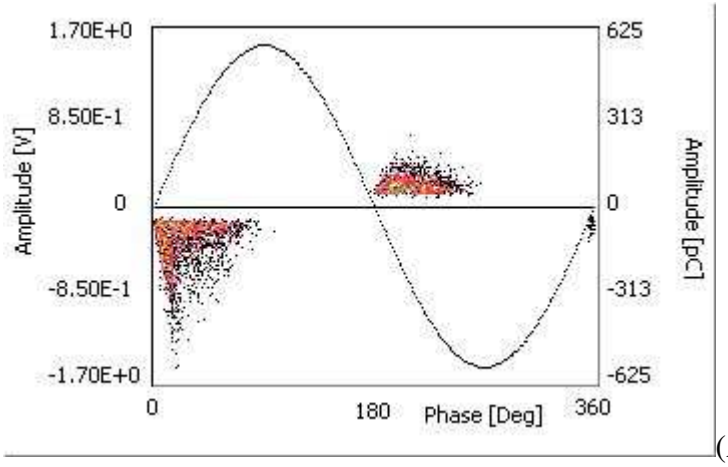
(B)	ND	QD	FD	stdQD	Fimin	SI	beta	FimeanD
Positive Polarity	0.57	0.40	2	-0.13	20	0.43	2.1	-5
Negative Polarity					18	0.41	2.1	-7

Fig. 2.56: *example of PD activity relevant to embedded delamination (between mica foils), (A) PRPD pattern, (B) table of parameters.*

Qualitative description of the PD activity related to category embedded delamination.

- Positive and negative distributions are symmetric.
- The dispersion of PD height is high.
- Phase intervals of PD distributions are centered around 45 and 225 degrees.
- Inception phase may take place before the zero crossing of the applied voltage.

Conductor delamination



(A)

(B)	ND	QD	FD	stdQD	Fimin	SI	beta	FimeanD
Positive Polarity	0.49	0.55	-8	0.68	19	0.79	1.7	9

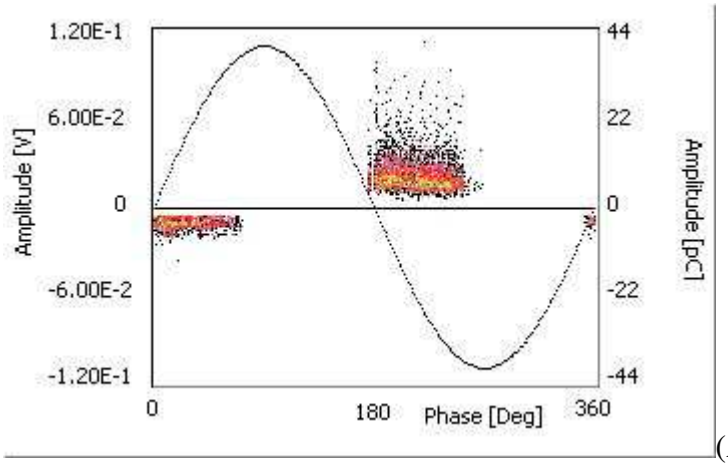
Negative Polarity					27	0.01	2.9	-11

Fig. 2.57: *example of PD activity relevant to conductor delamination, (A) PRPD pattern, (B) table of parameters.*

Qualitative description of the PD activity related to category conductor delamination.

- Positive and negative distributions are not symmetric.
- The dispersion of PD height is higher for positive PD than for negative PD.
- Phase intervals of PD distributions are centered around 45 and 225 degrees.
- Inception phase may take place before the zero crossing of the applied voltage.

Slot discharges



(A)

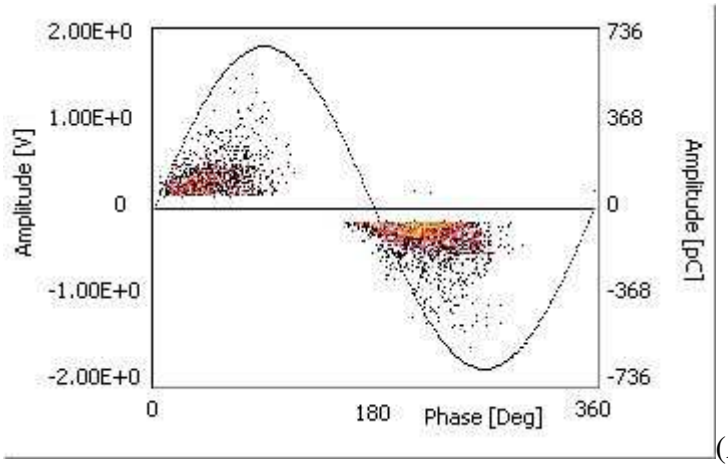
(B)	ND	QD	FD	stdQD	Fimin	SI	beta	FimeanD
Positive Polarity	0.28	-0.56	-3	-0.65	-7	0.10	3.0	5
Negative Polarity					-5	0.24	2.4	-4

Fig. 2.58: *example of PD activity relevant to LV delamination (slot discharges), (A) PRPD pattern, (B) table of parameters.*

Qualitative description of the PD activity related to category slot discharges.

- Positive and negative distributions are not symmetric.
- The dispersion of PD height is higher for negative PD than for positive PD.
- Phase intervals of PD distributions are centered around 45 and 225 degrees.
- Inception phase of positive discharges is, most likely, lower than that of negative discharges.

Semicon/grading discharges (corona)



(A)

(B)	ND	QD	FD	stdQD	Fimin	SI	beta	FimeanD
Positive Polarity	0.29	0.10	26	0.05	6	0.66	2.0	-32
Negative Polarity					-20	0.00	1.9	-23

Fig. 2.59: *example of PD activity relevant to stress grading, (A) PRPD pattern, (B) table of parameters.*

Qualitative description of the PD activity related to category slot discharges.

- Positive and negative distributions are not symmetric.
- The dispersion of PD height is higher for negative PD than for positive PD.
- The number of negative PD is higher than that of positive PD.
- Phase intervals of PD distributions are centered around 45 and 225 degrees.
- Inception phase of negative discharges is lower than that of positive discharges.

References section 2

[1] CIGRE Report WG 21.03, Recognition of Discharges, Electra N. 11, December 1969.
[2] T. Okamoto, T. Tanaka, “Novel Partial Discharge Measurement Computer-Aided Measurement System”, IEEE Trans. on Electrical Insulation, Vol.21, N.1, pp.1015-1019, December 1986.
[3] F.H. Kreuger, E. Gulski, A. Krivda, "Classification of Partial Discharges", IEEE Trans. on Electrical Insulation, Vol. 28, N. 6, pp.917–931, December 1993.
[4] A. Krivda, "Automated Recognition of Partial Discharges", IEEE Trans. on Dielectrics and Electrical Insulation, Vol. 2, N. 5, pp. 796–821, October 1995.
[5] T.Okamoto, T.Tanaka, "Novel Partial Discharge Measurement Computer-Aided System", IEEE Trans. on Electrical Insulation, vol.21, n.6, pp.1015-1019, December 1986.
[6] E.Gulski, F.H.Kreuger, "Computer-aided recognition of discharge sources", IEEE Trans. on Electrical Insulation, vol.27, n.2, pp.469-479, April 1992.
[7] E.Gulski, *Computer-Aided Recognition of PD Using Statistical Tools*, Ph.D. Thesis, Delft University Press, 1991.
[8] E. Gulski, H.P. Burger, G. H. Vaillancourt, R. Brooks, "PD Pattern Analysis During Induced Test of Large Power Transformers", IEEE Trans. on Dielectrics and Electrical Insulation, Vol.7, N.1, pp.95-101, February 2000.
[9] H.G Kranz, "Diagnosis of PD Signals Using Neural Networks and Minimum Distance Classification", IEEE Trans. on Electrical Insulation, Vol. 28, N.6, pp.1016-1024, December 1993.
[10] C. Cachin, H. J. Wiesman, "PD Recognition with Knowledge-Based Preprocessing and Neural Networks", IEEE Trans. on Dielectrics and Electrical Insulation, Vol.2, n.4, pp.578-589, August 1995.
[11] L. Satish, B. I. Gururaj, “Use of Hidden Markov Models for Partial Discharge Pattern Classification”, IEEE Trans. on Electrical Insulation, Vol.28, N.2, pp.172-182, April 1993.
[12] B. Fruth, J. Fuhr, "Partial Discharge Pattern Recognition - A Tool for Diagnosis and Monitoring of Ageing", CIGRE, Paris, France, Paper 15/33-12, August 1990.
[13] C. Hudon, M. Belenc, "The Importance of Phase Resolved PD Pattern Recognition for On-Line Generator Monitoring", IEEE ISEI, pp.296-300, Arlington (Virginia), USA, June 1998.
[14] A.Contin, G.C.Montanari, C.Ferraro, “Partial-Discharge Source Recognition by Weibull Processing of the Pulse-Height Distributions”, IEEE Trans. on Dielectrics and Electrical Insulation, Vol.7, N.1, pp.48-58, February 2000.
[15] N. Hozumi, T. Okamoto and T. Imajo, "Discrimination of Discharge Patterns Using a Neural Network", IEEE Trans. on Dielectrics and Electrical Insulation, Vol. 27, N.3, pp.550–556, June 1992.
[16] C. Cachin, H. J. Wiesman, "PD Recognition with Knowledge-Based Preprocessing and Neural Networks", IEEE Trans. on Dielectrics and Electrical Insulation, Vol.2, n.4, pp.578-589, August 1995.

[17] M. Hoof, B. Freisleben and R. Patsch, "Partial Discharge Source Identification with Novel Discharge Parameters Using Counterpropagation Neural Networks", IEEE Trans. on Dielectrics and Electrical Insulation, Vol.4, N.1, pp.17–32, February 1997.
[18] A. Krivda, E. Gulski, L. Satish, W. S. Zaengl: "The Use of Fractal Features for Recognition of 3-D Discharges", IEEE Trans. on Dielectrics and Electrical Insulation, Vol.2, N.5, pp.889-892, October 1995.
[19] M. Cacciari, A. Contin, G. C. Montanari, "Use of a Mixed-Weibull Distribution for the Identification of Partial Discharge Phenomena", IEEE Trans. on Dielectrics and Electrical Insulation, Vol.3, N.6, pp.1166-1179, December 1995.
[20] A. Contin, G. C. Montanari, A. Cavallini, "Random Sampling Techniques for PD-Pulse Shape Analysis", IEEE Trans. on Dielectrics and Electrical Insulation, Vol.7, N.1, pp.30-39, February 2000.
[21] Contin, G.C. Montanari, M. Conti, M. Cacciari, "An invariant diagnostic marker for the identification of partial discharge sources in electrical apparatus", IEEE ICSD, pp. 287-290, Eindhoven, Thye Netherlands, June 2001.
[22] A.. Cavallini, M. Conti, G.C. Montanari, A. Contin, "Indexes for the recognition of insulation system defects derived from partial discharge measurements", IEEE ISEI, pp. 511-515, Boston, USA, April 2002.
[23] G. C. Stone, "Tutorial on Rotating Machine Off-Line and On-line Partial Discharge Testing", CIGRE/EPRI Colloquium on Maintenance of Motors and Generators, Vol. 3, Florence, Italy 1997.
[24] A.Contin, G.C.Montanari, A.Cavallini, G.Pasini, M.Conti, "An Integrated Diagnostic Tool Based on PD Measurements", EIC Vol., N., pp. 219-224 -, Cincinnati (USA), Ottobre 2001.
[25] A. Cavallini, M. Conti, D. Fabiani and G. C. Montanari, “A summary of Research Activities on Diagnostic at LIMAT”, IEEE SDEMPED, pp. 107-109, Italy, Grado (GO), Settembre 2001.
[26] A. Cavallini, M. Conti, D. Fabiani and G.C. Montanari, “Evaluation of Corona-resistent Magnet Wires through Partial Discharge and Space Charge Measurements”, IEEE EIC/EMCWA, Cincinnati, OH (USA), Ottobre 2001.
[27] M. De Nigris, G. Rizzi, F. Ombello, F. Puletti, A. Cavallini, G.C. Montanari, M. Conti, “Cable Diagnosis Based on Defect Location and Carachterization Through Partial Discharge Measurements”, IEEE , CIGRE, Paris, September 2002.
[28] A.Contin, M.Conti, A.Cavallini, G.C.Montanari, "Inferring PD Identification Through Fuzzy Tools", IEEE-CEIDP 2002 Annual Report, pp. 668-702, Cancun, Mexico, October 2002.
[29] A.Contin, M.Conti, A.Cavallini, G.C.Montanari, R.Schifani, R.Candela, P. Romano, "Searching for PD-Based Indexes able to Infer the Location of Internal Defects in Insulation", IEEE-CEIDP 2002 Annual Report, pp. 703-706, Cancun, Mexico, October 2002.
[30] A. Cavallini, M. Conti, G. C. Montanari, “On-Field Partial Discharge Measurements: a Feasible Approach to Noise rejection and Defect Assessment”, INSUCON 2002, pp. 103-108, Berlino, Giugno 2002.
[31] A. Cavallini, M. Conti, G. C. Montanari, "On-field partial discharge measurements: a feasible

approach to noise rejection and defect assessment", INSUCON 2002, pp. 183-188, Berlin, Germany, June 2002.
[32] A. Contin, A. Cavallini, M. Conti, G. C. Montanari, “Advanced PD Inference in On-Field Measurements. Part 2: Identification of Defects in Solid Insulation Systems”, IEEE Trans. on Dielectrics and Electrical Insulation, Vol. 10, n. 3, pp. 528-538, June 2003.
[13] A. Cavallini, M. Conti, G. C. Montanari, A. Contin, F. Guastavino, F. Ombello, "Early detection of electrical tree through advanced PD measurement inference techniques", JICABLE, pp. 612-616, Versailles, France, June 2003.
[33] Hudon, A. Contin, M. Bélec, A. Cavallini, G. C. Montanari, D.N. Nguyen, M. Conti , "Evolution in automatic phase resolved partial discharge pattern recognition for rotating machine diagnosis ", IEEE ISH, pp., Delft, Holland, September 2003.
[34] A. Cavallini, M. Conti, A. Contin, G. C. Montanari, F.Puletti, "Partial discharge cross-talk recognition in rotating machines by pulse-shape analysis: preliminary results ", IEEE EIC, Indianapolis, USA, September 2003.
[35] G. C. Montanari, A. Cavallini, G. Mazzanti, M. Conti, F. Ciani, S. Serra, "First electron availability and PD generation in insulation cavities", IEEE CEIDP, pp. , Albuquerque, USA, October 2003.
[36] A. Cavallini, M. Conti, G. C. Montanari, F. Puletti, A. Contin, "A new methodology for the identification of PD in electrical apparatus: properties and applications", IEEE Trans. on Dielectrics and Electrical Insulation, submitted for revision.
[37] A. Cavallini, F. Ciani, M. Conti, P. F. H. Morshuis, G. C. Montanari, "Modeling memory phenomena for partial discharge process in insulation cavities", IEEE CEIDP, pp. , Albuquerque, USA, October 2003.
[38] A. Cavallini, M. Conti, G. C. Montanari, C. Arlotti, A. Contin, "PD inference for the early detection of electrical tree in insulation systems", IEEE Trans. on Dielectrics and Electrical Insulation, Vol. 10, n. 6, pp., December 2003.
[39] S. Pöyhönen, M. Conti, A. Cavallini, G. C. Montanari, F. Filippetti, “Insulation defect localization through partial discharge measurements and numerical classification”, submitted to IEEE ICSD 2004, Toulouse, France, November 2003.
[40] A. Cavallini, F. Ciani, M. Conti, G.C. Montanari, “The effect of space charge on phenomenology of partial discharges in insulation cavities”, submitted to IEEE ICSD 2004, Toulouse, France, November 2003.
[41] M. Conti, A. Cavallini, G.C. Montanari, F. Guastavino, “Identification of electrical tree growth in insulation systems by fuzzy logic techniques based on partial discharge acquisition”, submitted to IEEE ICSD 2004.
[42] A. Contin, A. Cavallini, M. Conti, G. C. Montanari, “Transition from PD to electrical tree in solid insulation inferred through enhanced detection tools”, submitted to IEEE ICSD 2004.
[43] M. Conti, A. Cavallini, G.C. Montanari, A. Contin, “A new algorithm for the identification of defects generating partial discharges in rotating machines”, to be submitted to IEEE ISEI 2004.

[44] M. Conti, G.C. Montanari, A. Cavallini, F. Ciani, “Partial discharge measurements to detect the threshold for electret behaviour in cellular polypropylene”, to be submitted to Journal of Applied Physics.
[45] M. Conti, A. Cavallini, G.C. Montanari, A. Contin, “Improved identification level for defects generating partial discharges in insulation systems”, to be submitted to IEEE CEIDP 2004.
[46] A.Cavallini, M.Conti, A.Contin, G.C.Montanari, S. Guillame, “Automatic Noise Rejection Tools in On-Field PD measurements”, *V° Volta Colloquium on Partial Discharges*, Como, Italy, September 2001.
[47] A. Cavallini, A. Contin, G. C. Montanari, F. Puletti, "Advanced PD Inference in On-Field Measurements. Part.1: Noise Rejection", IEEE Trans. on Dielectrics and Electrical Insulation, 2003.
[48] C. S. Kim, T. Mizutani, S. Kobayashi, N. Morimoto, S. Tsuji, H. Saitou, “PD characteristics of a void between semiconducting layer and insulating layer”, IEEE ICPADM 2003, Vol. 1, pp. 315-318, Nagoya, Japan, June 2003.
[49] B.A. Fruth, D.W. Gross, “Modelling of streamer discharges between insulating and conducting surfaces”, IEEE-ICSD, pp. 350-355, Leicester, UK, June 1995.
[50] A. Contin, M. Conti, A. Cavallini, G. C. Montanari, R. Schifani, R. Candela, P. Romano, "Searching for PD-based indexes able to infer the location of internal defects in insulation", IEEE-CEIDP 2002 Annual Report, pp.703-706, Cancun, Mexico, October 2002.
[51] L. A. Dissado, “Understanding electrical trees in solids: experiment to theory”, IEEE ICSD, pp. 15-26, Eindhoven, the Nederlends, June, 2001.
[52] L. A. Dissado, J. C. Fothergill, *Electrical degradation and breakdown in polymers*, Ed. Peter Peregrinus Ltd., London, 1992.
[53] J.V. Champion, S.J. Dodds, “Modelling partial discharges in electrical trees”, IEEE ICSD, pp. 291-294, June 1998, Leicester.
[54] Suwarno, H. Hichicava, Y. Susuoki, T. Mizutani, K. Uchida, "Partial Discharge Patterns of Electrical Treeing in Polyethylene", IEEE ICPADM, pp.379-382, Brisbane, Australia, July 1994.
[55] H. Shu, I. J. Kemp, “Pulse Propagation in Rotating Machines and its Relationship to PD Measurements”, IEEE ISEI, pp.411-414, Baltimore, USA, June 1992.
[56] J.P. Zondervan, E. Gulski, J. J. Smit, "Fundamental Aspects of PD Patterns of On-Line Measurements on Turbogenerators", IEEE Trans. on Dielectrics and Electrical Insulation, Vol.7, N.1, pp.59-70, February 2000.
[57] R. Miller, W. K. Hogg, “Pulse Propagation of Slot and Internal Discharges in Stator Windings of Electrical Machines”, ISH, paper 63.05, Athens, Greece, August 1983.

Section 3

Section 3 describes a logical / mathematical approach to fuzzy logic and artificial intelligence, which is suited to PD interpretation, allowing to transfer human reasoning to the machine and to derive rules and concepts directly from training data. In particular, decision trees, concept learning and instance based learning algorithms supported by fuzzy logic will be described.

In this section:

- Introduction
- General terms of the problem
- Fundamentals of fuzzy logic
- Description of “linguistic” and “matrix” fuzzy engine
- Fundamentals of artificial intelligence
- Decision trees
- Concept learning
- Instance based learning and synthesis of different approaches
- References

Introduction

Dealing with defect identification, it can be assumed that output categories (categories of insulation defects) are selected according to a specific diagnostic strategy, which, in general, provides a variety of tasks. Accordingly, a large number of quantities shall be derived from the acquired data in the attempt to characterize the PD activity with respect to specific tasks. Goal of the identification process is to associate a PD acquisition to the output categories, through a procedure which can be run automatically. In this way, information about the PD generating defect is provided, which will constitute the basis for further diagnostic evaluation.
In order to make the identification procedure automatic, an inference machine must be provided with an algorithm that associates input quantities to output categories (identifier). How to develop such an algorithm is discussed in this report.
Before describing technical aspects, it is important to distinguish two situations:

- *Machine teaching*. The developer himself is able to infer the output categories, given specific quantities, on the basis of his own experience and understanding of PD phenomena (related to the considered task). In this case, the problem for the developer is to transfer his experience to the machine.
- *Machine learning*. The developer himself is not confident about the identification result, and, perhaps, is not even confident about which quantities to use, because of a lack of experience and understanding of the phenomena associated to the considered task. In this case, the problem for the developer is to train the machine in such a way that it can autonomously find the optimal quantities and algorithm, with respect to a set of training examples.

In the case of machine teaching, the best approach to develop the identifier would be the one that allows the machine to reproduce human reasoning as much as possible.
In the case of machine learning, a crucial aspect is constituted by the reliability of the training dataset; in addition, the developer is supposed to select the most convenient artificial intelligence (AI) technique and to adapt it to the specific task.
In both cases, it would be desirable that the identifier be provided with enough flexibility to allow any user (thus, not necessarily the developer) to adjust and optimize the algorithm along with his knowledge improvements and / or according to new training examples. Such a flexibility would allow a customization of the identifier, too.

The primary, fundamental step for identification is the selection of output categories. Such a selection often imposes a trade off: on one hand, output categories should provide useful information for maintenance planning (and further diagnostic evaluation); on the other hand, they should be derivable from the acquired data, that is, they must be related to observable PD phenomena. If the former condition is not verified, the identification procedure does not have

practical application; if the latter condition is not verified, a more complex question arises. In fact, quantities extracted from PD activity in general are partially related to the output categories: the stronger the relation between derived quantities and output categories (input and output of the identifier), the more the identifier results simple and reliable. So, the question is: for a given task, how far is it convenient to proceed in the development of the identifier, rather than to seek for new, more significant quantities and / or make adjustments to the output categories? It is opinion of the author that the validity of the identifier is given by its ability to provide a general description of the output categories in terms of the input quantities, regardless to its performance on the training dataset. Obviously, its performance on the training dataset will determine its actual applicability, that is, if it performs too poorly, it is not useful. In terms of concept learning, the identifier is valid when it formulates general concepts about the given output categories using the input quantities. In the case of machine teaching, such a validity is highly expectable, if the experience transferred from the developer to the machine is based on a true comprehension of the PD phenomena. In the case of machine learning, the same kind of validity is not guaranteed. In fact, it is possible that the identifier manages to identify the training examples with a good performance, just because it adapts to those specific examples, loosing general validity (overfitting). This phenomenon could be accepted only if the training dataset was really reliable and representative of all possible situations. In the case of insulation defect identification, it seems very unlikely that the training dataset can be endowed with such characteristics. Therefore, machine learning should be thought as a tool which can help in understanding PD phenomena, rather than a way to bypass such a understanding. In this light, human effort would be aimed mainly at obtaining reliable example data according to output categories and at the extraction of significant quantities, while AI techniques would determine not only the best identifier, but also the most appropriate concepts which describe output categories in terms of input quantities, thus providing the developer with a feedback about the followed approach and his speculations.

In this work, the following solutions were adopted:

- Machine teaching: a typology of fuzzy engine was set up, in order to provide the maximum flexibility, reusability and extendibility.
- Machine learning: two approaches were followed, one based on decision tree theory, able to select the most significant quantities and to optimize their discretization (fuzzyfication), the other specifically aimed at concept learning. The use of Monte Carlo methods and / or genetic algorithms will be described as well, because it helps in the optimization of AI algorithms themselves.

Furthermore, it will be shown how all the proposed approaches are compatible, and meet in one general identifier typology, allowing further improvement and customization.

In order to find the most suited approach to develop machine learning and teaching algorithms, many techniques and approaches were considered, regarding both fuzzy logic [1, 2, 4, 5 - 19] and artificial intelligence [5 – 14].

General terms of the problem

For a specific identification task, a certain number of output categories and of input parameters, i.e. those quantities selected for that identification task, will be given.

- Number of output categories: N_{out}
- Number of input parameters: p

As regards input parameters (x_1,…,x_n), it can be assumed that they are real numbers; their domain is specified by equation (3.1).

$$x_k \in I_k \text{, being } I_k \subseteq \mathbb{R}, \forall k = 1,..., p \tag{3.1}$$

In general, input parameters may assume any value in their domain, so that it must be assumed that they vary continuously.
Therefore, the input domain, *D*, is the Cartesian product of the domains of single inputs, as expressed in equation (3.2).

$$D \subseteq \mathbb{R}^p,\ D = \prod_{k=1}^{p} I_k \tag{3.2}$$

As regards output categories, two approaches could be followed:

- Define one discrete variable, y, such that $y \in \{1, 2,\ldots,N_{out}\}$.
- Define y as a vector of N_{out} variables, (y_1, y_2,…, y_{Nout}), where $y_k \in \{0, 1\}$, $\forall$ k = 1,…, N_{out}

These two definitions are equivalent, but the second one allows more flexibility, therefore it will be adopted here. Thus, the output space, Y_{out}, results defined.

Given these premises, the identifier can be seen as a function, f_{ID}, which associates the vector of input parameters to that of output categories.

$$\begin{aligned} &f_{ID} : D \rightarrow Y_{out} \\ &y = f_{ID}(x) \end{aligned} \tag{3.3}$$

Fundamentals of fuzzy logic

Traditionally, any approach to logic is based on two Aristotelian assumptions:

- principle of identity and no contradiction;
- principle of third excluded (*tertium non datur*).

The former principle affirms that if a statement is true it cannot be false at the same time, the latter affirms that if a statement is not true, then it is necessarily false.

Any concept can be thought as the membership of a certain object to a proper set; for instance, an object is cold if it belongs to the set of cold objects. Hence, the above mentioned principles can be reformulated in terms of set theory. Let S be the universe-set, A a generic set, representing a certain concept, and $\neg A$ the complement of A ($\neg A$ stands for *not A*)

The principle of identity and no contradiction can be formulated as in equation (3.4).

$$A \cap \neg A = \varnothing \tag{3.4}$$

The principle of third excluded can be formulated as in equation (3.5).

$$A \cup \neg A = S \text{ i.e. } \neg A = S - A \tag{3.5}$$

These principles are adequate for crisp concepts; on the contrary, concepts which are intrinsically fuzzy (e.g. “low” and “high”, “cold” and “hot”, “tall” and “short”) require these principles to be extended. In general, any logic approach which does not hold or extends either one of those principles is said “fuzzy approach”.

In the following, a theoretical approach to fuzzy logic is presented, which is simplified with respect to traditional approaches (see, e.g., [Zimmermann]) by means of appropriate choices and restrictions. These simplifications are introduced in order to reproduce optimally human reasoning. Despite the simplifications, this approach results sufficiently general for the tasks discussed in this work. In fact it is fully compatible with the AI techniques that will be described later and allows great flexibility. Therefore, such an approach matches all the requirements which were anticipated earlier about machine teaching / learning techniques.

The principle of identity and no contradiction states that an object either belongs to a set or it does not. Therefore, considering the membership of an object, x, to a set, X, as a function, the same principle can be reformulated by affirming that the codomain of a membership function is a set containing two elements only: 0 (false) and 1 (true). In this light, the principle of identity and no contradiction is expressed by equation (3.6).

$$\in_X : x \to \{0,1\} \tag{3.6}$$

The fuzzy extension of the principle of identity and no contradiction is obtained assuming that the codomain of a membership function is the interval [0, 1], that is, all real numbers between 0 and 1, 0 and 1 included. This extension is expressed by equation (3.7),

$$\mu_X : x \to \{[0,1]\} \ , \{[0,1]\} \subset \square \tag{3.7}$$

where symbol "μ" is used instead of "∈" to indicate the adoption of fuzzy logic.
This assumption is equivalent to admit degrees of truth and states that an object can partially belong to a set and, maybe, partially belong to another set, at the same time. Hence, with this assumption neither one of the two Aristotelian principles is valid anymore.

Observation
Because the output of the identifier was defined as a vector with variables belonging to {0, 1}, the adoption of this fuzzy approach determines that the same variables range in [0, 1]. In fact, equation (3.7) must be valid in general. Thus, it results that the identifier associates the input vector to each output category with a membership that ranges in [0, 1].

The basic working principle of any fuzzy engine is to generate a fuzzy partition (mapping) of the input space and successively associate portions of such a space to output categories. In order to achieve a mapping of the input space, it is necessary that input parameters have discrete values. Therefore, if input parameters are characterized by continuous values, they must be discretized, that is, each parameter domain shall be divided in a limited number of sub-domains. When using fuzzy logic, the discretization process is said "fuzzyfication" and sub-domains are fuzzy sets. Successively, the association of the fuzzy input space to output categories takes place through operators based on "if - then" relations, called "fuzzy rules".
Hence, a fuzzy approach to identification consists in establishing fuzzy sets (for instance: "low", "medium", "high") for each input quantity, creating a mapping of the input environment. Then, outputs are derived by linguistic rules such as "If parameter x is low and parameter y is medium or high, then output #2". Therefore, a fuzzy identifier needs, besides the input vector, a set of references in order to fuzzyfy input quantities and a set of rules in order to associate fuzzyfied inputs to output categories. Such a structure is schematically represented by figure 3.1.

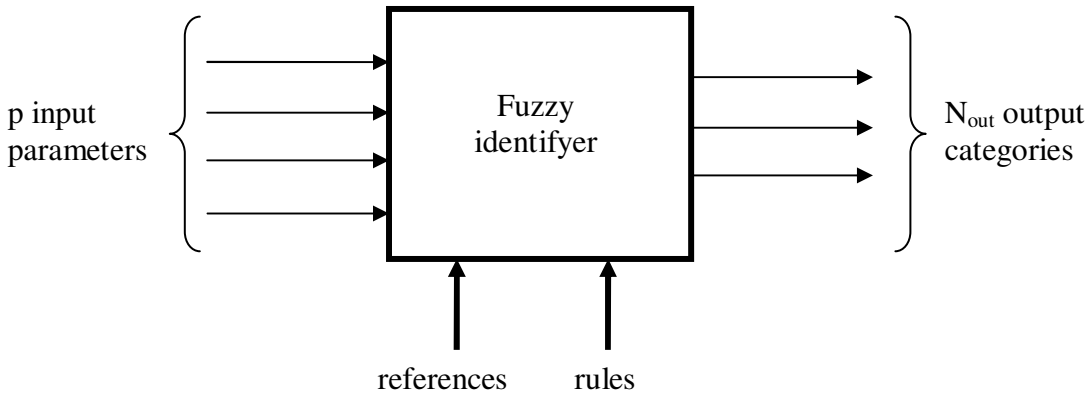


Fig. 3.1: *schematic representation of a fuzzy identifier.*

Thus, the first step to build a fuzzy identifier is fuzzyfication, that is, the attribution of a membership function to each sub-set, for each parameter.

Figure 3.2 shows an example of fuzzyfication, supposing that the j-th input parameter, x_j, is fuzzyfied with three trapezoidal fuzzy-sets.

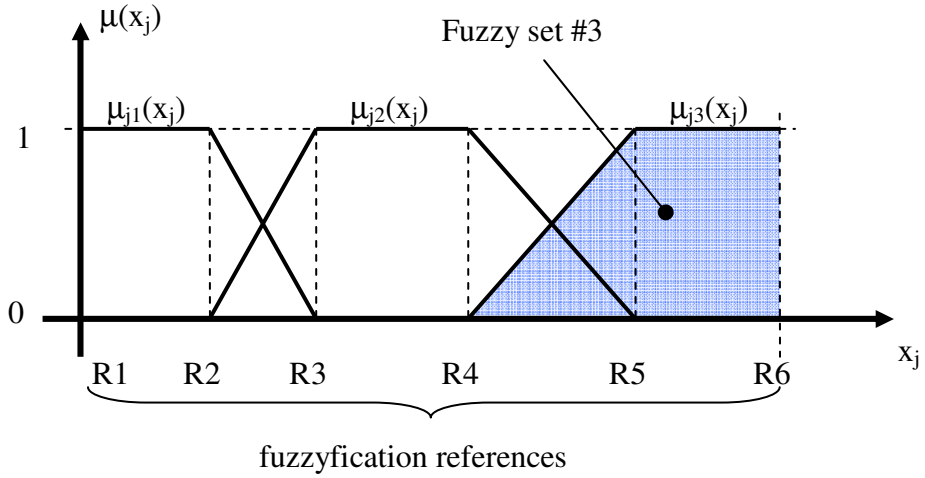


Fig. 3.2: *example of fuzzyfication of parameter x_j in three fuzzy sets.*

Note that the notation μ_{ji} is used, for simplicity, instead of the equivalent (and more precise) notation μ_{Zji}, where Z_{ji} indicates the i-th fuzzy set of the j-th parameter.

In the following, an approach to fuzzy logic will be discussed which is peculiar of this thesis work.

Notation

Let p be the number of input parameters.

Let **f** = $(f_1,\ldots f_p)$ be a vector containing the number of fuzzy-sets used to fuzzyfy the p input parameters; e.g. f_k is the number of fuzzy sets for the k-th parameter.

Definition of the membership function of a fuzzy set.

$$\textit{Membership function, } \mu_{ji}\left(x_j\right): I_j \rightarrow [0,1] \tag{3.8}$$

Provides the j-th parameter with membership degree to the i-th fuzzy-set. Its output is a real number ranging in [0, 1]; j ranges in {1,…, p}, i ranges in {1,…, f_j}.

→ Requirements for membership functions.

- Each fuzzy membership μ_{jk} must be uniformly continuous in its domain I_j.
- Each fuzzy membership must be equal to 1 at least once, as specified in equation (3.9).

$$\forall j = 1,..., p \quad \forall i = 1,..., f_j \quad \exists x_j \in I_j : \mu_{ji}\left(x_j\right) = 1 \tag{3.9}$$

- For each fuzzy set, it must be possible to find at least one value of the input parameter, such that the membership function is not-increasing-monotone for greater values and not-decreasing-monotone for smaller values. Such a requirement is formalized in equation (3.10).

$$\begin{gathered} \forall j = 1,..., p \quad \forall i = 1,..., f_j \quad \exists \overline{x}_j \in I_j : \\ \frac{d\mu_{ji}}{dx_j}\left(x_j\right) \geq 0 \ \forall x_j < \overline{x}_j \ \text{ AND } \ \frac{d\mu_{ji}}{dx_j}\left(x_j\right) \leq 0 \ \forall x_j > \overline{x}_j \end{gathered} \tag{3.10}$$

Thus, the discretization of the input parameters produces a separation of the input domain in sub-domains. This process is schematized in figure 3.3.

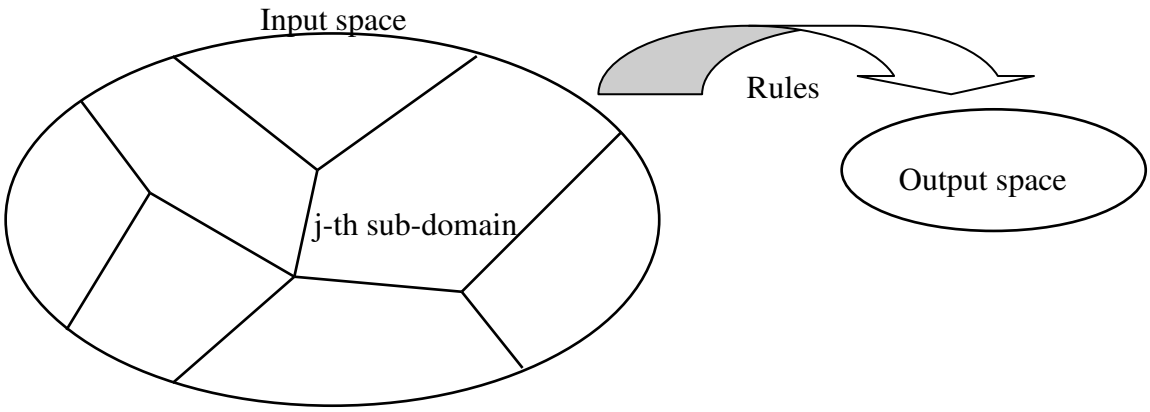


Fig. 3.3: *schematization of the relationship between mapped input space and output space.*

The effect of discretization is that the output vector, *y*, is specified for each sub-domain, by means of rules. If sub-domains are fuzzy sets, they may overlap since their boundaries are fuzzy. In this light, the adoption of fuzzy logic has two main advantages. The first advantage consists of a reduction of the impact of approximation intrinsic to discretization, that is, it avoids an abrupt shift from one domain to another one. The second advantage, as will be shown later, consists of the possibility to express rules in a linguistic form, which optimally fits human reasoning.

Thus, being F_j the sub-domains resulting from discretization, fuzzy rules define a function, f_d, which has as domain the mapped input space and as codomain the output space. With reference to equation (3.3), equation (3.11) defines function f_d.

$$\begin{aligned} & f_d : S \to Y_{out} \\ & y = f_d(F) \end{aligned} \tag{3.11}$$

At this point, a fundamental distinction has to be made. Two different kinds of fuzzy sets are considered. The first kind consists of those fuzzy sets defined in equation (3.7), relevant to a specific parameter (dimension), which perform the mapping of the input space through the fuzzyfication process. The other kind of sets is constituted by portions of the input space, which, in general, have p dimensions. In particular, the p-dimensional sets which constitute the mapping of the input space (sub-domains) are fuzzy set. In fact, each one of them is the fuzzy equivalent of the Cartesian product of N_{out} intervals that would result from a non fuzzy discretization.

To proceed in the description of this logic approach, it is essential to define logic operators, union and intersection, for fuzzy sets pertinent to a specific dimension.
Given two sets pertinent to the same (e.g. the j-th) parameter, X_{jh} and X_{jk}, union and intersection operators are defined by equations (3.12) and (3.13), respectively.

$$X_{jh} \cup X_{jk} = \bar{X}_j \subseteq I_j : \mu_{\bar{X}_j} = \mu_{jh} + \mu_{jk} \tag{3.12}$$

$$X_{jh} \cap X_{jk} = \bar{X}_j \subseteq I_j : \mu_{\bar{X}_j}(x_j) = 0,\ \forall x_j \in I_j \text{ , i.e. } \bar{X}_j = \varnothing \tag{3.13}$$

It has to be observed that the definition of intersection between fuzzy sets provided by (3.13) may appear in contrast with the meaning of fuzzy logic itself. Such a definition implies that it is not allowed to associate the intersection of fuzzy sets to the output space. Indeed, in the opinion of the author, fuzzy sets are to be intended as elementary portions of a discretized space, therefore anything that stands between them is not part of the input domain. If a higher resolution is required in the discretization (mapping) of the input space, e.g. the situation where a parameter is Low and Medium at the same time must be associated directly to the output space (being object of a rule condition), a new, intermediate fuzzy set shall be defined, e.g. Low-Medium set. In this way, the

approach to fuzzy logic results more robust, more suited to human reasoning and more flexible, in the prospective to integrate artificial intelligence techniques (as will be proved in the following paragraphs).

Given these definitions, a further constraint shall be imposed over fuzzy sets pertinent to the same dimension, as expressed by equation (3.14).

$$\sum_{i=1}^{f_j} \mu_{ji}\left(x_j\right) = 1,\ \forall x_j \in I_j,\ \text{j=1,....p} \tag{3.14}$$

This condition, that will be addressed here as "fuzzy set complementarity", is common in many fuzzy approaches [Zimmermann, Babuška] and often is called "partition of unity".

It is worth to analyze the condition of fuzzy set complementarity more deeply. For the j-th parameter, fuzzy set complementarity means that, for any input x, the sum of the memberships equals one. Therefore, x belongs to the k-th fuzzy set as much as it does not belong to the other fuzzy sets.
In this light, the complementary set to the k-th fuzzy set can be defined as the fuzzy set which has a membership function complementary to μ_j. This definition can be formalized as in equation (3.15).

$$\neg X_{jk} = \bar{X}_j \subseteq I_j : \mu_{Y_j} = 1 - \mu_{X_{jk}} \tag{3.15}$$

Because of fuzzy set complementarity condition (3.14), equation (3.15) becomes

$$\neg X_{jk} = \bar{X}_j \subseteq I_j : \mu_{Y_j} = \sum_{i \neq k} \mu_{ji} \tag{3.16}$$

Because of the definition of union between fuzzy sets (3.12), the complement to a fuzzy set can be expressed as

$$\neg X_{jk} = \bigcup_{i \neq k} X_{ji} = I_j - X_{jk} \tag{3.17}$$

By the comparison of equation (3.17) and equation (3.5), it can be argued that the principle of third excluded (*tertium non datur*) is still valid, according to choices and definitions adopted.

The membership of a generic input vector, ***x***, to a specific (p-dimensional) sub-domain, *F*, has to be defined. Let **k** = (k_1,…, k_p) be a vector of indexes, such that the j-th value of k represents the index of the mono-dimensional fuzzy set to which *F* belongs in the j-th dimension. In other words, **k** is the vector of coordinates of *F* in the input p-dimensional space. The membership of **x** to *F* is given

by equation (3.18).

$$\mu_{F_k}(x) = \prod_{j=1}^{p} \mu_{jk(j)}\left(x(j)\right) \tag{3.18}$$

To proceed in the description of this logic approach, it is essential to define logic operators, union and intersection, for two generic sub-sets of the input space.
According to equations (3.12) and (3.13), and given (3.17), union and intersection operators can be derived also for sub-domains through equations (3.19) and (3.20), which represent the p-dimensional extension of (3.12) and (3.13).

$$F_k \cup F_h = \bar{F} \subseteq S : \mu_{\bar{F}} = \mu_{F_k} + \mu_{F_h} \tag{3.19}$$

$$F_k \cap F_h = \bar{F} \subseteq S : \mu_{\bar{F}} = 0, \text{ i.e. } \bar{F} = \varnothing \tag{3.20}$$

Thus, sub-domains constitute elementary portions of the input space, in fact they cannot be further divided, because of discretization process. In addition, sub-domains constitute a partition of the input space, in fact they are mutually disjoint and their union is the entire space, because of the fuzzy set complementarity condition.
Therefore, any sub-set of the input space can be considered as the union of a certain number of (elementary) sub-domains. At this point, union and interception of two generic sub-sets of the input space can be defined. In fact, given two generic sub-sets of the input space, *A* and *B*, union and interception operators are provided by equations (3.21) and (3.22), respectively.

$$A \cup B = \bigcup_k F_k,\ F_k \subseteq A \ \vee\ F_k \subseteq B \tag{3.21}$$

$$A \cap B = \bigcup_k F_k,\ F_k \subseteq A \ \wedge\ F_k \subseteq B \tag{3.22}$$

In (3.21) and (3.22), symbols "∨" and "∧" have the meaning of "inclusive or" and "and", respectively, as in traditional logic.

It is important to observe that equation (3.18) can be extended (equation (3.23)) in such a way that dimension index *j* runs in a vector, **d**, which corresponds to a subset of the vector (1,…,p).

$$\mu_{E_{d,k}}(x) = \prod_{s=1}^{z} \mu_{d(s)\,k(d(s))}\left(x\left(d(s)\right)\right) \tag{3.23}$$

In equation (3.23) z is the number of elements in $\boldsymbol{d}$ (length of $\boldsymbol{d}$).
Let $\boldsymbol{d}$ be a vector with elements belonging the set {1,…, h-1,…,h+1,…,p}. In this case, equation (3.23) consists of the product of p-1 factors, because the h-th dimension is not included in $\boldsymbol{d}$. The same result could have been obtained with equation (3.18) by imposing $\mu_{hk(h)}=1$. Hence, $E_{d,k}$ in equation (3.23) is the union of all F_k (sub-domains) which have in common all dimension indexes except for the h-th.

Equations (3.18) and (3.23) can be further generalized. Let **C** be a cluster of p vectors. The elements of the j-th element of **C** constitute a sub-set of {1,…, f_j}, where f_j is the number of fuzzy sets of the j-th parameter. Given these premises, equation (3.24) can be formulated.

$$\mu_{E_C}(x) = \prod_{j=1}^{p}\left(\sum_{k\in C_j}\mu_{jk}(x)\right) \qquad \text{with } C=\{C_j\}, j=1,...,p\ ,\ C_j\subseteq\{1,...,f_j\} \tag{3.24}$$

Equation (3.24) calculates the membership of a generic input vector to E, which is a subset of the mapped input space. According to equation (3.24), E consists of the union of all sub-domains $F_{\boldsymbol{k}}$ (where $\boldsymbol{k}$ is a vector as specified in (3.18)) such that $k_j \in C_j$. The number of index vectors k which match condition $k_j \in C_j$ is given by the product of p terms, where the j-th term is the number of elements of C_j. If K is the set containing these vectors k, E can be expressed by equation (3.25).

$$E=\bigcup_{k\in K}F_k \tag{3.25}$$

Note that the result of equation (3.23) with a given $\boldsymbol{k}$ and $\boldsymbol{d}$ (1,…, h-1,…,h+1,…,p) can be obtained with equation (3.24) with C such that $C_j = (k_j)\ \forall j\neq h$ and $C_h = (1,\ldots,f_h)$.

Since the output vector is defined on portions of the (p-dimensional) space, in particular on sub-domains, as effect of discretization, it is important to specify how f_{ID} shall be evaluated on a generic input vector given a specific portion of the input space.
Let A be a generic subset of S (the mapped input space) and let x be a generic input vector to be evaluated (see equation (3.3)). The identification of x with respect to (assuming) A is provided by equation (3.26),

$$f_{ID}(x\mid A) = f_d(A)\cdot\mu_A(x) \tag{3.26}$$

where f_d is the equivalent of f_{ID}, with a discretized domain.

The definition of "identification of *x* assuming *A*" provided by equation (3.26) can be interpreted as the identification that would result for *x* in case *A* was the only portion of the mapped input space to be associated to the output space, that is, in case *A* was the (definition) domain of f_d.

From equations 21 – 23, two further relations can be derived, that is equations (3.27) and (3.28), regarding the identification of an input vector given a set which is the interception or the union of other two sets, respectively.

$$f_{ID}(x \mid (A \cap B)) = f_d(A \cap B) \cdot \mu_{(A \cap B)}(x) \tag{3.27}$$

$$f_{ID}(x \mid (A \cup B)) = f_d(A) \cdot \mu_A(x) + f_d(B) \cdot \mu_B(x) - f_d(A \cap B) \cdot \mu_{(A \cap B)}(x) \tag{3.28}$$

Total membership theorem

Given p input parameters, let *S* be the mapped input space that results from the fuzzyfication of the inputs, performed according to equations (3.8) – (3.12). Assuming that the output space is associated to elementary portions of *S*, (sub-domains F_j) through a set of fuzzy rules (function f_d), any input vector x can be associated to the output space by evaluating the membership of x to the sub-domains and the value of function f_d in the same sub-domains, according to equation (3.29).

$$f_{ID}(x) = \sum_k f_d(F_k) \cdot \mu_{F_k}(x) \tag{3.29}$$

In (3.29) the sum is extended to all the sub-domains.

Proof

$$f_{ID}(x) = f_{ID}(x \mid S) = f_{ID}\left(x \mid \bigcup_k F_k\right) \tag{3.30}$$

Because the sub-domains constitute a partition of S, results

$$F_h \cap F_k = \varnothing \ \forall (h,k) : h \neq k, \text{ and } \bigcup_k F_k = S \tag{3.31}$$

Therefore, introducing (3.26) in (3.30), according to (3.27) and (3.28), results (3.29).

Observation

Equation (3.29) has operative validity. In fact, it provides an expression for (3.2) which is evaluable by the machine. In fact, when the identifier is run, it must evaluate function f_{ID} for a given input vector, x. This vector corresponds to a certain subset, G, of the mapped input space. G consists of the union of all elementary domains, each one weighted by the membership of x to that sub-domain, and could be visualized as in figure 4.4.

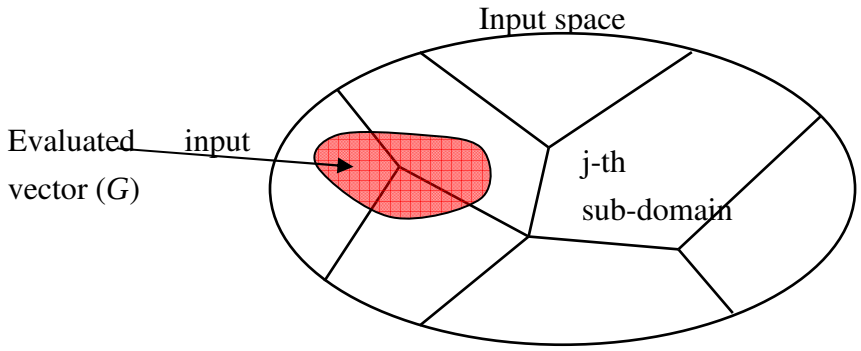


Fig. 3.4: *representation of an evaluated dataset as a sub-set the input space.*

Adjacent superposition theorem

For a given input parameter (e.g. the j-th parameter), membership superposition is limited to adjacent fuzzy sets.

Proof

Let x_B be the maximum value of the j-th parameter for which $\mu_{jk}(x_j) > 0$. Supposing that μ_{jk+2} exists, let x_A be the minimum value of the j-th parameter for which $\mu_{jk+2}(x_j) > 0$.
If, for absurd, the k-th and the k+2-th fuzzy sets overlapped, it would result that $x_A < x_B$, therefore it would result that
$\mu_{jk}(x) + \mu_{jk+2}(x) > 0,\ \forall x \in [x_1, x_3]$, being $x_1, x_3 \in I_j$: $\mu_{jk}(x_1) = \mu_{jk+2}(x_3) = 1$
Let x_2 be a value of the j-th parameter such that $x_2 \in I_j : \mu_{jk+1}(x_2) = 1$

For (9), (10) and (14), results that $x_1 < x_2 < x_3$
Hence, $\mu_{jk}(x_2) + \mu_{jk+2}(x_2) > 0,$ since $x_2 \in [x_1, x_3]$
Therefore, it would result that $\mu_{jk}(x_2) + \mu_{jk+2}(x_2) + \mu_{jk+2}(x_2) > 1$

But this result is absurd, because it is against the complementarity condition expressed in (3.14).
An example of such an absurd fuzzy set geometry is shown by figure 3.5.

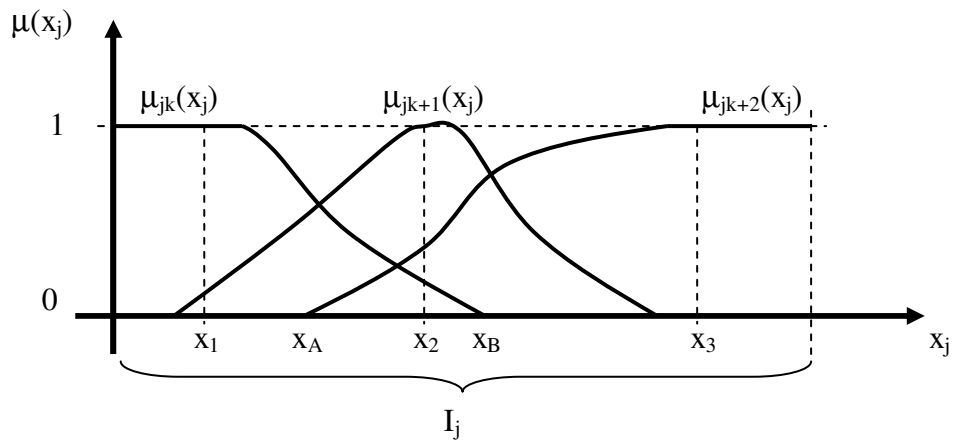


Fig. 3.5: *example of absurd fuzzy set geometry.*

In the following, further definitions will be given concerning the fuzzyfication process. In particular, concepts of local and global fuzzyness degree will be introduced.

Supposing to have the j-th parameter fuzzyfied in 3 fuzzy sets ($f_j = 3$), with membership function as shown in figure 3.6.

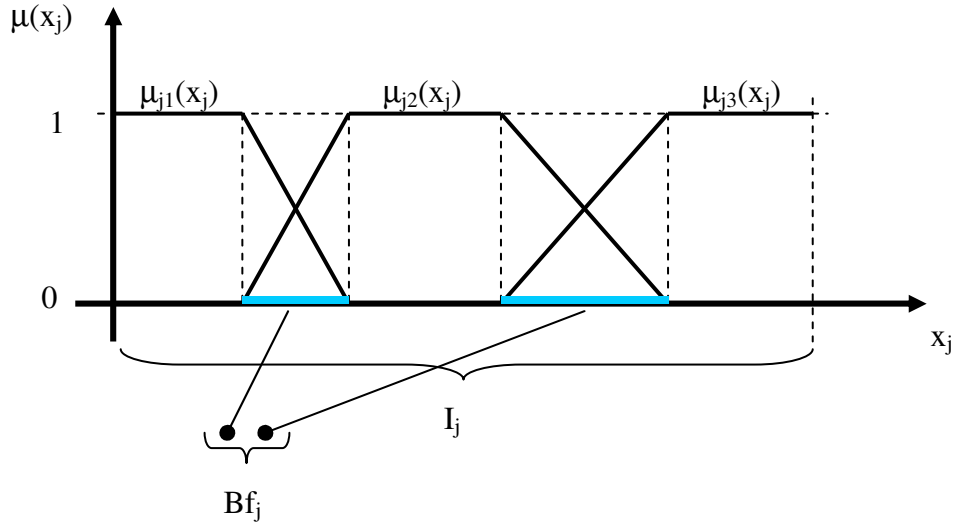


Fig. 3.6: *representation of "local fuzzyness degree".*

In Fig. 3.6,

- I_j : domain of the j-th parameter (local domain);
- μ_{ji} : membership of the j-th parameter to the i-th fuzzy set (i = 1,2,3 in this example).
- Bf_j : subset of the local domain, containing values of the input parameter x_j such that $\mu_i(x_j) \in]0, 1[, \forall$ i (i = 1,2,3 in Fig. 3.3).

Local fuzzyness degree (Gf_j) is given by equation (3.32).

$$Gf_j = \frac{Bf_j}{I_j} \tag{3.32}$$

Global fuzzyness degree (Gf) is given by equation (3.33).

$$Gf = 1 - \frac{\prod_{j=1}^{p} \left(I_j - Bf_j \right)}{\prod_{j=1}^{p} I_j} \tag{3.33}$$

Gf_j and Gf range in [0, 1]. Gf_j is pertinent to a single parameter, that is, to one dimension of input space; Gf is the extension of Gf_j to p parameters (p-dimensional space).

Focusing on a single dimension ($Gf_j \equiv Gf$), and with condition of fuzzy set complementarity.
Fig 3.7 shows an example of fuzzyfication with Gj = 1; figure 3.8 shows an example of fuzzyfication with Gj = 0.

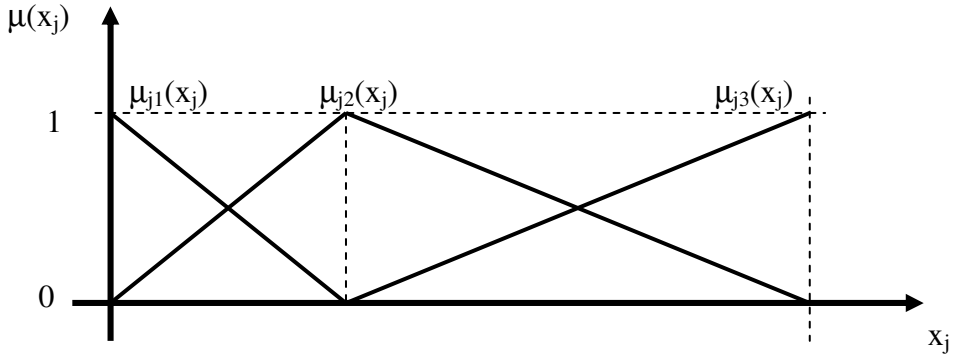


Fig. 3.7: *fuzzyfication of parameter x_j in three fuzzy sets with Gf = 1.*

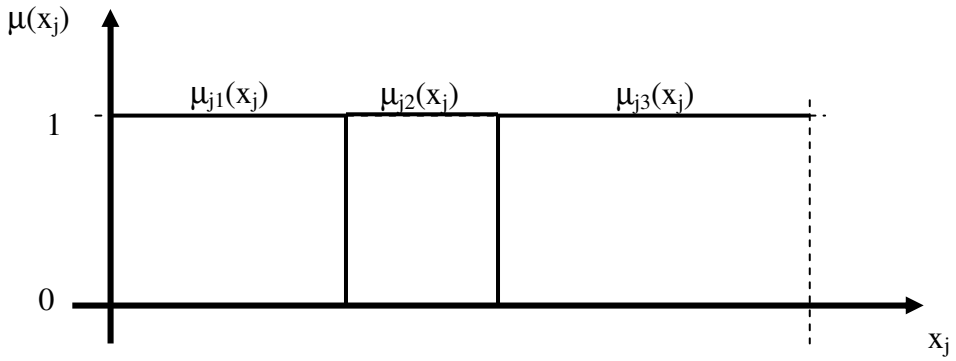


Fig. 3.8: *fuzzyfication of parameter x_j in three fuzzy sets with Gf = 0.*

Fig. 3.8 represents the division of parameter x_j in three crisp intervals; in fact, each membership

assumes only values 0 or 1.

Observation

The proposed approach is valid for a generic identifier operating on a multidimensional discretized space. The discretization is supposed fuzzy (general case) but, when the degrees of fuzzyness for each parameter assumes value 0, the same theory actually describes a crisp identifier (particular case).

At this point, it is essential to give a description of fuzzy rules, which allow to define function f_d, that is, to address portions of the mapped input space to the output space, through linguistically interpretable statements.
An example of fuzzy rule statement is presented in the following:

If { (parameter 1 is Low) AND [(parameter 2 is Medium or High) OR (parameter 3 is Low)] }
then Output #3

The following definitions regard fuzzy rules.

Condition: is the logical expression enclosed in round parenthesis. It regards a single input parameter (a single dimension of the input space).

Rule antecedent: is the logical expression enclosed in graph parenthesis. It regards all p input parameters, that is, it consists of p conditions somehow related to each other. It specifies a portion of the mapped input space.

Condition operator: or, (and[7]): operator that describes the relationship among the fuzzy-sets that may be present in the same condition. Therefore, such an operator is pertinent to a single input parameter. These operators are defined by equations (3.12) and (3.13), respectively.

Rule operators: AND, OR: operators that describes the relationship among the conditions present in a specific rule. Therefore, such operators are pertinent to a single rule. These operators are defined by equations (3.21) and (3.22), respectively. In case the expression is evaluated directly, i.e. without the use of total membership theorem, equations (3.27) and (3.28) shall be also used for the two

[7] The use of operator "and" inside a condition is possible but, since its definition (3.13), it has no practical application.

operators, respectively.

Elementary condition: a condition that involves only one fuzzy set.

Elementary rule: a rule obtained by relating p elementary conditions with AND operators, that is, a rule such that the portion of the mapped input space to which it refers coincides with a sub-domain.

Size of the input-fuzzy-space (D): is the total number of (elementary) sub-domains. It is given by expression (3.34):

$$D = \prod_{j=1}^{p} f_j \tag{3.34}$$

Let z_j be the maximum number of overlapping fuzzy-sets for the j-th parameter. Fuzzy sets overlap when their memberships are grater than zero at the same time, for a certain value of the input quantity.

Maximum number of overlapping status (Z): maximum number of sub-domains that can be activated at the same time. It is given by expression (3.35):

$$Z = \prod_{j=1}^{p} z_j \tag{3.35}$$

For adjacent superposition theorem, (3.35) can be specialized in (3.36).

$$Z = 2^p \tag{3.36}$$

Example

The approach to fuzzy logic described here is meant to reproduce human reasoning. Therefore, the following example (application) describes a simple identification task handled by human mind.
Imagine that a person witnessed a murder, but did not have the chance to see clearly. Asked by the police officers to provide a description, the witness says: « I saw the murderer was tall and thin ». Supposing that the police has a database containing various information about prejudiced, among which height and weight, the task is to construct an identifier to automatically select those records of the database which fit the description provided by the witness. The goal is to apply as much as possible human-like reasoning, in order to get the identifier to perform as the witness would if he

valuated by sight all the persons in the database (e.g. « That person could be the murderer; that other person could be the murderer too, but with a lower likelihood »).

Height is certainly to be chosen as input parameter. This parameter has a domain (e.g. [100, 300] in centimeters) which is dense. In other words, given any two persons of different height, it is possible that a third person exists, whose height is intermediate with respect to the others.
The fact that the witness remembers that the murderer is tall means that his mind performed a discretization of the parameter height, e.g. he divided the set of all persons in three sets: short, average and tall.

It is noteworthy to observe that a mathematical discretization of parameter height domain, according to the traditional (Aristotelian) approach, would require to select two reference height values (e.g. 160 cm and 180 cm), so that the three sub-sets would be:
[100, 160[→ "short people"
[160, 180[→ "average height people"
[180, 300] → "tall people"
In this case, a person of 180.1 cm would belong to the set of "tall people", while a person of 179.9 cm would belong to the set of "not tall people". On the contrary, it is self evident that human mind would consider these two persons of the same height.
Hence, it can be speculated that human mind does perform discretization (when needed, as in this identification task) of the input domain in sub-domains, but the way it discretizes is fuzzy, as long as the original domain is dense.
Moreover, it can be observed that sub-sets resulting from this discretization are perceived by human mind as a partition. In fact, a person height is either high or medium or low, there is no other possibility; in addition, a person is tall, as much as he is not short or medium. Hence, the fuzzy set complementarity condition seems to be essential to fit human reasoning; in other words, third excluded principle seems to be valid despite fuzziness. Furthermore, if human mind divides persons in three sets with respect to height attribute, including an average size, it is absurd that a person is perceived tall and short at the same time. Therefore, also adjacent superposition theorem seems a requirement to fit properly human reasoning.

Following the described fuzzy approach, the parameter height, h, may be fuzzyfied as reported in figure 3.9.

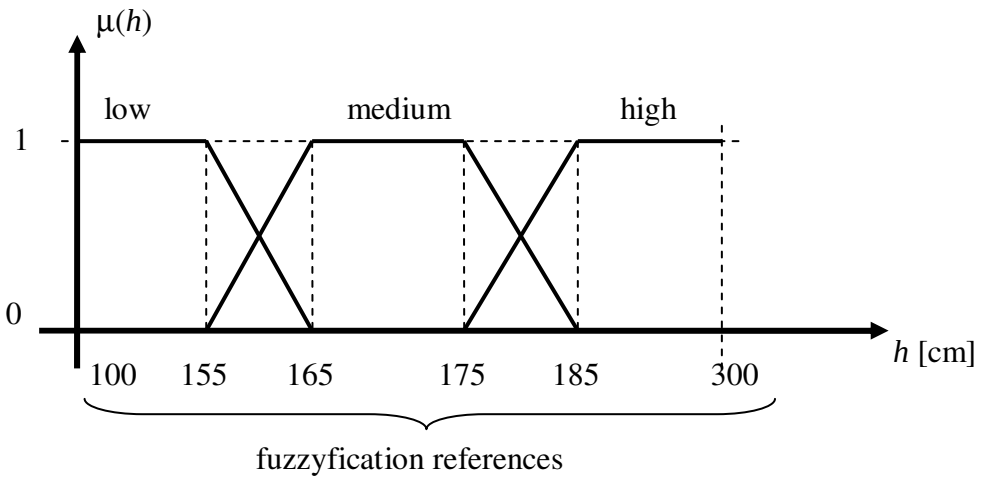


Fig. 3.9: *fuzzyfication of parameter h, (human) height.*

Clearly, the fuzzy references should be chosen according to the concept of "short" and "tall" of the witness.

As regards the second information available, i.e. that the murderer is thin, the weight by itself does not seem to be a significant parameter to get the machine to select thin persons from the database. In fact, "thin" is a person whose weight is small with respect to his height. Therefore, a new quantity shall be derived. One possibility is to use *BMI* (Body Mass Index), which can be calculated known height and weight, *w*, through equation (3.37),

$$BMI = \frac{w}{h^2} \tag{3.37}$$

where *w* is expressed in kg and *h* in m.
BMI could be fuzzyfied as reported in figure 3.10 (according to some nutritional table).

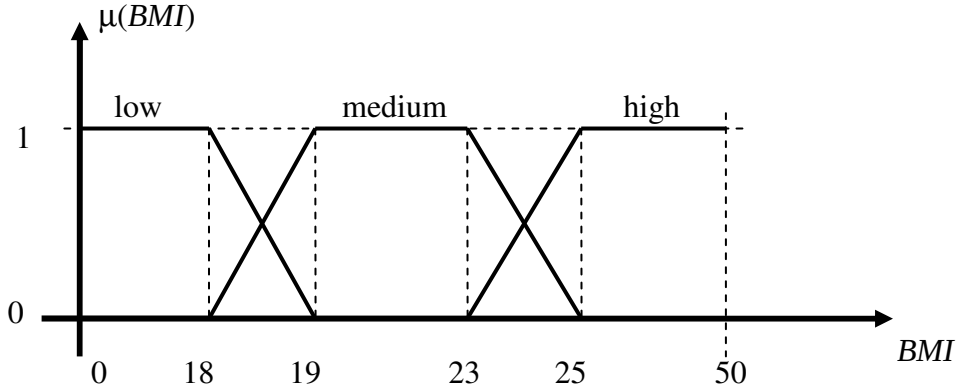


Fig. 3.10: *fuzzyfication of parameter BMI, body mass index.*

Clearly, low values of *BMI* correspond to underweight, medium values to normal weight and high values to overweight.

Thus, a two dimensional input space is generated, by selecting *BMI* and *h* as input parameters. Then, the fuzzyfication process provides a mapping of the input space, as shown by figure 3.11.

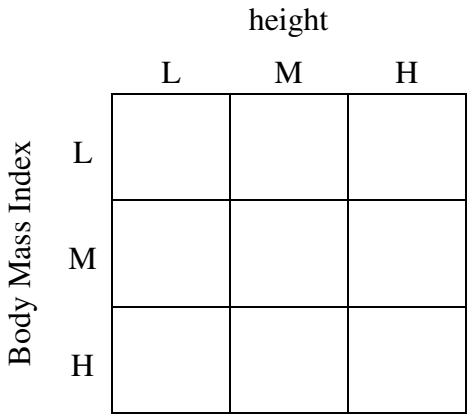


Fig. 3.11: *representation of the mapped input space.*

As regards the output categories, in this example only two categories are of interest: "suspect" and "not suspect". Hence, the output vector, y, has two variables: y_1 and y_2.
Let y_1 be associated to category "suspect".
Let y_2 be associated to category "not suspect".

Actually, the information provided by the witness constitute the knowledge for the identifier, in terms of values of function f_d specified for a portion A of the input space, that is, values of function f_{ID} specified assuming A.
The fuzzy rule set consists of only one rule:

If { (*BMI* is L) AND (*h* is H) } ==> suspect

According to the proposed approach, the transfer of the knowledge provided by this rule to the identifier consists of the following assignment,

$$f_d(A) \leftarrow [y_{1A}, y_{2A}] \tag{3.38}$$

where A is the portion of the mapped input space specified by the antecedent of the rule, and y_{1A} and y_{2A} are 1 and 0, respectively (suspect, not suspect).
In the absence of other rules, one can assume that any portion of the mapped input space which is not A is associated to "not suspect" category. This knowledge can be transferred to the identifier through assignment (3.39),

$$f_d\,(S-A) \leftarrow [y_{1\neg A}, y_{2\neg A}] \tag{3.39}$$

where $S\text{-}A \equiv \neg A$, and $y_{1\neg A}$ and $y_{2\neg A}$ are 0 and 1, respectively.
These assignments can be represented graphically as shown in figure 3.12.

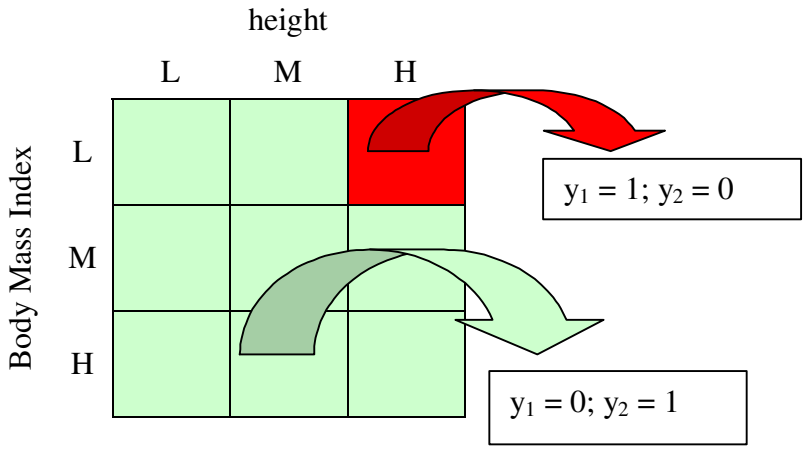


Fig. 12: *graphical representation of the rule set.*

Table 3.1 contains an example of the "police" database, focusing on height and weight and showing values calculated for *BMI*. It noteworthy to observe that this database provides input data (i.e. each record provides an input vector), contrarily to other databases, which provide the basis of knowledge for the identifier.

Tab. 3.1: *database containing input vectors.*

Name	Heigth [cm]	Weight [kg]	*BMI*
Prejudiced #1	173	76	25.4
Prejudiced #2	182	59	17.8
Prejudiced #3	194	70	18.6

Hence, the first record of this database provides the input vector x = (173, 25.4), where x_1 correspond to h and x_2 corresponds to *BMI*.
Tables 3.2, 3.3 and 3.4 show the result of the fuzzyfication process for the input vectors relevant to record 1, 2 and 3, respectively. Therefore, values of μ_{1k} and μ_{2k} are calculated, for k = 1 (fuzzy set "low"), 2 (fuzzy set "medium") and 3 (fuzzy set "high").

Tab. 3.2: *fuzzyfication result for the first record (of database of table 1).*

Prejudiced #1	crisp	fuzzy		
		L	M	H
h	173	0	1	0
BMI	25.4	0	0	1

Tab. 3.3: *fuzzyfication result for the second record (of database of table 1).*

Prejudiced #2	crisp	fuzzy		
		L	M	H
h	182	0	0.3	0.7
BMI	17.8	1	0	0

Tab. 3.4: *fuzzyfication result for the third record (of database of table 1).*

Prejudiced #3	crisp	fuzzy		
		L	M	H
h	194	0	0	1
BMI	18.6	0.4	0.6	0

To evaluate the membership of a given input vector, x, to a specific sub-domain of the mapped input space, F_{ij}, equation (3.18) shall be used.

For example, considering sub-domain $F_{3\text{-}1}$ (first parameter, h, is L and second parameter, BMI, is H), which is related to "suspect" category in the output space, equations (3.40), (3.41) and (3.42) calculate the membership to $F_{3\text{-}1}$ of the first, second and third input vector, respectively.

$$x = (173,\ 25.4) \quad \mu_{F_{3-1}}(\mathrm{x}) = \mu_{1\text{-}3}(x_1) \cdot \mu_{2\text{-}1}(x_2) = 0 \cdot 0 = 0 \tag{3.40}$$

$$x = (182,\ 17.8) \quad \mu_{F_{3-1}}(\mathrm{x}) = \mu_{1\text{-}3}(x_1) \cdot \mu_{2\text{-}1}(x_2) = 0.7 \cdot 1 = 0.7 \tag{3.41}$$

$$x = (194,\ 18.6) \quad \mu_{F_{3-1}}(\mathrm{x}) = \mu_{1\text{-}3}(x_1) \cdot \mu_{2\text{-}1}(x_2) = 1 \cdot 0.4 = 0.4 \tag{3.42}$$

Since the "suspect" category was assigned (in a crisp way) with membership 1 to sub-domain $F_{3\text{-}1}$ and with membership 0 elsewhere, equations (3.40) – (3.42) provide directly the desired information (e.g. the prejudiced #2 belongs to the "suspect" category with membership 0.7, that is, he may be a suspect with likelihood 70 %).

In a more general case, the total membership theorem could be applied. In this light, figure 3.13 shows in graphical form the membership of each input vector to all the sub-domains.

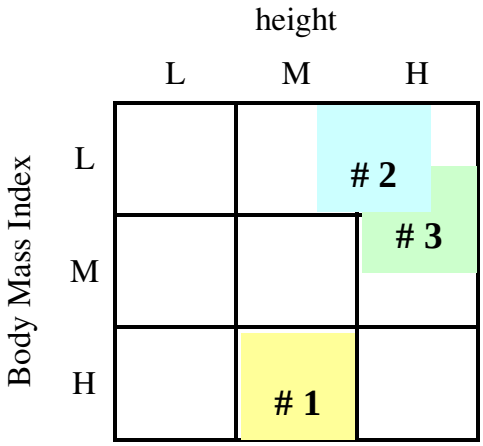


Fig. 3.13: *graphical representation of the membership of records 1-3 to single sub-domains.*

In Fig. 3.13 one can recognize qualitatively that the first record overlaps to sub-domain $F_{2\text{-}3}$ (100 %), the second record overlaps to sub-domains $F_{2\text{-}1}$ (30 %) and $F_{3\text{-}1}$ (70 %) and the third record overlaps to sub-domains $F_{3\text{-}1}$ (40 %) and $F_{3\text{-}2}$ (60 %).

Description of "linguistic" and "matrix" fuzzy engines

This paragraph describes two possible ways to implement an identifier according to the described approach. These two implementation strategies are quite different, although they get the same result, and may be addressed to as "linguistic" fuzzy engine and "matrix" fuzzy engine.

Linguistic fuzzy engine

The implementation strategy of the "linguistic" fuzzy engine consists of the evaluation of expressions which are derived directly from linguistic fuzzy rules. In other words, each linguistic rule is "translated" into a mathematically evaluable expression taking advantage of the correspondences among linguistic operators and mathematical operators. Hence, for a given input (vector), output values are derived for every rule of the rule set; then, a criterion shall be chosen to merge these values into a single, final output.

To get into details, the following rule can be taken as reference, assuming, for example, 3 input parameters, each one fuzzyfied with three sets, and 4 output categories.

If { (parameter 1 is Low) AND [(parameter 2 is Medium or High) OR (parameter 3 is Low)] }
then Output #3

"(parameter 1 is Low)" → represents the condition on the first parameter, x_1. It is calculated as the membership of x_1 to the first fuzzy set (fuzzy set "Low") → $\mu_{1-1}(x_1)$. If the rule consisted of this condition only, its antecedent would be calculated by equation (3.23) with $\boldsymbol{d}$ = (1) and $\boldsymbol{k_1}$ = 1.

"(parameter 2 is Medium or High)" → represents the condition on the second parameter, x_2. Operator "or", which is relevant to a single condition, corresponds to the union of fuzzy sets relevant to the same parameter (dimension). Therefore, according to equation (3.12), this condition can be evaluated as → $\mu_{2-2}(x_2)+\mu_{2-3}(x_2)$. If the rule consisted of this condition only, its antecedent would be calculated as the sum of the results obtained applying equation (3.23) with $\boldsymbol{d}$ = (2) and $\boldsymbol{k_2}$ = 2 and with $\boldsymbol{d}$ = (2) and $\boldsymbol{k_2}$ = 3.

"(parameter 3 is Low)" → represents the condition on the third parameter, x_3. → $\mu_{3-1}(x_3)$. If the rule consisted of this condition only, its antecedent would be calculated by equation (3.23) with $\boldsymbol{d}$ = (3) and $\boldsymbol{k_3}$ = 1.

Operator "AND" between conditions on different parameters corresponds to the intersection of

different sub-sets of the multidimensional input space, which is defined by equation (3.22). Because the rule is valuated directly, equation (3.27) has to be used.

Operator "OR" between conditions on different parameters corresponds to the union of different sub-sets of the multidimensional input space, which is defined by equation (3.21). Because the rule is valuated directly, equation (3.28) has to be used.

Therefore, the result of this rule would be evaluated by a "linguistic" fuzzy engine through equation (3.43).

$$y_3 = \mu_{1\text{-}1}(x_1) \cdot \left[\left(\mu_{2\text{-}2}(x_2) + \mu_{2\text{-}3}(x_2) \right) + \mu_{3\text{-}1}(x_3) - \left(\mu_{2\text{-}2}(x_2) + \mu_{2\text{-}3}(x_2) \right) \cdot \mu_{3\text{-}1}(x_3) \right] \qquad (3.43)$$

Equation (3.43) is the mathematical expression of the rule above.
Clearly, a fuzzy engine rule set shall be provided with at least one rule for each output category. Therefore, each element of the output vector will be determined. In case the rule set contains more than one rule pertinent to the same output category, a criterion must be chosen to merge these rule outputs. A possible criterion consists in taking the maximum of the output values (e.g. $y_3 = \max(y_{3\text{-rule 1}}, \ldots, y_{3\text{-rule n}})$).
In general, the output vector obtained with this procedure is not normalized, that is, the sum of its elements does not equal one. In fact, the sub-sets of the mapped input space specified by the rule antecedents may have a not-empty interception.

Example

The same example described in the previous paragraph will be considered here. In this case there are two input parameters (x_1 = *BMI*, x_2 = *h*) and two output categories (y_1 = *"suspect"*, y_2 = *"not suspect"*). The rule set consists of two rules:
Rule 1) { (*BMI* is L) AND (*h* is H) } ==> suspect
Rule 2) { (*BMI* is M or H) OR (*h* is L or M) } ==> not suspect

Let the input vector be provided by table 3.5.

Tab. 3.5: *input vector to be evaluated.*

Prejudiced #4	crisp	fuzzy		
		L	M	H
h	182	0	0.3	0.7
BMI	18.6	0.4	0.6	0

Applying the "linguistic" fuzzy engine, rules 1 and 2 shall be evaluated as equations (3.44) and (3.45), respectively.

$$y_1 = \mu_{1\text{-}1}(x_1) \cdot \mu_{2\text{-}3}(x_2) = 0.4 \cdot 0.7 = 0.28 \tag{3.44}$$

$$\begin{aligned} y_2 &= \left(\mu_{1\text{-}2}(x_1) + \mu_{1\text{-}3}(x_1)\right) + \left(\mu_{2\text{-}1}(x_2) + \mu_{2\text{-}2}(x_2)\right) + \\ &- \left(\mu_{1\text{-}2}(x_1) + \mu_{1\text{-}3}(x_1)\right) \cdot \left(\mu_{2\text{-}1}(x_2) + \mu_{2\text{-}2}(x_2)\right) = \\ &\left(0.6 + 0\right) + \left(0 + 0.3\right) - \left(0.6 + 0\right) \cdot \left(0 + 0.3\right) = 0.72 \end{aligned} \tag{3.45}$$

It can be noted that, in this case, the output vector is normalized, i.e. $y_1 + y_2 = 1$. This is a consequence of the fact that the rule antecedents correspond to sub-sets which constitute a partition of the mapped input space. In fact, the second rule was derived from the first by equation (3.39).
This fact is shown graphically by figures 3.14 and 3.15, where the sub-sets of the mapped input space corresponding to rule antecedents are evidenced with blue color (Figs. 3.14 and 3.15 refer to rules 1 and 2, respectively), while the location of the input vector is evidenced with red color.

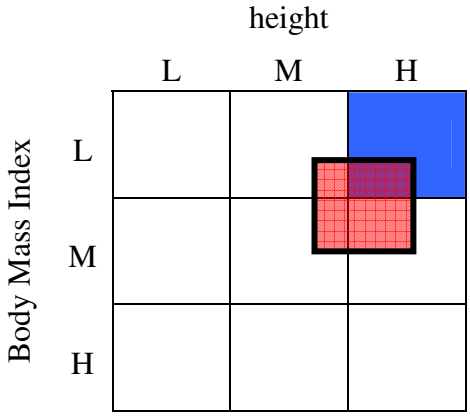


Fig. 3.14: *graphical representation of the membership of the input vector to the "suspect" category.*

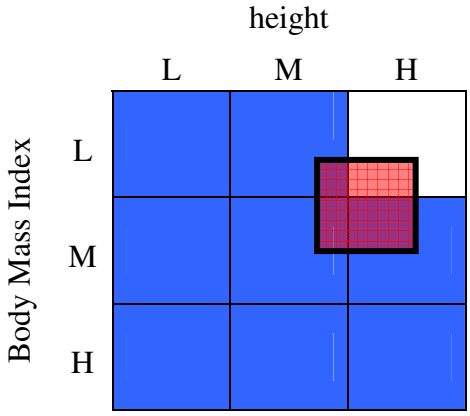


Fig. 3.15: *graphical representation of the membership of the input vector to the "not suspect" category.*

Matrix fuzzy engine

The implementation strategy of the "matrix" fuzzy engine consists of the attribution of output vectors to single sub-domains, through the rule set; then, total membership theorem is applied.
If p is the number of input parameters and **f** is the vector that specifies the number of fuzzy sets of each parameter, the number of sub-domains, D, is given by equation (3.34). Let N_{out} be the number of output categories. Given these premises, a matrix M ($D \times N_{out}$), that is, with D rows and N_{out} columns, can be set up, in such a way that the cell m_{ij} of M contains the value of the j-th element of the output vector ($j = 1, \ldots, N_{out}$) associated to the i-th sub-domain ($i = 1, \ldots, D$).
Therefore, the rule set is used to "fill up" the elements of M, and this process takes place only once (it represents the transfer of the knowledge to the machine). Successively, when an input vector has to be identified, the membership degrees of this vector to the sub-domains are evaluated, then equation (3.29) is applied.

Example

The same example described in the previous section will be considered here. Thus, $p = 2$, $\mathbf{f} = (3, 3)$ and $N_{out} = 2$. Therefore, matrix M is of the type shown in figure 3.16.

Sub-domains	Output categories	
	y_1	y_2
#1 → $k_1 = 1, k_2 = 1$		
#2 → $k_1 = 2, k_2 = 1$		
#3 → $k_1 = 3, k_2 = 1$		
#4 → $k_1 = 1, k_2 = 2$		

#5 → $k_1 = 2, k_2 = 2$		
#6 → $k_1 = 3, k_2 = 2$		
#7 → $k_1 = 1, k_2 = 3$		
#8 → $k_1 = 2, k_2 = 3$		
#9 → $k_1 = 3, k_2 = 3$		

Fig. 3.16: *representation of the matrix which stores the identifier knowledge.*

In matrix *M* sub-domains are listed by incrementing one of the elements of the index vector, **k**, at the time. Conventionally, the element which corresponds to the first parameter is incremented first, and so on. To make this process more clear, the mapped input space is shown in figure 3.17, together with the row index of each sub-domain.

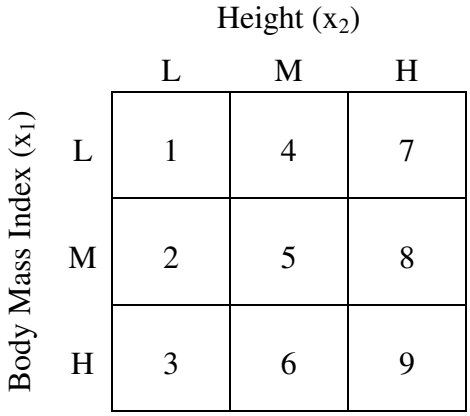


Fig. 3.17: *correspondence among sub-domains and rows of M.*

The first rule fills up the cells m_{7-1} and m_{7-2} of *M* with values 1 and 0, respectively. In fact, the rule states: "{ (x_1 is L) AND (x_2 is H) } ➔ ($y_1 = 1$, $y_2 = 0$)", hence **k** = (1, 3).
The second rule "{ (x_1 is M or H) OR (x_2 is L or M) } ➔ ($y_1 = 0$, $y_2 = 1$)" fills up all remaining cells of *M* with 0 in the first column and 1 in the second column.

Let the input vector be the one provided by table 5, that is, x = (18.6, 182). To evaluate the membership degrees of x to the sub-domains, it can be observed that they are all equal to 0, except for those pertinent to the sub-domains which are activated at the same time. The maximum number of sub-domains which are activated at the same time, Z, is provided by equation (3.36). From table 5 one can see that the indexes relevant to non zero memberships are:

x_1 (*BMI*) → {1, 2} $n_1 = 2$ (number of fuzzy sets activated for parameter 1)

x_2 (*h*) → {2, 3} $n_2 = 2$ (number of fuzzy sets activated for parameter 2)

The number of sub-domains which are activated at the same time by input vector x, Z_x, is given by equation (3.46).

$$Z_x = \prod_{i=1}^{p} n_i \tag{3.46}$$

In this case, $Z_x = 2 \times 2 = 4$. Hence, 4 index vectors ($\mathbf{k_1}$,…$\mathbf{k_4}$) must be determined, which identify the sub-domains activated by *x*. These vectors can be thought as the possible sorted couples of elements such that the first element is extracted from an urn with elements {1, 2} and the second is extracted from an urn with elements {2, 3}. In this case, the index vectors are:

$\mathbf{k_1} = (1, 2)$ $\mathbf{k_2} = (2, 2)$ $\mathbf{k_3} = (1, 3)$ $\mathbf{k_4} = (2, 3)$

As can be observed in figure 16, the sub-domains identified by vectors $\mathbf{k_1}$,…$\mathbf{k_4}$ correspond to rows 4, 5, 7 and 8 in matrix *M*, respectively.

To calculate the membership of *x* to these sub-domains equation (3.18) is used, as shown in (3.47).

$$\begin{aligned}
\mu_{k_1}(x) &= \mu_{1-1}(x_1) \cdot \mu_{2-2}(x_2) = 0.4 \cdot 0.3 = 0.12 \\
\mu_{k_2}(x) &= \mu_{1-2}(x_1) \cdot \mu_{2-2}(x_2) = 0.6 \cdot 0.3 = 0.18 \\
\mu_{k_3}(x) &= \mu_{1-1}(x_1) \cdot \mu_{2-3}(x_2) = 0.4 \cdot 0.7 = 0.28 \\
\mu_{k_4}(x) &= \mu_{1-2}(x_1) \cdot \mu_{2-3}(x_2) = 0.6 \cdot 0.7 = 0.42
\end{aligned} \tag{3.47}$$

Figure 16 reports values of function f_d for each activated sub-domain; therefore, the value of f_d in the sub-domain that corresponds to $\mathbf{k_1}$ consists of the fourth row of *M*, and so on.

Hence, the identification result is obtained by applying the total membership theorem. Clearly, the sum in equation (3.29) shall be extended to the activated sub-domains only, because for all the others the term $\mu_F(x)$ is equal to 0. The result is reported in (3.48).

$$\begin{aligned}
&f_{ID}(x) = \sum_{i \in \{4,5,6,7\}} f_d(F_i) \cdot \mu_{F_i}(x) = \\
&= (0,\ 1) \cdot 0.12 + (0,\ 1) \cdot 0.18 + (1,\ 0) \cdot 0.28 + (0,\ 1) \cdot 0.42 = \\
&= (0.28,\ 0.72)
\end{aligned} \tag{3.48}$$

It can be observed that (3.48) provides directly the desired result, without any need to merge the contributes of different rules; in addition, the output vector is always normalized.

Comparison of the two proposed strategies

Both of the described engines are consistent. In the following, a list of advantages and disadvantages is reported for the two strategies.

"Linguistic" fuzzy engine

Advantages

- Allows the developer (or the user) to keep contact with the identifier knowledge (interpretability). In other words, the identifier operation can be checked by the human being through the evaluation of linguistic rules.
- Is relatively simple and fast (in terms of machine executing time), when the dimension of the problem is high (large number of sub-domains, *D*) but, at the same time, the rule set is small (i.e. few rules for each output category).

Disadvantages:

- Does not promote the debugging / improvement process. In fact, adjustment to this kind of engine often involve the addition of new rules. A large number of rules is disadvantageous, because the identifier becomes slower and its interpretability decreases (in fact, "concepts" are actually associated to a restricted number of basic rules).
- Any modification to the identifier requires to change the implementation code.

"Matrix" fuzzy engine

Advantages

- Promotes the debugging / improvement process. Adjustment can be done either on "large scale", adding general rules, or locally, modifying the output of single sub-domains. Adjustments do not affect the identifier complexity (thus its speed), which is fixed (it is related to *D*).
- Is fast, even if the problem is very complicated.
- Modifications to the identifier involve changes in the matrix *M*, which can be loaded as external file, but do not affect the implementation code.

Disadvantages:

- The original meaning of linguistic rules gets lost. Interpretability is associated only to the possibility to inspect the matrix locally, that is, to check the output associated to a specific sub-domain, together with its neighborhood.
- If the rules used to build the matrix are few and, at the same time, *D* is very large, the identification process is handled in a way which is more complex than what is actually needed.

Fundamentals of artificial intelligence

Artificial intelligence is concerned with the learning of symbolic representations of concepts, therefore it is aimed at programming the machine in such a way that the program itself can improve with experience.

The design of an AI algorithm involves the following steps:

- Determine type of training experience
- Determine target function
- Determine representation of learned function
- Determine learning algorithm

Determine type of training experience

Training experience is provided by test examples. For this reason, much effort must be spent in carrying out PD measurement which lead to effective and reliable results. Hence, it is essential to have a good acquisition instrument and strategy, to get significant and sufficient information, and a good database handling program, in order to optimize the information available. In addition, training experience could be provided by some human expert, therefore the learning algorithm should be able to integrate prior knowledge, too.

Determine target function

The point is to focus on what type of knowledge will be learned, in order to select the output categories. The selection of output categories is a crucial aspect of defect identification. As was stated earlier, output categories are provided by the general identification strategy; however, AI tools should be able to provide a feedback about the output categories choice, evaluating the convenience to make adjustments to adapt those categories to the considered identification task.

Determine representation of learned function

This step consists of the choice of input parameters and of a way to relate these parameters to each other (e.g. input space mapping, linear combination, and so on). In other words, one must determine a way to formulate hypotheses about the target function, through a symbolic representation which can be handled by the machine.
Such a representation should be representative of the concepts to be learned and, at the same time, it

should allow interpretability and flexibility, so that the identification algorithm can be checked by human experts and merged with other algorithms.

Determine learning algorithm

The learning algorithm choice determines how the machine is supposed to search the hypotheses space (e.g. gradient descent, linear programming, distance evaluation, and so on), to find the hypothesis (or the hypotheses set) which best fits the observed data and any prior knowledge held by the learner. Therefore, a general optimization strategy has to be found.

Furthermore, the realization of AI algorithms is concerned with issues as the following:

- Questioning how mach data is sufficient for a given algorithm and a specific identification task.
- Enable the estimation of the confidence in learned hypotheses.
- Evaluating the possibility to alter automatically the hypotheses representation to improve its ability to fit the output categories.
- Create the conditions for the integration of prior knowledge even if it is only approximately correct.

A comparison can be made between the set up of AI algorithms, which regards machine learning, and the identification approach devised for machine teaching. Referring to the general description of the identification problem provided by (3.3), it can be stated that

- machine teaching consists of the definition of function f_{ID} according to prior knowledge,
- machine learning consists of the search for the best function f_{ID} according to a database (and, maybe, considering prior knowledge, too).

An important concept pertinent to AI is the so called “inductive bias”. Inductive bias can be defined as the set of assumptions that, together with the training data, deductively justify the identifications assigned by the learner, that is, the AI algorithm, to future instances. Hence, inductive bias consists of some kind of restriction for the identifier, which is essential to allow it to generalize from the training (observed) examples to identify unseen instances.

In the following paragraphs, two AI topics are dealt with: decision trees and concept learning. The former topic is aimed basically at the determination of function f_{ID} from a database, the latter is particularly useful for the estimation of the validity of f_{ID} and for the interaction with human knowledge. These two AI strategies are characterized by different inductive bias. Decision trees are provided with a “preference bias”, as they search incompletely a complete hypothesis space;

concept learning algorithms are characterized by "restriction bias", as they search completely an incomplete hypothesis space.
In this light, it must be observed that the fuzzy engines described with regard to machine teaching have no inductive bias, since no restriction was made on the possibility to formulate hypotheses. In fact, the formulation of a generic hypothesis is provided by assignment (3.49),

$$f_d(A) \leftarrow y \tag{3.49}$$

where $\boldsymbol{y}$ is the value specified for the output vector, and A is the subset of the input space to which the hypotheses (rule) refers. According to the described approach, A can be any subset, with no restriction. In fact, rule operators are defined in such a way that the rule set can refer to any subset A of the mapped input space. In terms of matrix fuzzy engine, a hypothesis consists of the association of a set of sub-domains to the output space. It is clear that no restriction was made about this process. Therefore, the described fuzzy engines are tautological, from an AI point of view, because they have no ability to generalize beyond the training examples (or the prior knowledge). Indeed, in machine teaching the prior knowledge is expected to cover all possible instances.

In the AI techniques described in the following paragraphs the training experience is provided by a database. Each record of such a database consists of an input vector, x, together with the correspondent output, y. In this case, let y belong to the set $\{1, 2, \dots, N_{out}\}$, where N_{out} is the number of output categories. As regards the input parameters, it must be underlined that these AI techniques are designed for discrete-valued parameters, which are called "attributes". Therefore, it will be necessary to remove this restriction, because the parameters calculated upon PD measurement data are typically continuous-valued.

In order to illustrate some of the basic concepts of AI, a simple identification problem is taken as example. The machine has to learn how to identify some fruit typology, i.e. apples, lemons and bananas, given three attributes: color, taste and shape. Hence, the number of output categories, N_{out}, equals 3, as well as the number, p, of input parameters (attributes). Let the attributes be discrete-valued as follows:
Color → {yellow (value #1), green (value #2), red (value #3)}
Taste → {sweet, sour, bitter}
Shape → {round, oblong}
Let the output categories, "apple", "lemon" and "banana", be associated to integers 1, 2 and 3, respectively. In analogy with machine teaching notation, let $\mathbf{f}$ be the vector which specifies the number of values for each attribute, $\mathbf{f} = (3, 3, 2)$. Let N be the number of records in the training data. Table 3.6 reports an example of training database for this identification task.

Tab. 3.6: *example of training database containing discrete-valued attributes and targets, which are associated to integers for simplicity.*

Record index	**attributes**						**Target value**	
	Color		**Taste**		**Shape**			
1	yellow	(1)	sweet	(1)	oblong	(2)	banana	(3)
2	yellow	(1)	sweet	(1)	round	(1)	apple	(1)
3	green	(2)	sweet	(1)	round	(1)	apple	(1)
4	red	(3)	sweet	(1)	round	(1)	apple	(1)
5	yellow	(1)	sour	(2)	round	(1)	lemon	(2)
6	yellow	(1)	sweet	(1)	oblong	(2)	banana	(3)
7	green	(2)	sweet	(1)	round	(1)	apple	(1)
8	yellow	(1)	sour	(2)	round	(1)	lemon	(2)
9	yellow	(1)	sour	(2)	round	(1)	lemon	(2)
10	red	(3)	sweet	(1)	round	(1)	apple	(1)

Definitions

- *Instance*, *x*: any vector of p elements which specifies a value for each attribute. It corresponds to the input vector defined for machine teaching. Example: *x* = < red, sour, round >

- *Set of all instances, X*: is the set of all instances associated to the attributes. It corresponds to the set of sub-domains in machine teaching. The number of elements (instances) of *X* is given by equation (3.34).
It can be observed that the records of the training database provides particular instances, for which a target function is defined. Let the set of training database instances be $X_{database}$. Let x_k be the instance associated to the k-th record of the training database.

- *Hypothesis, h*: function that specifies the relation between an instance and a specific output category. For the k-th output category $h_k : X \rightarrow \{0,\ 1\}$
It corresponds to function f_d defined for machine teaching.

- *Target function, c*: function that associates training examples (instances) to the output space.
$$c : X_{database} \rightarrow \{1,...,N_{out}\}$$

- *Target vector, **t***: vector of integers. The k-th element of **t** indicates the output category associated to the k-th record of the training database, that is, $t_k = c(x_k)$, $\forall k = 1,...,N$.

- *Positive / negative example*: a database example is positive with respect to a given output category

if it is associated by the target function to that category.

$\forall x \in X_{database}$ x is positive with repect to the k-th output category if $c(x) = k$

- *Power set,* H^P: is the set of all possible hypotheses which can be formulated with a given set of attributes. It consists of the set of all possible subsets of X. The number of elements of the power set, D_P, is given by equation (3.50),

$$D_P = 2^D \tag{3.50}$$

where D is the number of elements of X.

Observation: D_P includes also the two "improper" subsets, that is, X itself and the null set. In fact, $2^n = \sum_{k=0}^{n} \binom{n}{k}$.

- *Hypotheses space,* H: is the set of all hypotheses which is searched by the learning algorithm. It is a subset of the power set. In particular, if no restriction is given, H and H^P coincide.

Decision trees

Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. Learned trees can also be represented as sets of if-then rules to improve human readability.
A tree is set up on the basis of a training database, $X_{database}$, consisting of a set of instances, each one provided with a target value. Hence, a tree algorithm consists of a function, *T*, which associates a given instance, *x*, to an output vector, *y*.

$$T(x): X \to Y \tag{3.51}$$

Let $X_{database\text{-}k}$ be the set of positive examples for the k-th output category and let n_k be the number of these examples. The number of *misclassified examples* relevant to the k-th output category, m_k, can be defined for a given tree, *T*, as the number of instances in $X_{database\text{-}k}$ which are associated by function *T* to categories other then the k-th; let *M* be the total number of misclassified examples.
With these premises, the *efficiency,* η, of a tree with respect to its training dataset (or any dataset provided with targets) is defined by equation (3.52).

$$\eta(T, X) = \frac{\sum_{k=1}^{N_{out}} n_k - m_k}{\sum_{k=1}^{N_{out}} n_k} = \frac{N - M}{N} \tag{3.52}$$

Likewise, the *relative efficiency,* η_k, pertinent to a specific output category, is defined by (3.53).

$$\eta_k(T, X) = \frac{n_k - m_k}{n_k} \tag{3.53}$$

Furthermore, a way to calculate a tree efficiency alternative to equation (3.52) is represented by equation (3.54). This quantity will be called “mean efficiency”.

$$\bar{\eta}(T, X) = \frac{\sum_{k=1}^{N_{out}} \eta_k}{N_{out}} \tag{3.54}$$

Clearly, equation (3.54) averages the efficiency factors relevant to single categories, therefore it assigns the same weight to every output category. On the contrary, equation (3.52) assigns to output categories weights in proportion to their size in the database (i.e. in proportion to n_k / N).

A tree is made of nodes; each node involves a check with respect to a given attribute. An example of tree node is reported in figure 3.18.

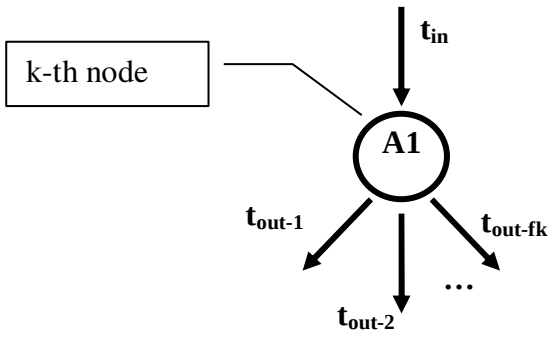


Fig. 3.18: *schematic representation of the k-th tree node, pertinent to the first attribute.*

Any node can be seen as a function characterized by one input and a at least two outputs. The input, $\mathbf{t_{in}}$, consists of the target vector, **t**, or a subset of **t**. The number of outputs of a node which is relevant to the j-th attribute is equal to the number of values of that attribute (f_j). The outputs consist of subsets of the input target vector, obtained partitioning $\mathbf{t_{in}}$ with respect to the j-th attribute. In particular, the h-th output of a node associated to the j-th attribute is given by (3.55),

$$t_{out-h} = \left(t_{in}\left(k\right)\right),\ k : x_k\left(j\right) = h \tag{3.55}$$

where k is the index of the elements of $\mathbf{t_{in}}$ for which the j-th attribute assumes its h-th value.

Clearly, the tree algorithm is iterative and the number of iterations equals the number of nodes. The node corresponding to the first iteration is called "root node" and takes as input the whole target vector, **t**, which has length *N*. The goal of this process is to get subsets of **t** which are either empty or characterized by all equal elements, indicating that the training dataset is separated in subsets which are homogeneous with respect to the output category.
At this point, it is essential to provide the tree construction algorithm with a criterion to choose the best strategy, i.e. to provide an inductive bias. In other words, one must determine how the hypotheses space, *H*, shall be searched, holding that no specific restriction is made about *H*. In this light, a basic assumption is made, affirming that, generally speaking, the simplest hypothesis shall be preferred. Such an assumption is often called "Occam's razor", because William of Occam[8] was the first to discuss this question, apparently while shaving.

[8] William of Occam (XIII century) was a philosopher and a Franciscan, who used to teach at Oxford University; the original formulation of his famous principle is: «*Pluralitas non est ponenda sine necessitate*».

Thus, the best tree would be the simplest (smallest) tree which fits the data. In order to minimize the tree size, at each iteration the algorithm should select the attribute which is most efficient with respect to identification (data separation according to the target function). Therefore, a technique is required to evaluate the efficiency of attributes with respect to the target vector. For this purpose, two basic concepts proper of information theory will be introduced: entropy of a dataset and information gain of an attribute [3].

Entropy is a positive real quantity which evaluates the "impurity" of a collection of examples. It is calculated on the basis of the target vector, as shown by equation (3.56).

$$entropy(t) = \sum_{k=1}^{N_{out}} -\frac{n_k}{N} \cdot \log_2\left(\frac{n_k}{N}\right) \tag{3.56}$$

Examples:

The target vector consists of only one value, thus it is homogeneous with respect to the correspondent output category.
t = (1, 1, 1, 1, 1, 1) ➔ entropy(**t**) = 0

The target vector consists of two output categories, which have the same size in the database.
t = (1, 1, 1, 2, 2, 2) ➔ entropy(**t**) = 1

Entropy associated to the database of table 6:
t = (3, 1, 1, 1, 2, 3, 1, 2, 2, 1) ➔

$$entropy(t) = -\frac{5}{10} \cdot \log_2\left(\frac{5}{10}\right) - \frac{3}{10} \cdot \log_2\left(\frac{3}{10}\right) - \frac{2}{10} \cdot \log_2\left(\frac{2}{10}\right) =$$

$$-0.5 \cdot (-1) - 0.3 \cdot (-1.74) - 0.2 \cdot (-2.32) = 1.49$$

Information gain evaluates the efficiency of an attribute in the output categories separation. It is function of the target vector, **t**, and of the vector, **a**, of the values of a specific attribute, A. It is calculated as the reduction of entropy associated to the partitioning of **t** with respect to **a**, according to equation (3.57),

$$gain(t,a) = entropy(t) - \sum_{k=1}^{v(a)} \frac{n_k}{N} \cdot entropy(t_k) \tag{3.57}$$

where v(**a**) is the number of values that the considered attribute assumes in **a**, n_k is the number of elements of **a** which assume the k-th value and $\mathbf{t_k}$ is the subset of **t** for which all correspondent values of **a** assume the k-th value.

Example.
With reference to the database of table 6, the information gain associated to the third attribute (shape) is calculated. In this case, **a** = (2, 1, 1, 1, 1, 2, 1, 1, 1, 1), therefore v(**a**) = 2. Hence, partitioning **t** with respect to **a**, two sub-databases are obtained, together with two subsets of the target vector, $\mathbf{t}|_{a=1}$ and $\mathbf{t}|_{a=2}$. The entropy of the partitioned database is equal to the weighted sum of the entropies of the single databases.

$$gain(t,a) = 1.49 - \frac{8}{10}\cdot\left[-\frac{6}{8}\cdot\log_2\left(\frac{6}{8}\right)-\frac{2}{8}\cdot\log_2\left(\frac{2}{8}\right)\right]-\frac{2}{10}\cdot\left[-\frac{2}{2}\cdot\log_2\left(\frac{2}{2}\right)\right]=$$
$$= 1.49 - 0.8\cdot 0.81 - 0.2\cdot 0 = 0.84$$

At this point, it would be possible to describe one of the most common algorithms for the construction of decision trees, which is referred to in literature as ID3 [3, 20]. Its basic principle is that the tree is grown top-down, iteratively; at each iteration the best attribute is selected (i.e. the attribute with highest value of information gain) and a node is created accordingly. Each node has as many branches output as the values assumed by its attribute; each branch is associated to a subset of the input database. If a database subset is characterized by null entropy, the correspondent branch is connected to tree ending (leaf), otherwise, the same procedure is called, recursively.
It is noteworthy to observe that this kind of tree, constructed on the basis of discrete-valued attributes, does not necessarily separate all records in the training dataset. In fact, attributes which have been incorporated higher in the tree are excluded, so that any given attribute can appear at most once along any tree path. Therefore, if all available attributes are already included, the tree growth stops, even if some of the database subsets (at the tree endings) have entropy greater than zero. This kind of algorithm will not be described in detail, because it is not applicable to the case under study in this PhD thesis. In fact, here it must be assumed that attributes, that is, input parameters, are continuous-valued.
Hence, a technique is required to convert a given attribute from continuous-valued to discrete-valued. The adopted approach consists of the evaluation, for each attribute vector **a**, of a correspondent boolean vector **b**, as specified by equation (3.58).

$$\begin{aligned} &b(k) = 1 \text{ if } a(k) \le c,\ b(k) = 0 \text{ if } a(k) > c \\ &\text{with } c \in [\min(a),\ \max(a)] \end{aligned} \tag{3.58}$$

The selection of the best value for the “threshold” c is aimed at the maximization of the information gain. Therefore, the values of the information gain for attribute **a** could be plotted as a function of c, with c ranging between the minimum and the maximum value of **a** in the database. An example of the result of this procedure, applied to a practical case (II level of identification / shape), is reported in figure 3.19.

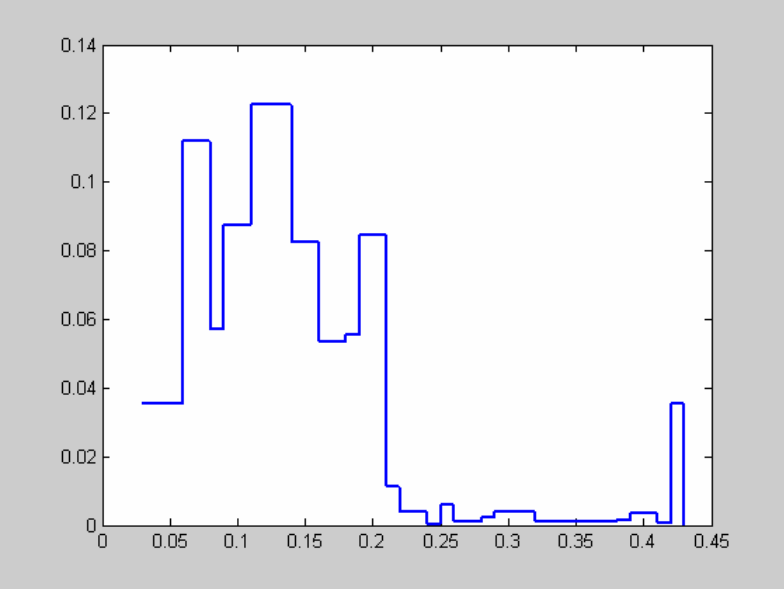


Fig. 3.19: *information gain as a function of a continuous-valued attribute.*

In the case of figure 3.19, the information gain is calculated for 1000 values of the threshold c. It can be seen that there are several relative peaks of the information gain. Although effective, this procedure is rather slow. Indeed, a more efficient technique can be adopted to find the optimum value for c. In fact, by sorting the target vector **t** according to the continuous-valued attribute, then identifying the adjacent records which have different values of **t**, a set of candidates for the value of c are generated; actually, it can be shown [3] that the peaks of the information gain must lie in correspondence of values of **a** relevant to those records. Figure 3.20 shows the values of the information gain calculated only in correspondence of the relative peaks, for the same example as figure 3.19.

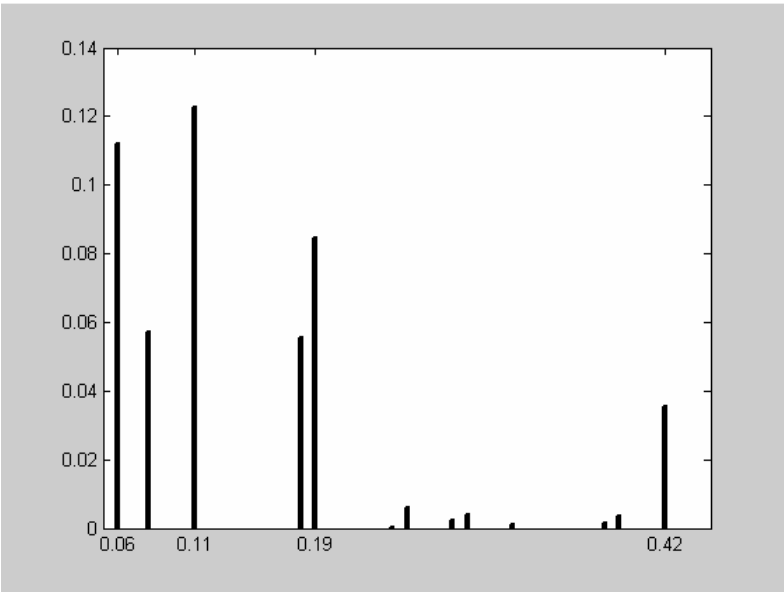


Fig. 3.20: *relative peaks of the information gain as a function of a continuous-valued attribute.*

As can be seen, the optimal value c=0.11 can be derived from both figures, but Fig. 3.20 has been

evaluated at a speed about 100 times higher than Fig. 19.
Thus, at any stage of the tree growth, it is possible to select, for each continuous-valued attribute, the set of values for which the information gain has a peak; accordingly, it is possible to find the attribute which is associated to the highest information gain overall (the most significant at that stage). It is important to observe that, contrarily to the case of discrete-valued attributes, in the case of continuous-valued attributes an attribute may be used several times along the same path through the tree, with different values of threshold c.

Therefore, an algorithm was set up, ID3n3, which allows to construct a decision tree on the basis of a training database with continuous-valued attributes; its basic structure is reported in table 3.7.

Tab. 3.7: *schematic description of algorithm ID3n3 (basic version)*

- For each attribute, a_k, find the best threshold: c_{kh}
($h = 1 + n_k$, where n_k is the number of previous occurrences of the k-th attribute in the tree)
- Find the most efficient attribute and the correspondent threshold: a_k , c_{kh}
- Create a node with attribute a_k and threshold c_{kh}
- FOR each value $v \in \{0, 1\}$ of the Boolean vector $\mathbf{b_k}$ obtained imposing ($\mathbf{a}_k \leq c_{kh}$)
    - Add an output (branch) to the node and evaluate the sub-database such that $\mathbf{b_k}(i) = v \ \forall i$
    - IF that sub-database has null entropy
        - Put a leaf below that branch, with label equal to any element of the target vector of the sub-database.
    - ELSE below that branch add the sub-tree obtained running ID3n3 on the current sub-database.
- END

Considerations about algorithm ID3n3

- It ends when the training data is fully separated; in other words, it returns a tree with efficiency equal to 1.
- It does not necessarily use all of the attributes contained in the training database; therefore it performs a selection of the most efficient attributes. Let p be the number of selected attributes (input parameters).
- Each parameter may occur more than once in the tree.

- Example

To make an example, a database relevant to the second level of identification will be taken as reference. In particular, the task regarding the identification of the defect shape, with distinction between flat and compact defects, is taken into account. Such a database is reported in table 3.8.

Tab. 3.8: *example database, relevant to the second level of identification task "defect shape".*

Record index	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	Target
1	0.22	0.23	0.33	0.32	1.00	1.00	0.00	0.00	0.00	0.00	6.8	8.0	1
2	0.38	0.37	0.44	0.37	1.00	1.00	0.00	0.00	0.00	0.00	4.1	4.1	1
3	0.21	0.33	0.30	0.39	1.00	1.00	0.00	0.00	0.00	0.00	3.3	4.3	1
4	0.32	0.23	0.43	0.24	1.00	1.00	0.00	0.00	0.00	0.00	3.8	3.7	1
5	0.24	0.12	0.40	0.15	1.00	1.00	0.00	0.00	0.00	0.00	4.0	5.7	1
6	0.16	0.25	0.27	0.29	1.00	1.00	0.00	0.00	0.00	0.00	3.2	3.8	1
7	0.42	0.29	0.43	0.30	1.00	1.00	0.00	0.00	0.00	0.00	4.0	4.2	1
8	0.21	0.27	0.21	0.33	1.00	0.99	0.00	0.01	0.00	0.00	4.0	2.9	1
9	0.28	0.33	0.30	0.35	1.00	0.03	0.00	0.97	0.00	0.00	2.3	1.8	1
10	0.21	0.31	0.42	0.34	1.00	0.74	0.00	0.26	0.00	0.00	2.7	1.8	1
11	0.26	0.23	0.29	0.29	0.65	0.50	0.35	0.50	0.00	0.00	1.6	1.4	1
12	0.14	0.23	0.18	0.24	1.00	1.00	0.00	0.00	0.00	0.00	2.8	2.7	1
13	0.08	0.26	0.10	0.26	1.00	1.00	0.00	0.00	0.00	0.00	3.4	8.3	1
14	0.18	0.35	0.27	0.36	0.86	0.83	0.14	0.17	0.00	0.00	2.0	2.4	1
15	0.32	0.34	0.41	0.35	1.00	1.00	0.00	0.00	0.00	0.00	2.9	2.5	1
16	0.29	0.29	0.32	0.31	0.22	0.13	0.78	0.87	0.00	0.00	1.4	1.4	1
17	0.41	0.25	0.47	0.31	1.00	1.00	0.00	0.00	0.00	0.00	4.5	3.3	1
18	0.08	0.21	0.11	0.27	1.00	1.00	0.00	0.00	0.00	0.00	1.6	1.9	1
19	0.21	0.41	0.24	0.41	1.00	1.00	0.00	0.00	0.00	0.00	2.5	2.4	1
20	0.32	0.33	0.42	0.34	1.00	1.00	0.00	0.00	0.00	0.00	3.9	3.0	2
21	0.08	0.19	0.12	0.22	0.11	0.82	0.89	0.18	0.00	0.00	2.0	2.6	2
22	0.06	0.09	0.07	0.12	0.36	0.16	0.64	0.84	0.00	0.00	1.8	2.1	2
23	0.03	0.10	0.04	0.11	1.00	1.00	0.00	0.00	0.00	0.00	3.3	3.4	2
25	0.09	0.25	0.37	0.31	1.00	1.00	0.00	0.00	0.00	0.00	3.4	4.1	2
25	0.25	0.22	0.43	0.23	1.00	1.00	0.00	0.00	0.00	0.00	5.2	5.9	2
26	0.29	0.23	0.49	0.25	1.00	1.00	0.00	0.00	0.00	0.00	5.7	5.7	2
27	0.11	0.22	0.13	0.27	1.00	1.00	0.00	0.00	0.00	0.00	7.0	5.1	2
28	0.43	0.25	0.43	0.26	1.00	1.00	0.00	0.00	0.00	0.00	3.4	3.6	2
29	0.28	0.28	0.41	0.33	0.95	0.64	0.05	0.36	0.00	0.00	2.7	2.6	2
30	0.18	0.37	0.26	0.40	1.00	1.00	0.00	0.00	0.00	0.00	3.2	3.2	2
31	0.08	0.29	0.18	0.29	0.95	0.91	0.05	0.09	0.00	0.00	2.4	2.0	2
32	0.39	0.31	0.39	0.33	0.93	0.80	0.07	0.20	0.00	0.00	2.1	1.9	2
33	0.06	0.08	0.07	0.13	1.00	1.00	0.00	0.00	0.00	0.00	3.0	2.6	2
34	0.19	0.36	0.24	0.40	0.89	1.00	0.11	0.00	0.00	0.00	2.0	2.1	2

As can be seen in table 3.8, twelve attributes are considered[9]. Of the 34 training examples, 19 are associated to the first output category (compact defect), 15 to the second output category (flat defect). The results of algorithm ID3n3 applied to this database will be presented in the following.
At the first iteration, the first attribute (a1) is selected as the most significant[10]. Therefore, the root

[9] They were described in detail in the second section.

[10] The values of its information gain (peaks) were reported in Figs. 19 and 20.

node of the tree is associated to attribute a1. The result of the attribute sorting with respect to information gain at the first iteration is reported in table 3.9.

Tab. 3.9: *result of the attribute sorting at the first iteration (root node).*

Attribute	Rank	Information gain	Corresponding ID marker
a1	1	0.1227	min(pF1Pos, pF1Neg)
a2	4	0.1121	min(pF2Pos, pF2Neg)
a3	3	0.1121	mean(pF1Pos, pF1Neg)
a4	2	0.1121	mean(pF2Pos, pF2Neg)
a5	7	0.0538	IntPos
a6	10	0.0515	IntNeg
a7	6	0.0538	SurPos
a8	9	0.0515	SurNeg
a9	12	0	CorPos
a10	11	0	CorNeg
a11	5	0.0789	betaPos
a12	8	0.0518	betaNeg

Table 9 shows that the first four attributes have a quite larger value of information gain with respect to the others. It is very important to remark that the attribute efficiency evaluation provided by the calculation of the information gain indicates which attribute would give the best separation (identification) result if only one attribute was available or, more precisely, if the tree ended at that node. In other words, the maximum information gain selects the best attribute at a given step, but does not account for further tree growth.
With reference to the section 2 of this thesis, it can be observed that the attribute selection (ranking) provided by decision trees is coherent with the interpretation criteria derived by human expert on the basis of experimental observation.

Therefore, one might decide to extract a sub-database containing all of the records but only the first four attributes, and run ID3n3 on such a database. In the following, the results will be reported of the application of ID3n3 to both the entire database and the reduced database.
Once the tree is set up, the selected attributes will be called (input) parameters, in analogy with the theory described for machine teaching.

The result obtained with the entire database (table 3.8) is reported in figure 3.21.

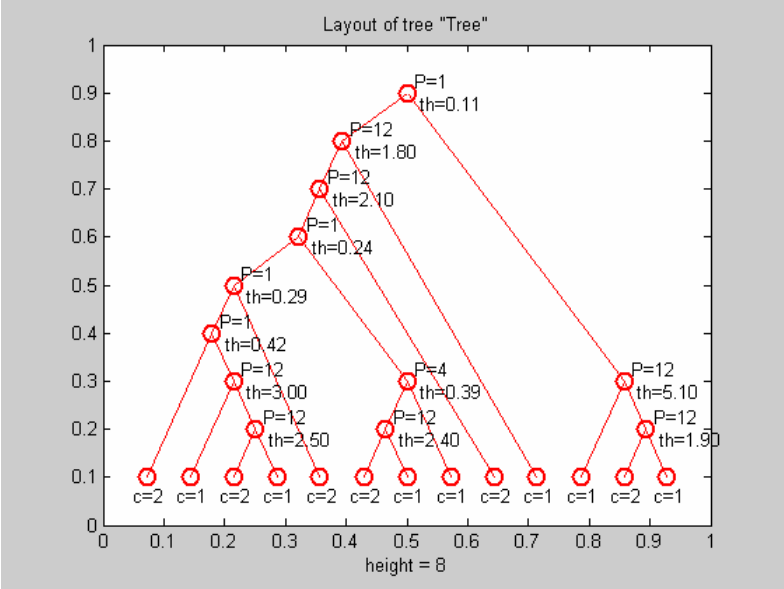


Fig. 3.21: *representation of the tree algorithm obtained with the entire dataset (12 attributes).*

In Fig. 3.21 "P=1" indicates that a given node is associated to the first parameter; "th=0.11" indicates the value of the parameter threshold for a given node; "c=1" indicates that a given leaf is associated to the first output category, i.e. it is equivalent to the statement "$y_1 = 1$ and $y_k = 0, \forall\ k \neq 1$"). The maximum number of nodes that lay on the same path from the root down to any leaf will be called "tree heigth".

The tree obtained with the database restricted to the four most significant attributes[11] is reported in figure 3.22.

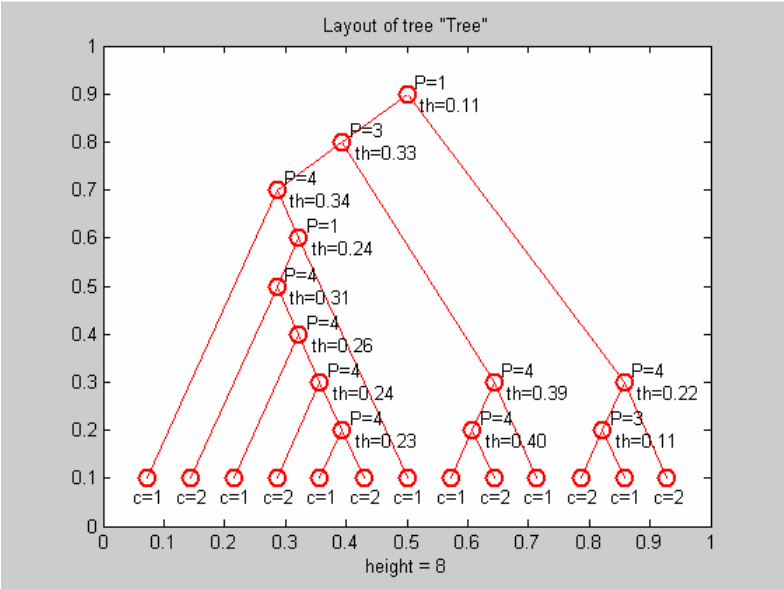


Fig. 3.22: *representation of the tree algorithm obtained with the dataset reduced to the 4 most significant attributes.*

[11] According to the attribute sorting performed at the first step of the tree construction (root node).

Both of the trees reported in Figs. 3.21 and 3.22 have twelve nodes and use three parameters. In particular, the parameters of the tree of Fig. 3.21 correspond to attributes 1, 4 and 12 in the training database (table 3.8), those of Fig. 3.22 correspond to attributes 1, 3 and 4. As mentioned before, both of these trees are characterized by an efficiency value equal to 1.

Thus, ID3n3 provides a discretization of the selected attributes (input parameters). In fact, it takes as input a training dataset and returns the thresholds which were used in the nodes, each one associated to a specific parameter. If the k-th parameter occurs n times in the tree, that is, n nodes are there which belong to that parameter, it is discretized in $f_k = n+1$ values. Clearly, such a discretization is crisp, and it is well represented by Fig. 8, which exemplifies the fuzzyfication process with fuzziness degree equal to 0.

For example, parameter 1 of the tree of Fig. 3.22 (attribute 1 in the database of table 3.8) is associated to two nodes, with thresholds 0.11 and 0.24. The discretization of parameter 1 provided by these two thresholds is shown qualitatively in figure 3.23 (interval limits are 0 and 1).

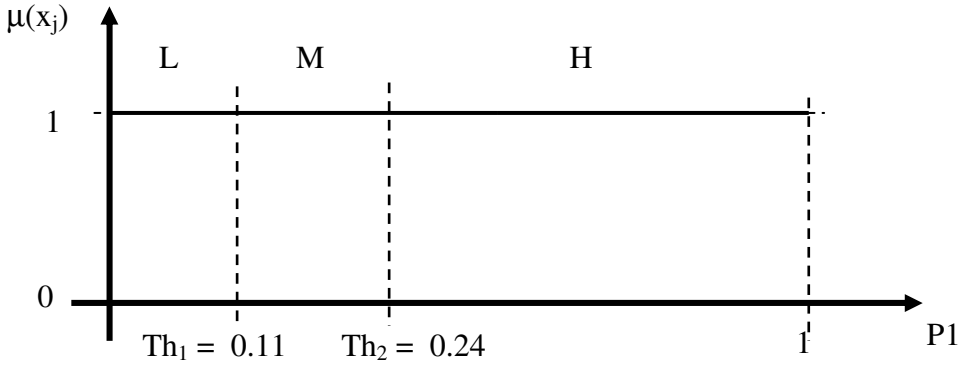


Fig. 3.23: *qualitative representation of the (crisp) discretization of par. 1, according to the tree of Fig. 22,*

Hence, a new dataset could be derived with the same target vector, such that the selected attributes are discrete-valued (e.g. parameter 1 would have values “low”, “medium” and “high”).

It is quite evident that the tree algorithms represented in Figs. 3.21 and 3.22 overfit the training dataset, that is, they are too specialized on the training examples and they loose generality. In fact, one can notice that these trees contain a large number of nodes associated to few parameters with thresholds close to each other. Indeed, overfitting is a crucial problem for tree algorithms, as well as for the majority of the AI techniques. As regards decision trees, to reduce overfitting consists basically of a simplification of the tree, i.e. a reduction of its number of nodes, according to Occam’s razor. Such a procedure is called pruning. There are two possible approaches to pruning:

- During the tree growth → pre-pruning;
- After the tree growth is complete → post-pruning.

As regards pre-pruning, the point is to determine, at each step of the tree growth (in correspondence of a node output where entropy is not null), if it is worth to proceed with a further node, rather than

putting a leaf, accepting a certain number of misclassified examples. In this light, a version of ID3n3 will be described in table 3.10, which is provided with a pre-pruning procedure.

Tab. 3.10: *schematic description of algorithm ID3n3 (version provided with pre-pruning)*

- Find the most efficient attribute and the correspondent threshold: a_k , c_{kh}
($h = 1 + n_k$, where n_k is the number of previous occurrences of the k-th attribute in the tree)
- Create a node with attribute a_k and threshold c_{kh}
- FOR each value $v \in \{0, 1\}$ of the Boolean vector $\mathbf{b_k}$ obtained imposing ($\mathbf{a}_k \leq c_{kh}$)
    - Add an output (branch) to the node and evaluate the sub-database such that $\mathbf{b_k}(i) = v \ \forall i$
    - FOR each value, s , of the target vector of the current sub-database
        - Evaluate the reduction of efficiency, $\Delta\eta$, if the tree associated every record of the current sub-database to the category s.
    - END
    - IF that sub-database has null entropy OR $\Delta\eta_{min} < \Delta\eta_{REF}$
        - Put a leaf below that branch, with label equal to the value of the target vector which corresponds to $\Delta\eta_{min}$.
    - ELSE below that branch add the sub-tree obtained running ID3n3 on the current sub-database.
- END

Observation

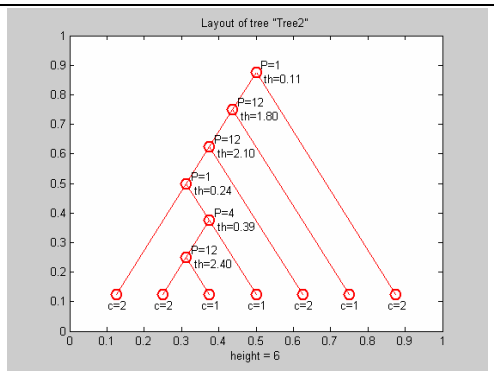
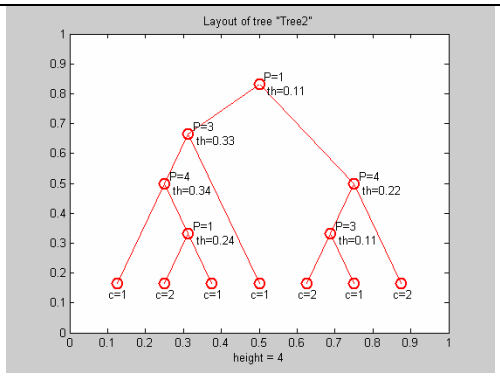
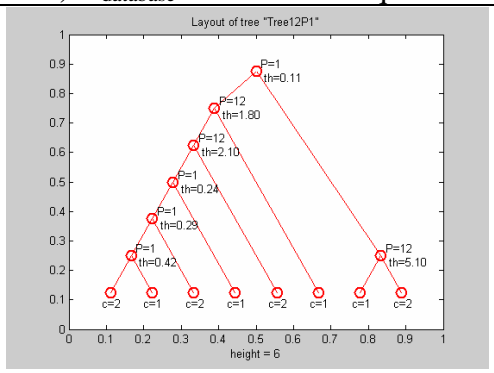
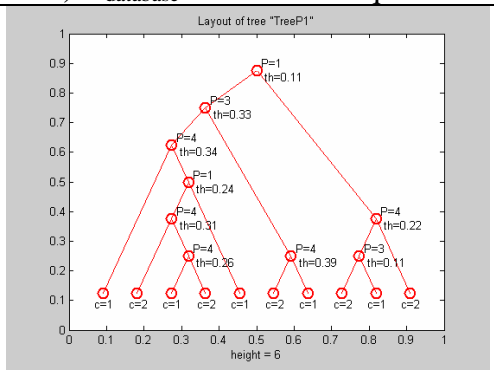
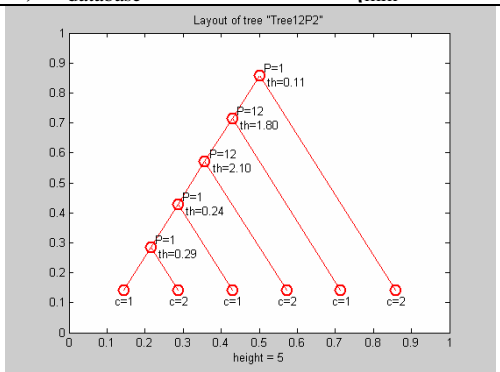
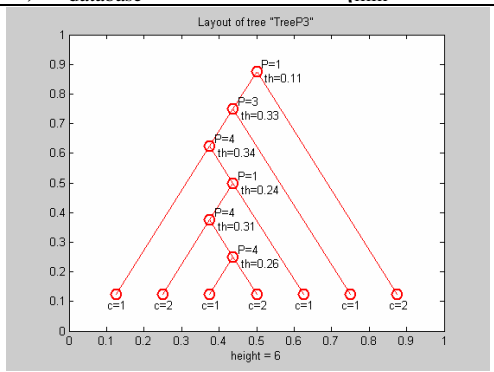
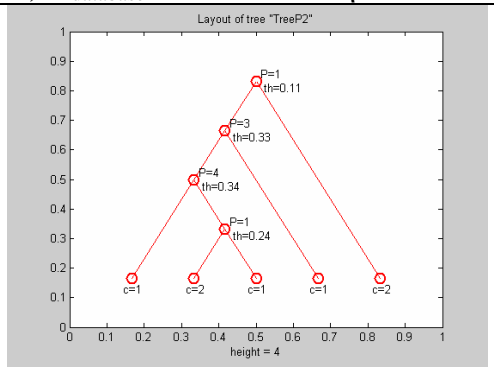
The efficiency of a pruned tree is strictly lower than 1. Therefore, the total number of misclassified examples, M, is larger than 0, and each output category is associated to a number of misclassified examples m_k ranging in $[0, n_k]$, where n_k is the number of example instances for the k-th category.

As regards post-pruning, the situation is more complex with respect to pre-pruning and will be discussed in deeper detail later. However, it can be observed that a tree where a given parameter is associated to a very large number of nodes (large also relatively to the other parameters) is not desirable. Therefore, an algorithm can be set up to prune the tree in such a way that all parameters are associated to the same number of nodes, equal to the minimum number of nodes pertinent to the same parameter (n_{min}). A tree which belongs to this typology will be called “squared”.

The tree squaring procedure is performed through a down-top iterative algorithm. The first nodes to be analyzed are the ones of high levels (close to the leafs); at each iteration, the current node is pruned if the parameter pertinent to that node occurs in the tree more than n_{min} times. When a node is pruned, in its place is put a leaf, associated to the target value to which corresponds the lowest efficiency reduction.

Table 3.11 shows the result of tree squaring and of the application of ID3n3 with pre-pruning on the training dataset of table 3.8 (12 attributes) and on the same dataset reduced to 4 attributes.

Tab. 3.11: *examples of pre-pruning and squaring of the trees of Figs. 3.21 and 3.22.*

 A) $X_{database}$ with 12 attributes - squared	 E) $X_{database}$ with 4 attributes - squared
 B) $X_{database}$ with 12 attributes - $\Delta\eta_{min}$= 0.050	 F) $X_{database}$ with 4 attributes - $\Delta\eta_{min}$= 0.050
 C) $X_{database}$ with 12 attributes - $\Delta\eta_{min}$= 0.075	 G) $X_{database}$ with 4 attributes - $\Delta\eta_{min}$= 0.075
Same as C	
D) $X_{database}$ with 12 attributes - $\Delta\eta_{min}$= 0.100	H) $X_{database}$ with 4 attributes - $\Delta\eta_{min}$= 0.100

Since a tradeoff exists between the complexity of a tree and its efficiency, it is important to define a quantity which evaluates the quality of a tree algorithm weighing both factors. Such a quantity is

provided by the tree function described by equation (3.59),

$$fitness(Tree) = \frac{\eta}{z^{0.2}} \tag{3.59}$$

where z is the number of nodes in the tree, excluding leafs and η is either the tree efficiency (equation (3.52)) or the tree mean efficiency (equation (3.54)).
Table 3.12 reports the number of nodes, the efficiency and mean efficiency for the trees of table 11, as well as the values of tree fitness calculated in both ways.

Tab. 3.12: *characterization of efficiency and fitness of the trees of table 11.*

Tree description	N. of nodes	efficiency	mean efficiency	fitness	(mean) fitness
A) $X_{database}$ with 12 attributes - squared	6	0.8529	0.8544	0.5960	0.5971
B) $X_{database}$ with 12 attributes - $\Delta\eta_{min}$= 0.050	7	0.9412	0.9474	0.6065	0.6105
C-D) $X_{database}$ with 12 attributes - $\Delta\eta_{min}$= 0.075	5	0.8529	0.8544	0.5960	0.5971
E) $X_{database}$ with 4 attributes - squared	6	0.7941	0.8018	0.6018	0.6077
F) $X_{database}$ with 4 attributes - $\Delta\eta_{min}$= 0.050	9	0.7941	0.8158	0.5549	0.5701
G) $X_{database}$ with 4 attributes - $\Delta\eta_{min}$= 0.075	6	0.9118	0.9070	0.6178	0.6146
H) $X_{database}$ with 4 attributes - $\Delta\eta_{min}$= 0.100	4	0.8542	0.8474	0.6191	0.6142

Before describing more advanced post-pruning techniques, it is important to analyze the possibility to derive from a tree algorithm its equivalent linguistic rules and to transform the crisp intervals provided by ID3n3 (according to threshold values) in fuzzy sets.

As regards the translation into linguistic rules, it is quite evident that a given tree is equivalent to a set of N_R rules, where N_R is the number of its leafs. As example, the translation of the tree obtained by squaring the tree of table 22 (table 11-E) is reported in the following.

Rule 1)[12] `IF { ( Par#1: Set2 or Set3 ) AND ( Par#2: Set3 ) AND ( Par#3: Set3 ) } ==> Category#1`

Rule 2) `IF { ( Par#1: Set3 ) AND ( Par#2: Set3 ) AND ( Par#3: Set1 or Set2 ) } ==> Category#2`

Rule 3) `IF { ( Par#1: Set2 ) AND ( Par#2: Set3 ) AND ( Par#3: Set1 or Set2 ) } ==> Category#1`

Rule 4) `IF { ( Par#1: Set2 or Set3 ) AND ( Par#2: Set1 or Set2 ) AND ( Par#3: Set1 or Set2 or Set3 ) } ==> Category#1`

[12] Rules are listed according to the position of the correspondent leafs in the tree, from the left to the right.

Rule 5) `IF { ( Par#1: Set1 ) AND ( Par#2: Set2 or Set3 ) AND ( Par#3: Set2 or Set3 ) } ==> Category#2`

Rule 6) `IF { ( Par#1: Set1 ) AND ( Par#2: Set1 ) AND ( Par#3: Set2 or Set3 ) } ==> Category#1`

Rule 7) `IF { ( Par#1: Set1 ) AND ( Par#2: Set1 or Set2 or Set3 ) AND ( Par#3: Set1 ) } ==> Category#2`

In the following, a procedure is proposed to change the discretization type from crisp to fuzzy. It consists in fuzzyfying the input parameters taking advantage of the thresholds provided by ID3n3. It is important to observe that the values that delimit the parameter domain (interval limits) must be known (they shall be provided by the parameter definition).

For seek of simplicity, the shape of the fuzzy sets is supposed to be trapezoidal. What is known is the number of fuzzy sets, f, the two interval limits and the f-1 threshold values provided by ID3n3. It is clear by the observation of Fig. 3.2 that the number of reference values needed to perform a fuzzyfication with f trapezoidal sets is 2*f (including the interval limits). Therefore, the number of references to be determined is 2*(f-1), that is, two references must be derived from each threshold. To describe this process, an example case will be used, with f = 3, as illustrated in figure 3.24.

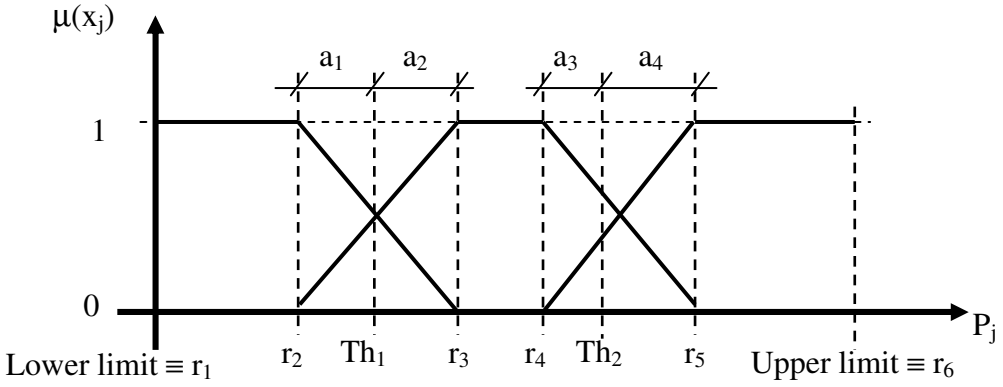


Fig. 3.24: *description of the procedure of automatic fuzzyfication with trapezoidal sets.*

To determine the unknown references (r_2,…, r_5 in Fig. 24), it is sufficient to determine an equal number of distances (a_1,…, a_4) of the references themselves from the threshold values. Hence, r_2 will be obtained as Th_1-a_1, r_3 as Th_1+a_2, and so on. Thus, 2*(f-1) linearly independent equations must be found, to determine univocally quantities a_k, then r_k. For this purpose, it is assumed that fuzziness should be more accentuated where the number of misclassified examples is higher ("fuzzyfication efficiency assumption"). In fact, if the efficiency of an algorithm is 1, there is no purpose for fuzzyfication; indeed, in such a case the algorithm would probably be subjected to overfitting, but this is a matter of pruning the tree, not yet to increase fuzziness.

From the definition of fuzziness degree (equation (3.32)) one can derive the condition (3.60).

$$\sum_{k=1}^{2\cdot(f-1)} a_{jk} = I_j \cdot Gf_j \qquad (3.60)$$

In equation (3.60) I_j indicates the domain of the j-th parameter, Gf_j the fuzziness degree of the j-th parameter. In the following, the pedicel "j" will be omitted, being all the description referred to a generic dimension.

Let m_i be the number of misclassified examples for which the parameter belongs to the i-th set (i = 1,…, f). The fuzzyfication efficiency assumption suggests a condition on the way the intervals where fuzzy sets overlap should be parceled out in correspondence of the thresholds. In the particular case of Fig. 24, this condition would consist of the proportion expressed by (3.61).

$$\frac{a_1 + a_2}{a_3 + a_4} = \frac{m_1 + m_2}{m_2 + m_3} \tag{3.61}$$

In addition, the fuzzyfication efficiency assumption suggests a further conditions on the way the intervals where fuzzy sets overlap in correspondence of each threshold should be parceled out between the sub-intervals located on the left and on the right of the thresholds. In the particular case of Fig. 3.24 these conditions would consist in the proportions expressed by (3.62).

$$\frac{a_1}{a_2} = \frac{m_1}{m_2} \; ; \; \frac{a_3}{a_4} = \frac{m_2}{m_3} \tag{3.62}$$

Thus, thanks to the conditions expressed by equations (3.60) - (3.62), a set of linearly independent equations can be written to derive quantities a_k, as shown by (3.63).

$$\begin{cases} a_{2k} + a_{2k-1} = \dfrac{m_k + m_{k+1}}{2 \cdot \sum_{z=1}^{f} m_z - m_1 - m_f} \cdot I \cdot Gf \\ a_{2k} - a_{2k-1} \cdot \dfrac{m_{k+1}}{m_k} = 0 \end{cases} \tag{3.63}$$
$$for \; k = 1, ..., f - 1$$

The values of the fuzzy references obtained applying the set of equations (3.63) shall be subjected to further constraints of the type: $r_2 \geq r_1$, $r_4 \geq r_3$, $r_6 \geq r_5$.

The fuzziness degree may be set either manually or automatically, for example by means of equation (3.64),

$$Gf = \frac{M}{N} \tag{3.64}$$

where M is the total number of misclassified examples and N is the total number of examples. In case *Gf* equals 0, the values of a_k will be set all equal to 0, and (3.63) will not be used.

Thus, the proposed approach allows to derive a set of if-then rules from a crisp database containing continuous-valued attributes, selecting the most significant attributes and determining the discretization references, optimized according to the rules themselves. The obtained algorithm is then pruned to improve its generality (thus avoiding overfitting); furthermore, it is "translated" to fuzzy logic, setting the fuzzy references on the basis of the distribution of misclassified examples.
Hence, the devised algorithm, ID3n3, allows to build a fuzzy engine automatically from a training database. If a "linguistic" fuzzy engine has to be implemented, the described procedure provides a set of linguistic rules that shall be written directly as code. If a "matrix" fuzzy engine has to be implemented, the described procedure provides both the references set and the rule matrix in a completely automatic way.

At this point, given the strong analogy between decision trees and fuzzy rule sets, some concepts will be defined, to allow a deeper understanding of both objects.

- Definitions and equivalences

→ *Node*: function which takes one input and returns at least two outputs, partitioning the input training dataset with respect to a given attribute (parameter). Therefore, it partitions the input space along a given dimension; it is equivalent to a rule condition.

→ *Leaf*: function which takes one input and returns one output, associating a set of training examples to a given output category. Therefore, it associates a subset of the (discretized) input space to an output category; it is equivalent to function f_d.

→ *Connection parent-children*: each output of the parent node constitutes the input for a child node. It corresponds to the AND operator between conditions within a rule.

→ *Root node*: it is a node which has no parent node.

→ *Branch*: set containing one leaf, the root node and all the nodes that connect that leaf to the root. It corresponds to a rule.

→ *Squared tree*: tree where all parameters (attributes used in the tree) are associated to the same number of nodes. It is associated to a discretized input space where all dimensions have the same size.

→ *Proper tree*: tree which has only one root node. It corresponds to a rule set for which a parameter can be found, that has a condition in every rule.

→ *Improper tree*: tree which has more than one root node. It corresponds to a rule set where, for each parameter, at least one rule can be found which is independent from that parameter.

→ *Parameter significance*: a parameter is as much significant as close are to the root its correspondent nodes. Therefore, the most significant parameter is the one associated to the root node. In terms of rule set, the most significant parameter is the one which is used in the largest number of rules.

→ *Parameter refinement*: a parameter is as much refined as numerous are its correspondent nodes. In terms of rules set, the most refined parameter is the one which is discretized (fuzzyfied) with the largest number of sets.

Observation.
The concepts of significance and refinement are not absolute, but relative to a given tree (rules set).

Post-pruning is the last aspect of decision trees which is worth to be analyzed more deeply. In fact, the proposed approach allows several "degrees of freedom" in the tree development on the basis of a training dataset; therefore, an optimization strategy shall be drown.
Primarily, it is important to remark that the best tree is not necessarily the one that uses at each node the attribute with highest information gain. In fact, information gain maximization leads to the best attribute choice at a given step, but does not account for further tree development. Likewise, the choice of the threshold which provides the maximum information gain for a given attribute does not necessarily lead to the best tree. Secondarily, it must be remarked that the best tree is not necessarily a proper tree. In fact, if a tree is proper, its equivalent rules set is subjected to the constraint that a parameter[13] has a condition in every rule. This is an unnecessary limitation. For example, considering the first level of identification, it is reasonable to state that the most significant parameter consists of the shape factor of the Weibull function that fits the amplitude PD distribution, β. However, it is also reasonable to argue that the best rules set may include rules which are independent from β, for example "If {(Fi_{min} is Low) AND (Fi_{mean} is Low)} => Internal".

Hence, in the following a list of possible "degrees of freedom" in the tree development is presented.

a) Use a training dataset restricted to certain attributes (e.g. those associated to higher information gain at the first check).
b) Use different values of pre-pruning threshold.
c) At each iteration (node), use the parameter which corresponds to rank *h* in the maximum information gain sorting (let *h* result from a given probability distribution).
d) In the calculation of the parameter threshold at a given node, use the relative peak of

[13] The parameter which corresponds to the root node in the tree.

information gain which corresponds to rank h (let h result from a given probability distribution).

Taking advantage of these degrees of freedom, an algorithm can be set up for tree development, ID3n4, which is provided by one or more random variables. Running ID3n4 N times, a population of N trees is obtained. All of these trees were developed on the basis of the same training dataset, but, in general, their associated input space is different, being the input space determined by the selected attributes together with their thresholds. Therefore, the objects to be optimized are not homogeneous. Indeed, the fitness function defined by (3.59) allows an optimization strategy simply by applying a Monte Carlo method. In fact, one could just generate a very large population of trees, calculate the fitness function for each tree and select the one characterized by highest fitness.
This technique would allow to select, together with the best tree within the population, an optimized input space. Successively, it would be possible to run an algorithm of the type of ID3 on a discrete-valued dataset, introducing again some aleatory variables. In this way, a new population of trees would be generated, this time homogeneous with respect to the input space, which could be optimized through more refined strategies, like genetic algorithms.

Finally, a procedure to obtain an improper tree characterized by improved efficiency (and fitness) is presented in the following.
Clearly, ID3n3, as well as other algorithms described in literature [22], provide proper trees. If ID3n3 is run with a high threshold of pre-pruning, it is expectable that a relatively large number of examples are misclassified. Generally speaking, if the pre-pruning threshold is relatively low, the misclassified examples may be thought as “outliers”; on the contrary, if the pre-pruning threshold is high, one may think that the set of misclassified examples require their own rules to be identified, providing a set of less probable, but not negligible instances. In this light, one could run a ID3 like algorithm on the original dataset, with the target vector set to 1 for the misclassified examples, and 2 for the well classified ones. In this way, a set of rules is provided which separate the misclassified examples from the others. At that point, a ID3 like algorithm is run on the training dataset restricted to the misclassified examples, using the target values that the examples had in the original dataset. The later rules set, put in “and” with the former, would provide the desired set of rules specific for the misclassified examples, characterized by a different root with respect to the tree generated at first. This procedure allows to improve the efficiency of the whole algorithm, avoiding the increment of the input space (e.g. number of parameters and number of fuzzy sets). This result is interesting, because a fuzzy engine of the “matrix” type is not affected by the number of rules, since its complexity depends only on the input space.

Concept learning

The problem of inducing general functions from specific training examples is central to learning. In fact, it is relatively easy to find an algorithm which fits the training sample, but such a solution may be lacking in generality. For example, decision tree based algorithms guarantee a good efficiency with respect to the training dataset, but pose the problem of pruning to avoid overfitting. In this chapter, an AI technique is described which does not necessarily provide a solution for the training dataset, but, when successful, it guarantees for generality. In addition, this technique was developed in such a way that, in case it is not successful, it selects the training examples which should be excluded (as outliers) for the algorithm itself to converge. This technique requires discrete valued attributes, and cannot be extended to continuous valued ones. Therefore, such an algorithm could be used in series to a decision tree algorithm, to get precious information about its generality and to provide a rule set in alterative to the one associated to the tree.

General speaking, concept learning can be described as the problem of automatically inferring the general definition of some concept, given examples labeled as members or non-members of a given category. Concept learning can be formulated as the problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples. Hence, this approach involves an inductive bias of the restrictive typology, that is, the algorithm searches completely in an incomplete hypotheses space.

To describe this approach the database of table 3.6 will be used as reference. In particular, the concept of "lemon" will be considered. In fact, in this approach each output category must be considered separately. Hence, focus is made on the training dataset of table 3.13.

Tab. 3.13: *example training dataset.*

Record index	**attributes**						**Target value**	
	Color		**Taste**		**Shape**			
1	yellow	(1)	sweet	(1)	oblong	(2)	Not lemon	(0)
2	yellow	(1)	sweet	(1)	round	(1)	Not lemon	(0)
3	green	(2)	sweet	(1)	round	(1)	Not lemon	(0)
4	red	(3)	sweet	(1)	round	(1)	Not lemon	(0)
5	yellow	(1)	sour	(2)	round	(1)	lemon	(1)
6	yellow	(1)	sweet	(1)	oblong	(2)	Not lemon	(0)
7	green	(2)	sweet	(1)	round	(1)	Not lemon	(0)
8	green	(1)	sour	(2)	round	(1)	lemon	(1)
9	yellow	(1)	sour	(2)	round	(1)	lemon	(1)
10	red	(3)	sweet	(1)	round	(1)	Not lemon	(0)

Hypotheses will be formulated as if-then rules, in order to achieve a perfect synergy with the other approaches, i.e. fuzzy engines and decision trees. An example of hypothesis is the following:

"IF { (Color is green or red) AND (Taste is sour) AND (Shape is round) } THEN output is Lemon"

In the following, since every rule refers to "Lemon" category, only the rule antecedent will be reported; in addition, it will be assumed that the first condition refers to attribute Color, the second to attribute Taste and the third to attribute Shape, therefore the attribute names will be omitted; finally, attribute values will be referred to through the correspondent integers, for the sake of brevity. With these assumptions, the above reported rule can be formulated as follows:

"{ (2 or 3), (2), (1) }"

In this examples, the set of all instances, X, consists of $3 \times 2 \times 2 = 12$ instances (or elementary rules, or sub-domains). Therefore, the power set of X, H_P, consists of 2^{12} elements, that is, all possible hypotheses which could be formulated, if no restriction was made (no inductive bias).
It is noteworthy to observe that, in order to formulate all those hypotheses, the operator OR between conditions is required. In the approach chosen in this thesis, the restriction (i.e. the inductive bias) consists of the fact that operator OR is prohibited. Therefore, rules (hypotheses) can be formulated only as the intersection (conjunction) of the attribute conditions.
To provide a graphical representation of such a restriction, one could focus only on the attributes Taste and Shape. In this case, the set of all instances includes $2 \times 2 = 4$ instances, and the power set $2^4 = 16$ hypotheses.

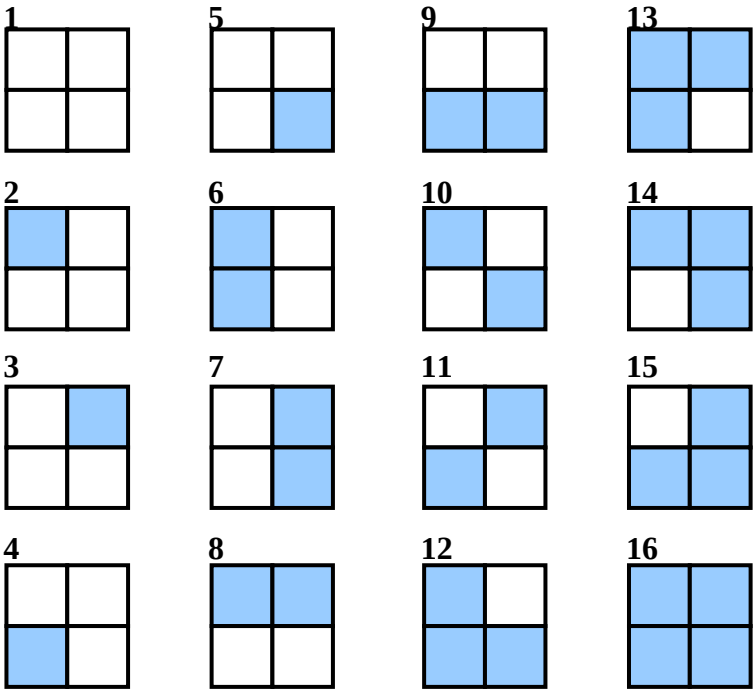


Fig. 3.25: *graphical representation of the power set (with respect of attributes Taste and Shape); each hypothesis is a matrix (2 × 2), rows ⇒ first attribute (Taste), columns ⇒ second attribute (Shape)*

Actually, hypothesis 1 (with reference to Fig. 3.25) is only theoretical, as it consists of the null set. Hypotheses 2 – 5 are the elementary rules (sub-domains), and constitute the set of all instances, *X*. Hypotheses 10 – 15 cannot be formulated without the use of OR operator, therefore they do not belong to the hypothesis space, *H*. For example, hypothesis 10 would be formulated as:
{ [(1) AND (1)] OR [(2) AND (2)] }, while hypothesis 12 would be formulated as { (1) OR (1) }. All other hypotheses satisfy the adopted restriction, therefore they constitute the hypothesis space, *H*. Clearly, hypothesis 16 represents the whole input space.
Since concept learning algorithms will search through the hypothesis space, it is essential to determine how many hypotheses are there in *H*. For this purpose, equation (3.65) has been derived for a general case with p attributes, given that the k-th attribute may assume f_k values.

$$N_H = \prod_{k=1}^{p} \left(2^{f_k} - 1 \right) \tag{3.65}$$

Applying this equation to the simple case of Fig. 25 (with two attributes, each provided with two values), the number of hypotheses in *H* results 9, namely hypotheses 2 – 9 and 16. Applying the same equation to the case of Tab. 13 (p=3, **f** = (3, 2, 2)), the number of hypotheses in *H* results 63.

The goal of a concept learning algorithm is to search through *H* to find all those hypotheses which are consistent with the training dataset. Therefore, it is essential to define precisely the concept of consistency of a hypothesis with respect to a training instance. Let *h* be a certain hypothesis, correspondent to a portion *G* of the input space and associated to the considered output category, and let *x* be an instance and *t(x)* its related target. Note that if *t(x)* coincides with the considered output category *x* is a positive example, otherwise it is a negative example.

- Definition
h <u>is consistent with</u> < *x*, *t(x)* > if and only if *h(x)* = *t(x)*.
In other words, if *x* is a positive example, its correspondent sub-domain must be included in *G*; on the contrary, if *x* is a negative example, its correspondent sub-domain must be external to *G*.
Clearly, a hypothesis is consistent with a training dataset if it is consistent with all of the instances which constitute the dataset.

One simple way to implement a concept learning algorithm would be to list all hypotheses in *H* and check them one by one. Once inconsistent hypotheses are rejected, the remaining subset of *H* constitutes the *version space*, H_V, i.e. the set of all hypotheses consistent with the training dataset. Although simple, this approach is not generally applicable, because *H* may be very complicated and include a huge number of hypotheses. Therefore, a more powerful approach will be followed, based on a general-to-specific ordering of the hypotheses in *H*, [3].

As cleared in the chapter dedicated to fuzzy logic, every hypothesis can be seen as a subset of the input space, which consists of the set of all instances, *X* (mapped input space). Hence, with respect to a given output category, c, the goal is to represent the concept of that category as a hypothesis (or a set of hypotheses). In this light, positive examples specify the characteristics that the concept may have, thus providing a progressive generalization of the hypothesis which represents the concept. On the other side, the negative examples specify combinations of characteristic which are not pertinent to the searched concept, thus providing a progressive specialization of the hypothesis which represents the concept. Therefore, one could manage to obtain a specific boundary, S, and a generic boundary, G, for the concept.
Thus, to compare hypotheses, it would be quite effective to determine whether a hypothesis is more general (or, vice versa, more specific) with respect to another.

- Definition
Given two hypotheses, h_1 and h_2, defined in a certain input space, X, let A_1 and A_2 be the subsets of X correspondent to h_1 and h_2, respectively.
$h_1 <_s h_2$ (h_1 is more specific than h_2) if $A_1 \subset A_2$;
$h_1 >_g h_2$ (h_1 is more general than h_2) if $A_1 \supset A_2$;
$h_1 \equiv h_2$ (h_1 is the same as h_2) if $A_1 = A_2$;
otherwise, h_1 and h_2 are not directly comparable.

On the basis of this definition, a partial ordering relation is established in the hypotheses space. Conventionally, the most specific hypothesis is considered the null set, while the most general hypothesis consists of the whole input space. It is noteworthy to observe that these two boundaries include the whole power set of hypotheses, since any hypothesis is more general than the null set and more specific than the whole input space, at the same time. In this light, the concept learning problem can be thought as the search for the boundaries, specific boundary S and general boundary G, which delimit (and summarize) the version space [21, 22]. More rigorous definitions of S and G are provided in equations (3.66) and (3.67).

- Definitions of S, G, with respect to hypothesis space *H* and training dataset $X_{database}$
The specific boundary, S, is the set of maximally specific members of *H* consistent with $X_{database}$.

$$S \equiv \left\{ s \in H \mid consistent\left(s, X_{database}\right) \wedge \left(\neg \exists s' \in H\right)\left[\left(s >_g s'\right) \wedge consistent\left(s', X_{database}\right)\right]\right\} \quad (3.66)$$

The general boundary, G, is the set of maximally general members of *H* consistent with $X_{database}$.

$$G \equiv \left\{ g \in H \mid consistent\left(g, X_{database}\right) \wedge \left(\neg \exists g' \in H\right)\left[\left(g' >_g g\right) \wedge consistent\left(g', X_{database}\right)\right]\right\} \quad (3.67)$$

In the following, the basic structure of a concept learning algorithm called “Candidate-Elimination” will be described [3].

Tab. 3.14: *schematic description of algorithm Candidate-Elimination*

- Initialize G to the most general hypothesis in H.
- Initialize S to the most specific hypothesis in H.
- FOR each training example, x
  - IF x is a positive example
    - Remove from G any hypothesis inconsistent with x
    - FOR each hypothesis s in S which is not consistent with x
      - Remove s from S.
      - Add to S all minimal generalizations h of s such that h is consistent with x and some member of G is more general than another hypothesis in S.
      - Remove from S any hypothesis which is more general than another hypothesis in S.
    - END
  - ELSEIF x is a negative example
    - Remove from S any hypothesis inconsistent with x
    - FOR each hypothesis g in G which is not consistent with x
      - Remove g from G
      - Add to G all minimal specialization h of g such that h is consistent with x and some member of S is more specific than another hypothesis in G.
      - Remove from G any hypothesis which is more specific than another hypothesis in G.
    - END
  - END
- END

To apply Candidate-Elimination, one must clarify how to determine minimal generalizations and minimal specializations of a hypothesis in order to make the hypothesis itself consistent with a given example.

Primarily, hypotheses will be expressed in a convenient form, from the computational point of view. In particular, any hypothesis (rule) antecedent can be expressed as a cluster of p Boolean vectors, such that the k-th vector consists of f_k elements. In the example described earlier, the number of attributes was p = 3 and the number of values assumed by each attribute was **f** = (3, 2, 2). With these premises, the rule antecedent

{ (Color is green or red) AND (Taste is sour) AND (Shape is round) }

which was expressed in a compact form as

“{ (2 or 3), (2), (1) }”

will be expressed in the following as

"{ (0, 1, 1), (0, 1), (1, 0) }"

- Definition. *Minimal generalization, h^g, of a hypothesis h to an instance x*

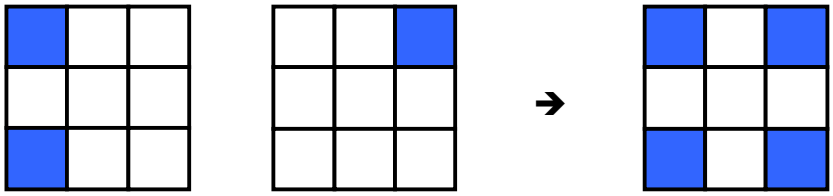
$$h^g \equiv \left\{ h' \in H \mid h'(k) = \left(h(k) \vee x \right),\ k = 1, \ldots p \right\} \tag{3.68}$$

- Definition. *Minimal specializations, $\{h^s{}_k\}$, of a hypothesis h to an instance x*

$$h^s{}_k \equiv \left\{ h' \in H \mid h'(k) = \left(h(k) \wedge \left(h(k)\ \ xor\ x \right) \right) \right\}_k ,\ k = 1, \ldots p \tag{3.69}$$

In order to provide graphical exemplifications of the minimal generalization and specialization processes, a simple case will be considered, supposing p = 2 and **f** = (3, 3), that is, two attributes, each one characterized by three possible values.

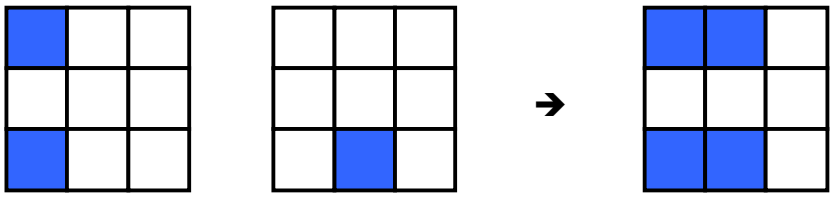
Examples of minimal generalization of a hypothesis *h* with respect to a positive example *x*.
Let *h* = { (1, 0, 1), (1, 0, 0) }
x = { (1, 0, 0), (0, 0, 1) } ➔ h^g = { (1, 0, 1), (1, 0, 1) }



h *(positive) x* h^g

Fig. 3.26: *graphical representation of an example of minimal generalization.*

x = { (0, 0, 1), (0, 1, 0) } ➔ h^g = { (1, 0, 1), (1, 1, 0) }

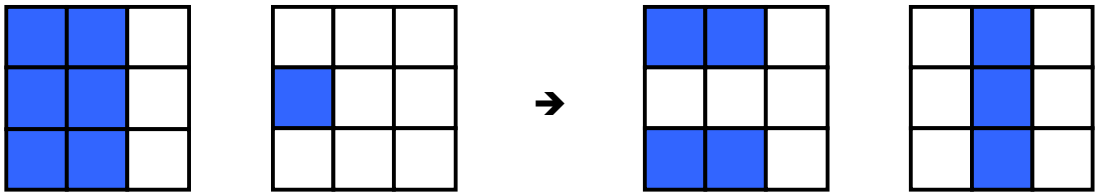


h *(positive) x* h^g

Fig. 3.27: *graphical representation of an example of minimal generalization.*

Examples of minimal specialization of a hypothesis *h* with respect to a negative example *x*.
Let *h* = { (1, 1, 1), (1, 1, 0) }

x = { (0, 1, 0), (1, 0, 0) } ➔ h^s_1 = { (1, 0, 1), (1, 1, 0) } ; h^s_2 = { (1, 1, 1), (0, 1, 0) }



h *(negative) x* h^s_1 h^s_2

Fig. 3.28: *graphical representation of an example of minimal specialization.*

It is noteworthy to observe that the minimal generalization process produces a unique hypothesis, because the instance *x*, coming from a positive example, requires to be fully satisfied by the new hypothesis. On the contrary, the minimal specialization process may produce from one to p equally specific hypotheses, because the instance *x* comes from a negative example. To examine a practical case, the situation of Fig. 3.28 can be considered. The instance *x* requires that it must not be verified that the first attribute assumes the second value and, at the same time, the second attribute assumes the first value. Therefore, two distinct situations (hypotheses) can be verified: the first attribute does not assume the second value and the second attribute assumes the first value, h^s_1, or the first attribute assumes the second value and the second parameter does not assume the first value, h^s_2. Clearly, the hypothesis for which the first attribute does not assume the second value and, at the same time, the second attribute does not assume the first value (Fig. 3.30) is consistent with *x*, too, but it is more specific than h^s_1, hence it is not a minimal specialization and would be thrown away the next step.

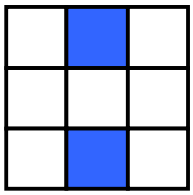
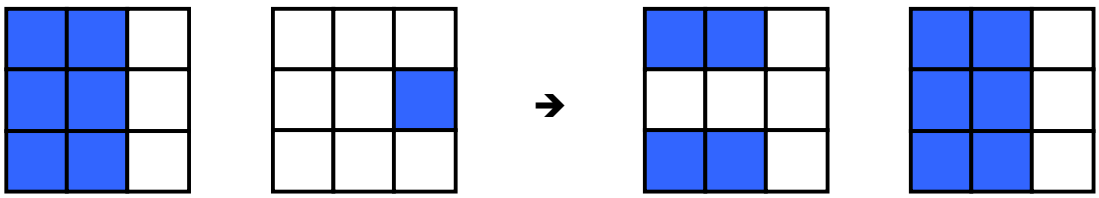


Fig. 29: *graphical representation of a non minimal generalization of the hypothesis of Fig. 28.*

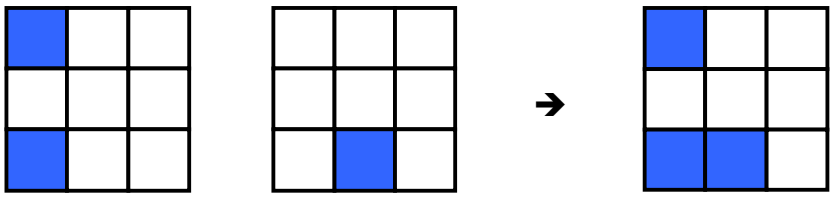
Let h = { (1, 1, 1), (1, 1, 0) }
x = { (0, 1, 0), (0, 0, 1) } ➔ h^s_1 = { (1, 0, 1), (1, 1, 0) } ; h^s_2 = { (1, 1, 1), (1, 1, 0) }



h *x* h^s_1 (h^s_2)

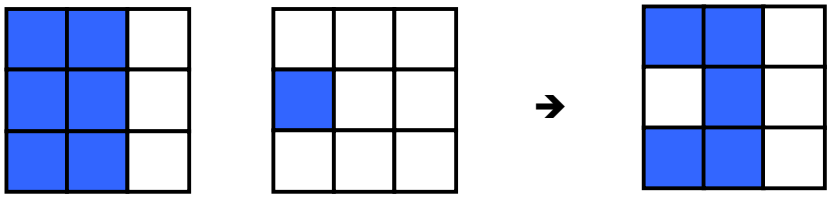
Fig. 30: *graphical representation of an example of minimal specialization.*

Finally, the examples of Figs. 3.27 and 3.28 will be reported again in figures 3.31 and 3.32, showing the result of minimal generalization and specialization process, respectively, in case there was no inductive bias, that is, no restriction in the formulation of hypotheses.



h *(positive) x* h^g

Fig. 3.31: *graphical representation of an example of minimal generalization in the absence of inductive bias.*



h *(negative) x* h^s

Fig. 3.32: *graphical representation of an example of minimal specialization in the absence of inductive bias.*

An example of the application of Candidate-Elimination algorithm on the case of table 13 will be reported in the following. For simplicity, the algorithm will be applied on two positive examples and, separately, on two negative examples. To begin, the specific boundary S can be initialized to the null set, the general boundary G to the entire input space.

➔ $S_0 = \{ \varnothing \}$, $G_0 = \{ \{ (1, 1, 1), (1, 1), (1, 1) \} \}$ (initialization)

The first positive example (fifth record in the database of table 3) consists of the instance

$x_1 = \{ (1, 0, 0), (0, 1), (1, 0) \}$

Clearly, a positive example has no effect on G, but induces a generalization of S. Therefore, the specific and general boundaries, at the first iteration, assume the following form:

$S_1 = \{ \{ (1, 0, 0), (0, 1), (1, 0) \} \}$

$G_1 = \{ \{ (1, 1, 1), (1, 1), (1, 1) \} \}$

The second positive example (eight record in the database of table 3) consists of the instance

$x_2 = \{ (0, 1, 0), (0, 1), (1, 0) \}$

The specific and general boundaries, at the second iteration, assume the following form:
S_2 = { { (1, 1, 0), (0, 1), (1, 0) } }
G_2 = { { (1, 1, 1), (1, 1), (1, 1) } }

If considered at this point, the specific boundary provides already an indication about the searched concept. In fact, S consists of a single hypothesis, which can be expressed in the form of linguistic rule as follows:

IF { (Color is Yellow or Green) AND (Taste is Sour) AND (Shape is Round) } THEN Lemon

Starting again from the initialization, S_0 and G_0, the effect of two negative examples is shown in the following.
The first negative example (first record in the database of table 3) consists of the instance
x_1 = { (1, 0, 0), (1, 0), (0, 1) }
Clearly, a negative example has no effect on S, but induces a specialization of G. Therefore, the specific and general boundaries, at the first iteration, assume the following form:
S_1 = { ∅ }
G_1 = { { (0, 1, 1), (1, 1), (1, 1) } ; { (1, 1, 1), (0, 1), (1, 1) } ; { (1, 1, 1), (1, 1), (1, 0) } }

The second negative example (second record in the database of table 3) consists of the instance
x_2 = { (1, 0, 0), (1, 0), (1, 0) }
The specific and general boundaries, at the second iteration, assume the following form:
S_2 = { ∅ }
G_2 = { { (0, 1, 1), (0, 1), (1, 1) } ; { (0, 1, 1), (1, 1), (0, 1) } ; { (1, 1, 1), (0, 1), (0, 1) };
{ (0, 1, 1), (1, 1), (1, 0) } ; { (1, 1, 1), (0, 1), (1, 0) } }

Even if the algorithm is stopped at this point, the general boundary does not provide yet an interpretable indication about the searched concept, because G consists of a large number of hypotheses. In practice, G specifies what a lemon cannot be, if one considers the subset of the input space external to G.
It has to be observed that Candidate-Elimination algorithm does not necessarily get to a complete or even partial convergence. A complete convergence would be reached if S and G eventually get to coincide to a single hypothesis. A partial convergence would be obtained if, once all records have been considered, S and G are constituted by more than one hypothesis, but S is still more specific than G. A partial convergence would mean that some kind of disjunction among hypotheses is required to express the searched concept. In table 3.15 a basic scheme is presented, showing a possible extension of Candidate-Elimination algorithm in such a way that it would return the indexes of the records that should be excluded in order to obtain (at least) a partial convergence. These records will be termed outliers.

Tab. 3.15: *schematic description of algorithm Candidate-Elimination extended to outlier search (CEn1)*

- Initialize G and S
- FOR each training example, x
  - IF x is a positive example
    - Remove from G any hypothesis inconsistent with x
    - FOR each hypothesis s in S which is not consistent with x, S' ← S
      - Remove s from S'.
      - Add to S' all minimal generalizations h of s such that h is consistent with x and some member of G is more general than another hypothesis in S'.
      - Remove from S' any hypothesis which is more general than another hypothesis in S'.
      - IF $S' = \{\emptyset\}$ or $\exists s \in S, \exists g \in G \mid s \geq_g g$
        - Add x to the list of outliers
      - ELSE S ← S'
    - END
  - ELSEIF x is a negative example
    - Remove from S any hypothesis inconsistent with x
    - FOR each hypothesis g in G which is not consistent with x, G' ← G
      - Remove g from G
      - Add to G all minimal specialization h of g such that h is consistent with x and some member of S is more specific than another hypothesis in G.
      - Remove from G any hypothesis which is more specific than another hypothesis in G.
      - IF $S' = \{\emptyset\}$ or $\exists s \in S, \exists g \in G \mid s \geq_g g$
        - Add x to the list of outliers
      - ELSE G ← G'
    - END
  - END
- END

The excluded records could be analyzed by the operator, if he has the possibility of data interpretation. If these records can be actually considered outliers, one could accept the learned concept and traduce it into a rule set. On the contrary, if the excluded records are considered reliable and fundamental for the learning, one might decide to search for more significant attributes or to adapt the output categories (searched concepts) to the available data.

Thus, a feedback from machine to human expert is obtained, together with a tool to guide the operator himself in the interpretation of the AI algorithms provided by the machine, in the attempt to derive general theories.

Actually, the proposed Candidate-Elimination algorithm (CEn1) was further modified, developing an approach to concept learning which is more robust and suited to the identification process. This approach is based on the consideration that for each output category, a complementary category can be devised (e.g. category "lemon" is complementary to category "not lemon"), in such a way that the negative examples of a given category can be seen as the positive examples of its complementary category. In this light, the minimal generalization process is performed twice (for the category in exam and its complementary category), while the minimal specialization process is avoided.
To proceed, the result of the application of CEn1 (extended version of Candidate-Elimination algorithm) to the case of table 8 will be presented. Since in this simple case there are only two output categories (compact and flat shaped defect), the result of CEn1 will be reported with respect to the positive records only, according to the new approach. In fact, the two categories are characterized by complementarity, that is, "flat" is the same as "not compact" and vice versa.

→ Category "flat shaped defect"
S_{flat} = { { (1, 0, 1), (1, 1, 1), (1, 1, 0)} }
Linguistically,
IF { (Par#1 is Low or High) AND (Par#3 is Low or Medium) } THEN defect is flat
Outliers: records 30, 34

→ Category "compact shaped defect"
$S_{compact}$ = { { (0, 1, 1), (0, 1, 1), (1, 1, 1)} }
Linguistically,
IF { (Par#1 is Medium or High) AND (Par#2 is Medium or High) } THEN defect is compact
Outliers: records 13, 18

The hypotheses (rules) determined as specific boundaries, S_{flat} and $S_{compact}$, represent subsets of the input space which satisfy two conditions: they are in agreement with the specifications of the inductive bias and they are consistent with all of the positive records (except for outliers), but not with all of the negative examples. Therefore, if the rules above were implemented (e.g. using a matrix fuzzy engine[14]), one would expect that a flat defect would have a high membership to the "flat" category, but there could be a non-zero membership to the "compact" category, too. In fact, the intersection of S_{flat} and $S_{compact}$ is not empty. Therefore, those rules represent only necessary conditions for the examined defect to belong to a specific category. Thus, the subsets of the input space which correspond to S_{flat} and $S_{compact}$ shall be called $(C_{flat})_{necessary}$ and $(C_{compact})_{necessary}$, respectively. It is useful to derive subsets of the input space which provide sufficient conditions for

[14] The fuzzyfication of the crisp intervals in the case of CEn1 will be dealt with in the next paragraph.

the examined defect to belong to a specific category, in this case, $(C_{flat})_{sufficient}$ and $(C_{compact})_{sufficient}$. The hypotheses (rules) which correspond to these conditions would satisfy two conditions: they are in agreement with the specifications of the inductive bias and they are consistent with all of the negative records (except for outliers), but not with all of the positive examples. Hence, if these rules were implemented one would expect that a defect might not be recognized, but, if a defect is addressed to a specific category, there would be no ambiguity.
To obtain such sufficient conditions equation (3.70) can be used.

$$\left(C_{categoryA}\right)_{sufficient} = \left(C_{categoryA}\right)_{necessary} \wedge \left(\neg\left(C_{category\neg A}\right)_{necessary}\right) \tag{3.70}$$

Clearly, equation (3.70) shall be split in p equations, one for each attribute. In addition, from this equation a further condition follows, expressed by (3.71).

$$\left(C_{categoryA}\right)_{sufficient} \subseteq \left(C_{categoryA}\right)_{necessary} \tag{3.71}$$

The resulting sufficient conditions are reported in the following.

→ Category “flat shaped defect”
S_{flat} = { { (1, 0, 0), (1, 0, 0), (0, 0, 0) } }
Linguistically,
IF { (Par#1 is Low) AND (Par#2 is Low) } THEN defect is flat

→ Category “compact shaped defect”
$S_{compact}$ = { { (0, 1, 0), (0, 0, 0), (0, 0, 1) } }
Linguistically,
IF { (Par#1 is Medium) AND (Par#3 is High) } THEN defect is compact

In can be observed that rules do not necessarily involve all attributes. The fact that a rule is independent from a given attribute means that either all or none of its values is present in the condition from which the rule is derived.
The described approach partially draws away from the original version of Candidate-Elimination, because it comes to the definition of necessary and sufficient conditions and actually abandons the version space. This approach is probably less exhaustive with respect to concept learning, but is very practical and useful. In fact, in this approach only the minimal generalization process is required, which is more simple and robust with respect to the specialization one, and provides the operator with an immediate evaluation of the generality of the found solution. In fact, the larger is the gap between the necessary and sufficient condition, the weaker is the effectiveness of the attributes in representing the concept (with respect to a given set of examples).

To explain better the meaning of this approach, it is useful to represent the searched concept, the sufficient condition and the necessary condition as sub-sets of the input space. Of course, the whole approach, as well as the traditional Candidate-Elimination approach, has as pre requisite that these concepts can be expressed by means of the some hypotheses contained in the power set. If this condition was not verified, the inductive bias, thus the restriction on adopted in the hypotheses formulation, would result inadequate. Figure 3.33 provides a graphical representation of the input space, the searched concept (sub-set C), the sufficient condition (sub-set A), and the necessary condition (sub-set B).

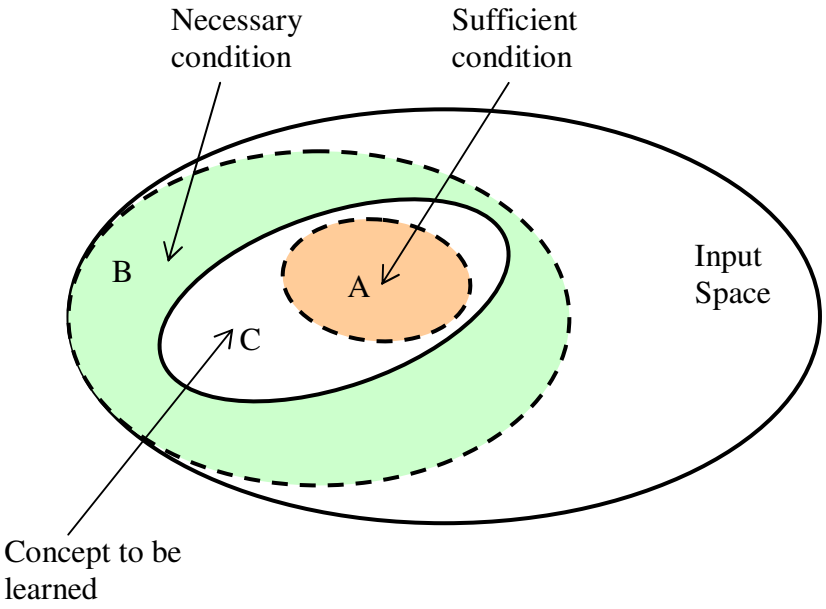


Fig. 3.33: *schematic representation of the necessary and sufficient conditions.*

As shown in Fig. 3.33, being C the unknown subset of the input space associated to the output category under consideration, it results that if an instance x is included in A, it belongs to that category ($A \subseteq C$, therefore $C_{sufficient} \Rightarrow$ x belongs to that category). On the other side, if an instance x is not included in B, it cannot belong to that category ($B \supseteq C$, therefore $\neg C_{necessary} \Rightarrow$ x does not belong to that category).

To clarify better the logic base of this approach, let Q be a generic concept, correspondent to one of the target values in a give training database. Hence, there will be a set of positive examples and a set of negative examples with respect to Q. Let C1 and C2 be the hypotheses obtained through a minimal generalization process to be consistent with the positive and the negative examples, respectively.

If either one of the following conditions is verified

- the training database is, by itself, representative of the whole database (population)
- the adopted inductive bias is actually able to generalize beyond the training examples

it can be assumed that C1 represents a necessary condition for Q (while C2 represents a necessary condition for ¬Q). This assumption is equivalent to affirm that if an object belongs to concept Q, its characteristics must be proper of at least some of the positive examples or be consistent with them

with respect to the given generalization criterium. Therefore, it results that

$Q \Rightarrow C1$ and that $\neg Q \Rightarrow C2$ (definition of necessary condition)

If $\neg Q \Rightarrow C2$ follows that $\neg C2 \Rightarrow Q$

Therefore, ¬C2, by itself, represents a sufficient condition for Q. Indeed, C2, being consistent with all negative examples, specifies the characteristics that an object must have to not belong to Q; therefore, if an object does not have any of those characteristic, it should belong to Q. All this is true if the two assumptions above are verified. Since there will be a certain degree of approximation, it might be not verified that C1 is strictly more general than ¬C2; therefore, it was found more safe to define the sufficient condition as the intersection of ¬C2 and C1, according to (3.70).

Finally, it can be noted how the necessary and sufficient criteria fit the physical reasoning which is on the basis of the attribute definition and of the identification process itself.

Instance based learning and synthesis of different approaches

As anticipated in the introduction paragraph of this section, the proposed approaches to machine teaching and learning will be combined, in the attempt to fit specific identification tasks and to improve the overall performance.

In the identification strategy, the definition of a new task consists essentially of the definition of a set of output categories. Assuming that the logical-mathematical approach introduced in this work is adopted (e.g. setting up a matrix fuzzy engine), the construction of the automatic identification procedure (for a specific task) consists of two basic steps.

- Selection of the input parameters, among all quantities (attributes) available[15].
- Training of the identification algorithm, that is, determine the matrix defining the rule set.

In case the operator (human expert) performs both of these steps on the basis of his own knowledge, the problem would be solved as a "machine teaching" case, and the resulting identifier would be completely supervised. This case was described in detail in the previous paragraphs. In this paragraph, focus is made on the following situations.

- The operator "teaches" the machine, but his knowledge is not enough to cover all possible instances, i.e., all configurations of input parameters.
- The operator cannot provide any supervision, hence both parameters selection and algorithm training must be performed automatically.

[15] These quantities may be provided by previous identification tasks (i.e. their outputs) or by statistical processing of the acquired data.

- The operator provides a partial supervision, but some aspects of the identifier construction must be performed automatically.

The fact that the matrix is only partially determined, because it does not cover all possible instances, means that the function f_d is defined only on a subset of the mapped input space. In other words, there are some elementary sub-domains which are not associated to the output space. In this case, the goal is to find an AI technique which allows the machine to identify unknown instances automatically on the basis of known ones. In this way, artificial intelligence would be applied only when the operator knowledge is missing. The AI techniques which could be suited for this purpose belong to a typology that is generally called "instance based learning algorithms" or "lazy algorithms", [3]. In the following, one basic technique is described, called *k-nearest neighbor*, and adapted to the case under study.

The k-nearest neighbor algorithm does not really search through the hypotheses space; it considers the query instance (input vector) and compares it to the known instances in terms of Euclideian distances. Given two instances x_A and x_B, their distance d can be defined [23] by equation (3.72),

$$d\left(x_A, x_B\right) = \sqrt{\sum_{k=1}^{p}\left(x_A(k) - x_B(k)\right)^2} \tag{3.72}$$

where p is the number of input parameters (i.e. the number of dimensions of the input space).
Clearly, this definition of distance is given for continuous-valued parameters, but it can be extended easily to the discrete-valued case. For example, if a given parameter is discretized in five sets, the five values which can be assumed by the parameter, Low, Low-medium, Medium, Medium-High, High, will be associated to integers 1, 2, 3, 4, 5, respectively.
The k-nearest neighbor algorithm considers the k known instances which have smallest distances from the query instance; then, it averages the outputs associated to these instances. Usually, such distances are weighted by means of a non-linear function, which is called *Kernel function*.
The inductive bias proper of this AI approach corresponds to the assumption that the classification of an instance will be most similar to the classification of other instances that are nearby in Euclidean distance.
This technique is very simple and effective, al long as the number of known instances (prior knowledge) is relatively large with respect to the size of the input space. The weak point of this method consists in the fact that no indication is provided to discriminate among the input parameters, which are supposed equally significant. This is the reason why much effort is usually spent in order to tune the kernel function.

In the following, an example case will be presented, with two input parameters (p = 2), each one

fuzzyfied in five sets (**f** = (5, 5)) and with two output categories, "blue" and "red". Figure 3.34 shows the mapped input space, where two regions are evidenced with colors blue and red, indicating how the sub-domains are associated to the output space by the prior knowledge.

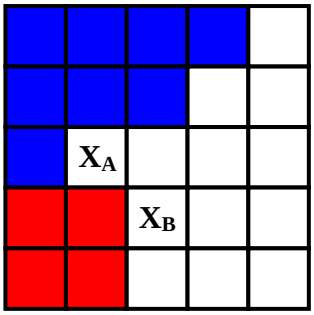


Fig. 3.34: *graphical representation of the mapped input space (two parameters, five sets each) and of prior knowledge (association of sub-domains to output categories "blue" and "red").*

In Fig. 34 x_A and x_B represent two query instances; white color characterizes sub-domains which are not associated to the output space, thus constituting unknown instances.

In the present work, k-nearest neighbor algorithm will be modified, in such a way that it is consistent with the proposed approach. First of all, a new definition of distance will be adopted, (3.73), which is supposed to be applied only on discrete-valued parameters (that is the case under study).

$$d\left(x_A, x_B\right) = \sum_{k=1}^{p} \left| x_A\left(k\right) - x_B\left(k\right) \right| \tag{3.73}$$

On the basis of this distance, a new definition is introduced: neighborhood of order k.

- Definition. *Neighborhood of order k (of a given instance)*
Given a query instance x_q, the neighborhood of order k is the set of instances (elementary sub-domains) which have a distance from x_q equal to k, where k is a natural number.

Figure 3.35 shows neighborhoods of order 1 and 2 of the instance *x* (par. 1 Medium, par. 2 Medium).

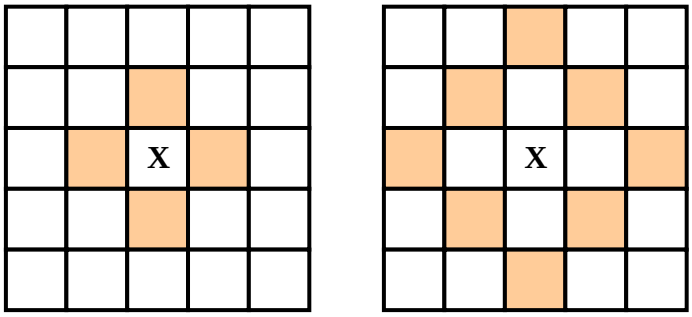


Fig. 3.35: *graphical representation of the neighborhoods of order 1 (3.35A) and 2 (3.35B) of instance x.*

For a given instance x, two quantities, n_k and z_{ki}, can be introduced:
→ n_k: number of instances in the neighborhood of order k of x.
→ z_{ki}: number of instances in the neighborhood of order k of x, which are associated to the i-th output category.

Example: in the case of Fig. 35, $n_1 = 4$, $n_2 = 8$; $z_{1\text{-blue}} = 1$, $z_{2\text{-blue}} = 3$; $z_{1\text{-red}} = 0$, $z_{2\text{-red}} = 1$.

On these basis, a new weight is defined by (3.74), which is function of the query instance and of the number of order of neighborhood.

- Definition. *Weight function $w_k(x)$*

$$w_1 = 1$$
$$w_{k+1} = w_k \cdot \frac{n_k - \sum_{i=1}^{N_{out}} z_{ki}}{n_k} \tag{3.74}$$

Therefore, the goal is to derive the output vector for an unknown instance x (i.e. $y = f_d(x)$) on the basis of the prior knowledge. For this purpose, equation (3.75) can be used, which provides the membership to the i-th output category, taking into account N orders of neighborhood.

$$y_i = \sum_{k=1}^{N} w_k \cdot \frac{z_{ki}}{n_k} \tag{3.75}$$

Example

The described approach to instance-based learning will be applied to the case of Fig. 3.34.
Figure 3.36 shows neighborhoods of order 1 and 2 of the instances x_A (par. 1 Medium, par. 2 Low-Medium) and x_B (par. 1 Medium-High, par. 2 Medium).

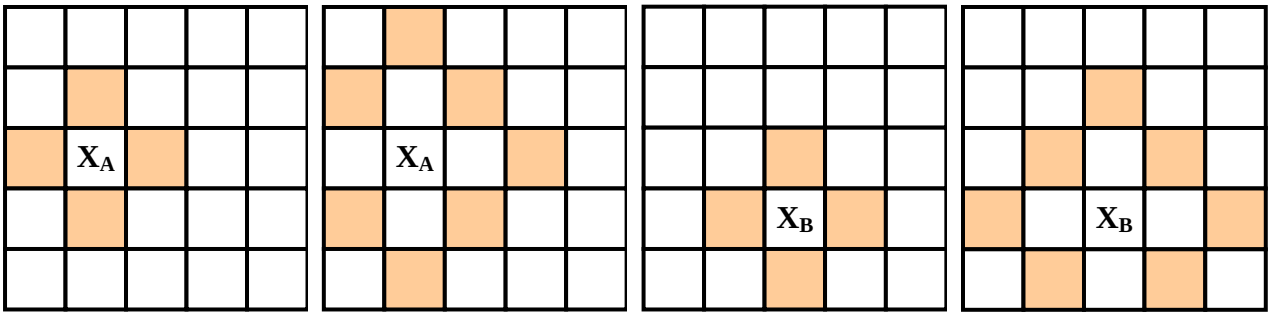


Fig. 3.36: *graphical representation of the neighborhoods of order 1 and 2 of instance x_A (3.36A and 3.36B, respectively) and of instance x_B (36C and 36D, respectively) .*

→ Instance x_A

$n_1 = 4$, $n_2 = 7$; $z_{1\text{-blue}} = 2$, $z_{2\text{-blue}} = 3$; $z_{1\text{-red}} = 1$, $z_{2\text{-red}} = 2$

The application of equation (3.75) for "blue" category, considering two orders of neighborhood, gives the following result:

$$y_{blue} = 1 \cdot \frac{2}{4} + \frac{1}{4} \cdot \frac{7}{7} = 0.61$$

For the "red" category:

$$y_{red} = 1 \cdot \frac{1}{4} + \frac{1}{4} \cdot \frac{2}{7} = 0.32$$

→ Instance x_B

$n_1 = 4$, $n_2 = 7$; $z_{1\text{-blue}} = 0$, $z_{2\text{-blue}} = 1$; $z_{1\text{-red}} = 1$, $z_{2\text{-red}} = 2$

The application of equation (3.75) for "blue" category, considering two orders of neighborhood, gives the following result:

$$y_{blue} = 1 \cdot 0 + \frac{3}{4} \cdot \frac{1}{7} = 0.11$$

For the "red" category:

$$y_{red} = 1 \cdot \frac{1}{4} + \frac{3}{4} \cdot \frac{2}{7} = 0.46$$

Important remarks

The elements of the output vector are not normalized, in fact, their sum (i.e. the output likelihood) may be smaller than unity. The reason of this choice derives from the consideration that this approach is not quite aimed at machine learning, instead, it is a machine learning technique which has the purpose to compensate a lack of knowledge within a machine teaching problem. In this light, it must be underlined that prior knowledge (on the basis of which the AI algorithm generalizes to identify unknown instances) is provided by the human expert. Therefore, it seems right to assign a lower likelihood to the identification of those instances which are unknown with respect to the expert knowledge. In equation (3.74) the weight function is defined in such a way that the likelihood equals the unity only if all instances are known in one of the N considered

neighborhoods. Hence, the purpose of the weight function in equation (3.75) is to meet this condition about likelihood, rather than to provide a non-linear transformation of the input space (like a usual Kernel function would). Indeed, such a non-linearity is already taken into account by the fuzzyfication process. Furthermore, with respect of the weight function, it can be noted that if the neighborhood of a given order is completely known, no further neighborhood is considered (its weight would be zero). Finally, it can be observed that this algorithm can be used fixing a reference for the desired likelihood, thus increasing progressively the order of neighborhood, N, to meet that condition. This would be a way to optimize the computation time with respect to the likelihood of the output.

The instance based learning algorithm described above was called "IBLn1", and will be summarized in table 3.16. It requires as inputs a query instance, x_q, a rule matrix, (obtained by any of the machine-teaching or machine-learning techniques described above) and a set of reference values to discretize (or fuzzify the query instance).

Tab. 3.16: *schematic description of algorithm IBLn1*

- Associate to each sub-domain a distance with respect to x_q.
- Initialize: n = 1 (neighborhood order), L = 0 (likelihood of $y_q = f_{ID}(x_q)$), flag = TRUE, initialize y_q to a vector of N_{out} elements equal to 0 (N_{out} number of output categories)
- WHILE flag
  - Find sub-domains which have distance n from x_q (n-th order neighborhood)
  - Calculate w_n (weight factor for the n-th order neighborhood), (3.74)
  - Evaluate y_{q-n} on the basis of (3.75)
  - $y_q \leftarrow y_q + y_{q-n}$
  - $L = \sum_{k=1}^{N_{out}} y_q(k)$
  - IF $n > N_{threshold}$ OR $L \geq L_{threshold}$
    - flag = FALSE
  - ELSE
    - $n \leftarrow n + 1$
  - END
- END

It is very important to remark that IBLn1 was described, in table 3.16, for the crisp case. When the discretization of the input space is crisp, any query instance, x_q, is associated to only one sub-domain, with membership strictly equal to 1. If the discretization of the input space is fuzzy (fuzzyfication), x_q is associated, in general, to a set of N sub-domains, with a correspondent set of N membership values (such that their sum equals 1). In this (fuzzy) case, IBLn1 will be run N times,

by means of the total membership theorem, (3.29).

It should be now clear how the approach to fuzzy logic developed allows to extend with extreme simplicity to fuzzy logic any AI algorithm which is based on a discretized input space.

Furthermore, it is worth to observe that this instance-based-learning approach can be used to train an identifier directly from a training database (e.g. the one reported in table 3.8), given a set of either crisp or fuzzy references for the discretization (or fuzzyfication) process. In fact, one could just derive a rule matrix (sometimes this rule matrix is called Fuzzy-Associative-Memory Bank, [17]) by associating the sub-domains correspondent to the training examples to the output space according to the target values correspondent to those examples. In this way, it is expectable that only a small part of the input space is associated to the output space, i.e. the rule matrix is sparse, because the training examples (prior knowledge) cover only a small part of the input space (all possible situations). Therefore, IBLn1 algorithm can be run iteratively on each sub-domain which is unknown (not associated to the output space), thus filling up progressively the rule matrix (knowledge extension on the basis of distances). Knowledge extension can be achieved at different speeds, depending on the maximum order of neighbor which is set for the IBLn1 algorithm.

The result of such a procedure applied to the training database of table 3.8 will be shown in the following. Furthermore, a comparison will be made between the rule matrix obtained with this instance based learning procedure and that returned by decision tree of table 11E. The comparison will be made in both crisp and fuzzy case. The crisp and fuzzy references derived automatically by the decision tree of table 11E were used as input for IBLn1, to make the comparison more homogeneous.

Tab. 3.17: *Comparison-synergy between decision trees and instance based learning – crisp case (distinction is made between speed 1, s1, and 2, s2, in the determination of extended VR).*

	VR crisp			VR crisp extended s1			VR crisp extended s2			VR tree	
Sub-domain	'Compact' category	'Flat' category		'Compact' category	'Flat' category		'Compact' category	'Flat' category		'Compact' category	'Flat' category
# 1	0.00	1.00		0.00	1.00		0.00	1.00		0.00	1.00
# 2	0.00	0.00		0.00	0.25		0.32	0.36		1.00	0.00
# 3	0.00	0.00		0.00	0.08		0.24	0.25		1.00	0.00
# 4	0.00	1.00		0.00	1.00		0.00	1.00		0.00	1.00
# 5	0.00	0.00		0.40	0.25		0.54	0.35		1.00	0.00
# 6	0.00	0.00		0.35	0.08		0.67	0.26		1.00	0.00
# 7	0.00	0.00		0.33	0.67		0.33	0.67		0.00	1.00
# 8	1.00	0.00		1.00	0.00		1.00	0.00		1.00	0.00
# 9	0.00	0.00		0.62	0.19		0.74	0.26		0.00	1.00
# 10	1.00	0.00		1.00	0.00		1.00	0.00		1.00	0.00
# 11	0.00	0.00		0.40	0.05		0.63	0.23		1.00	0.00
# 12	0.00	0.00		0.35	0.03		0.73	0.19		1.00	0.00
# 13	0.00	1.00		0.00	1.00		0.00	1.00		0.00	1.00
# 14	1.00	0.00		1.00	0.00		1.00	0.00		1.00	0.00
# 15	1.00	0.00		1.00	0.00		1.00	0.00		1.00	0.00
# 16	0.00	1.00		0.00	1.00		0.00	1.00		0.00	1.00

# 17	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
# 18	0.50	0.50	0.50	0.50	0.50	0.50	0.00	1.00
# 19	0.00	0.00	0.33	0.00	0.46	0.30	1.00	0.00
# 20	0.00	0.00	0.31	0.14	0.60	0.29	1.00	0.00
# 21	0.00	0.00	0.55	0.06	0.82	0.17	1.00	0.00
# 22	0.00	0.00	0.21	0.38	0.40	0.55	0.00	1.00
# 23	0.50	0.50	0.50	0.50	0.50	0.50	1.00	0.00
# 24	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
# 25	0.00	0.00	0.07	0.46	0.32	0.66	0.00	1.00
# 26	0.00	0.00	0.64	0.24	0.71	0.29	1.00	0.00
# 27	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00

Tab. 3.18: *Comparison-synergy between decision trees and instance based learning – fuzzy case (distinction is made between speed 1, s1, and 2, s2, in the determination of extended VR).*

	VR fuzzy		VR fuzzy extended s1		VR fuzzy extended s2		VR tree	
Sub-domain	'Compact' category	'Flat' category	'Compact' category	'Flat' category	'Compact' category	'Flat' category	'Compact' category	'Flat' category
# 1	0.02	0.98	0.02	0.98	0.02	0.98	0.00	1.00
# 2	0.00	0.00	0.07	0.24	0.20	0.30	1.00	0.00
# 3	0.00	0.00	0.02	0.08	0.20	0.24	1.00	0.00
# 4	0.01	0.33	0.11	0.60	0.18	0.65	0.00	1.00
# 5	0.25	0.00	0.41	0.13	0.52	0.27	1.00	0.00
# 6	0.00	0.00	0.28	0.06	0.49	0.24	1.00	0.00
# 7	0.00	0.00	0.28	0.45	0.35	0.55	0.00	1.00
# 8	0.71	0.04	0.78	0.09	0.81	0.14	1.00	0.00
# 9	0.29	0.04	0.67	0.15	0.74	0.23	0.00	1.00
# 10	0.50	0.19	0.53	0.32	0.54	0.36	1.00	0.00
# 11	0.00	0.00	0.15	0.11	0.34	0.35	1.00	0.00
# 12	0.00	0.00	0.15	0.05	0.42	0.26	1.00	0.00
# 13	0.33	0.67	0.33	0.67	0.33	0.67	0.00	1.00
# 14	0.16	0.00	0.42	0.22	0.53	0.35	1.00	0.00
# 15	0.41	0.00	0.59	0.08	0.72	0.21	1.00	0.00
# 16	0.00	0.69	0.05	0.81	0.11	0.85	0.00	1.00
# 17	0.06	0.16	0.35	0.43	0.45	0.52	1.00	0.00
# 18	0.63	0.37	0.63	0.37	0.63	0.37	0.00	1.00
# 19	0.00	0.00	0.18	0.11	0.32	0.44	1.00	0.00
# 20	0.00	0.00	0.21	0.18	0.38	0.41	1.00	0.00
# 21	0.00	0.00	0.12	0.08	0.47	0.41	1.00	0.00
# 22	0.00	0.02	0.24	0.39	0.34	0.57	0.00	1.00
# 23	0.49	0.51	0.49	0.51	0.49	0.51	1.00	0.00
# 24	0.02	0.00	0.37	0.30	0.53	0.44	1.00	0.00
# 25	0.00	0.19	0.08	0.57	0.21	0.76	0.00	1.00
# 26	0.02	0.20	0.24	0.60	0.31	0.68	1.00	0.00
# 27	0.22	0.55	0.32	0.65	0.33	0.67	1.00	0.00

At this point, the situation is considered where a partial or null supervision is provided by the operator. Actually, a lack of knowledge or comprehension for the operator may determine three basic issues:

- Select the input parameters and fuzzify[16] them;
- Derive rules to separate training data (automatic identifier);
- Derive concepts / conditions to promote a physical explanation of the problem.

To select and fuzzify input parameters the AI techniques based on decision trees shall be used. In fact, the decision tree algorithms described in this thesis do not require any supervision and provide parameter selection and fuzzyfication. Decision trees also provide a rule set, in both linguistic and in matrix form. Therefore, decision trees allow to achieve an automatic identifier directly from the training dataset, thus bypassing human expert. As a drawback, decision trees do not provide sufficient indications to promote a physical comprehension of the problem. In this light, concept learning algorithms provide both an identification tool and a generalization of the problem, which can be subjected to a direct validation process (on the basis of the comprehension of the physics of the phenomena, of models and so on). However, concept learning algorithms require a previous parameter selection and discretization; hence, they may be applied in series to either decision tree techniques (no supervision) or parameter selection and fuzzyfication provided by human expert (partial supervision).
If the human expert is able to provide both the input parameters and a rule set, he can carry out an automatic identifier with a full supervision; he might even avoid to set up a training database, if has enough knowledge and manages to transfer it to the machine. However, the expert may decide to take advantage of AI techniques at any stage of the process.
Most of the times, the rule set (identifier) does not perform well in all situations. In fact, it is reasonable to assume that the human knowledge is not complete and that the training database is only partially representative of all possible situations. Therefore, it is necessary to dispose of a technique to adjust the identifier on the basis of new experience and to extend its capabilities, thus compensating the lack of prior knowledge. Such a technique is supplied by instance based learning (e.g. the modified k-nearest neighbor described in this paragraph), which allows to adjust and optimize the rule set on the basis of a training dataset or, even, of single instances (provided, e.g., by wrong identifications or improved experimental experience). Hence, instance based learning techniques allow to improve and to tune the identifier, regardless to the way followed to set up the identifier itself. This achievement was obtained thank to the logical-mathematic approach adopted, which constitutes a common basis for all the described techniques.
Furthermore, it is important to discuss the need for checking the identification tools, subjecting the whole identification strategy to a retro-action (revision phase). Such a revision phase should take place on two levels:

- Verify the identifier performance on new data;
- Validate the physical considerations associated to the identification procedure.

[16] It can be observed that the discretization process is intended, in this work, as a particular case of fuzzyfication; in particular, a fuzzyfication with fuzziness degree equal to zero.

These levels of revision are only partially related to each other. To explain this fact an analogy could be done with a grammar problem. One person may be very confident with grammar rules, which provide an explanation of the concepts and indicate a general way to proceed. However, this same person might perpetrate blunder mistakes if he does not have experience on specific situations; in fact, a large amount of exceptions and anomalous situations require specific “rules”. On the contrary, who is very confident with specific situations but not with basic grammar rules would probably fail in solving unknown situations, because of his lack of general comprehension.
The whole process involved in the development of the identification procedure is synthesized in the flow chart of figure 3.37.

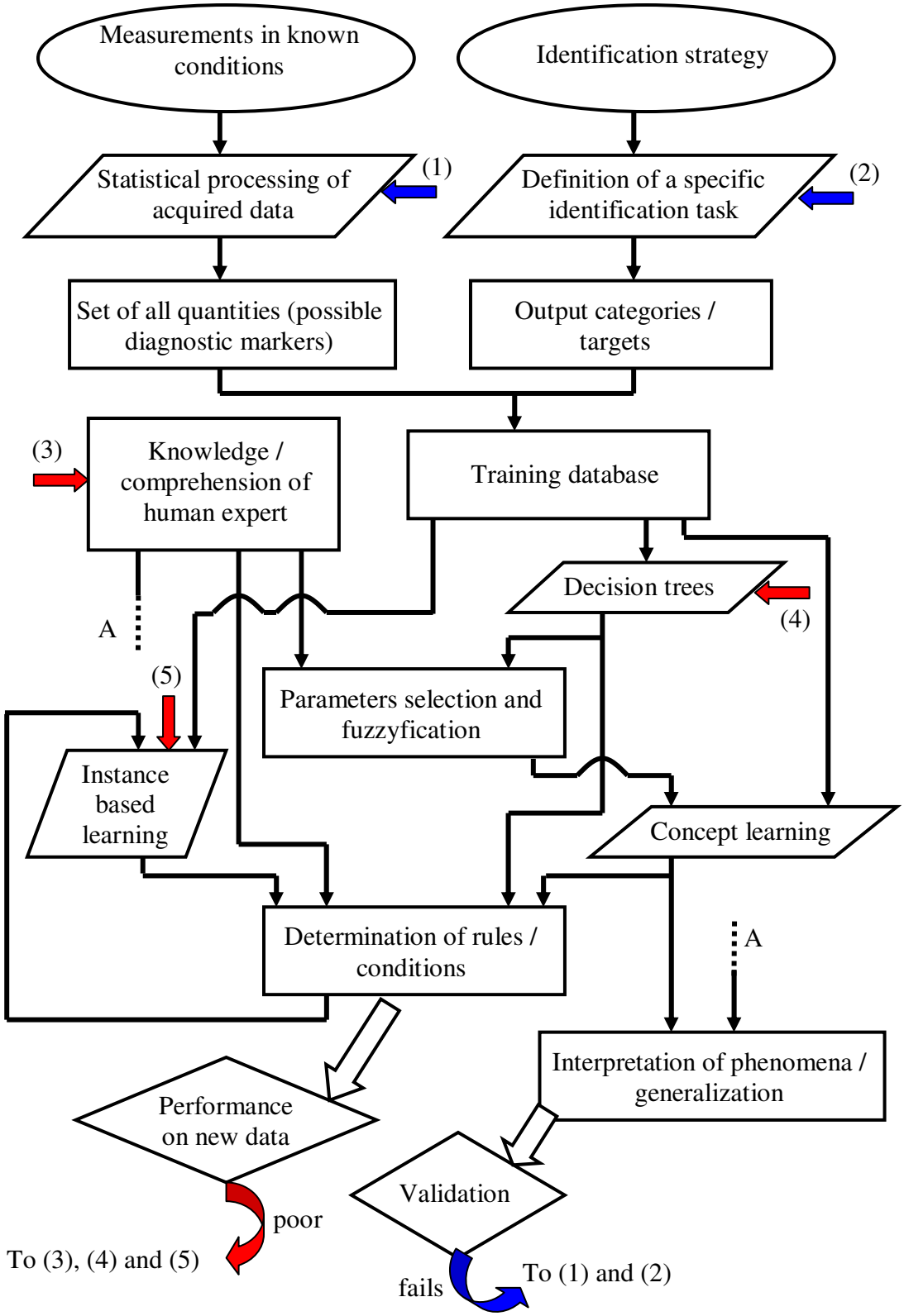


Fig. 3.37: *flow chart which synthesizes the whole process of developing an identification procedure.*

References section 3

[1] A. Jones, A. Kaufmann, H.-J. Zimmermann, *Fuzzy sets theory and applications*, D. Reidel Publishing Company, Dordrecht, Holland, 1986.
[2] Babuska R., *Fuzzy Modelling for Control*, Kluwer Academic Publisher, Boston, 1998.
[3] T. M. Mitchell, *Machine learning*, McGraw-Hill Companies, U.S.A., 1997.
[4] M. M. A. Salama, R. Bartnikas, "Fuzzy logic applied to PD pattern recognition", IEEE Trans. on Dielectrics and Electrical Insulation, Vol.7, No. 1, February, 2000, pp. 118-123.
[5] Danikas, M.G., Gao, N., Aro. M. "Partial discharge recognition using neural networks: a review", Electrical Engineering, Vol. 85, No. 2, May, 2003, pp. 87-93.
[6] Specht, D.F. "Probabilistic neural networks for classification, mapping, or associative memory", in Proc. of the IEEE Int. Conf. on Neural Networks, IEEE ICNN, 1988, Vol. 1, pp. 525 -532.
[7] N. Cristianini, J. Shawe-Taylor, *Support vector machines and other kernel-based learning methods*, Cambridge University Press, 2000.
[8] Abdul Rahman M.K., Arora, R., Srivastava, S.C, "Partial discharge classification using principal component analysis", in IEE Proceedings Science, Measurement and Technology, Vol. 147, No.1, January, 2000, pp. 7-13.
[9] Satish, L., Zaengl, W.S.: "Can fractal features be used for recognizing 3-d partial discharge patterns?", IEEE Trans. on Dielectrics and Electrical Insulation, Vol.2, No. 3, June, 1995, pp. 352-359.
[10] W. J. Krzanowski, *Principles of Multivariate Analysis*, Oxford University Press, 1988.
[11] N. Hozumi, T. Okamoto and T. Imajo, "Discrimination of Discharge Patterns Using a Neural Network", IEEE Trans. on Dielectrics and Electrical Insulation, Vol. 27, n.3, pp.550–556, June 1992.
[12] M.J.D. Powell "An Efficient Method for Finding Variables Without Calculating Derivatives", 6 Computer Journal n. 7, 1964, s. 152-162
[13] V.N. Vapnik "Statistical Learning Theory" John Wiley & Sons, 1998
[14] N. Cristianini, J. Shawe-Taylor "Support Vector Machines and Other Kernel-Based Learning Methods", Cambridge University Press, 2000.
[15] C.C. Lee "Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part I" IEEE Trans. On Syst, Man, and Cybern." Vol. SMC-20, n. 2, pp. 404-418, 1990
[16] C.C. Lee "Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part II" IEEE Trans. On Syst, Man, and Cybern." Vol. SMC-20, n. 2, pp. 419-435, 1990
[17] L.X. Wang and J.M. Mendel "Generating Fuzzy Rules from Numerical Data, with Applications" USC SIPI Report n. 169, University of Southern California, Los Angeles
[18] J.C. Bezdek "Pattern Recognition with Fuzzy Objective Function Algorithmus " Plenum Press, New York, 1981
[19] G. Vachkov "Multilevel Fuzzy Rule Based Modelling", Proceedings EUFIT 1994, Vol. 1, pp

240-245, Aachen, 1994
[20] J.R. Quinlan “C 4.5. Programs for Machine Learning”, Morgan Kaufmann Publishers, San Mateo, California, 1992
[21] A. Srinavasan, C. Batur, Ch.Ch. Chan “Using Inductive Learning to Determine Fuzzy Rules foe Dynamic Systems”, Engng Applic. Artif. Intell., Vol. 6, n. 3, 257-264, Pergamon Press Ltd., 1993
[22] P. Otto “Fuzzy-Modelling of Nonlinear Dynamic Systems by Inductive Learned Rules”, Third European Congress on Intelligence Techniques and Soft Computing, Aachen, August 1995, EUFIT 1995, Proceedings, Vol. 2, pp. 858-864.
[23] Cover, T. M., Hart, P. E.: “Nearest neighbor pattern classification,” IEEE Trans. on Information Theory, Vol. IT-13, January, 1967, pp. 21-27.

Appendix A

Effect of discretization and fuzzyfication on diagnostic databases

As discussed in the second section, a diagnostic database, in particular a training database, consists, in general, of N attribute fields and a target field (figure A.1).
If it is a crisp database, the values in the N attribute fields are real numbers.

Record	Attribute 1	…	Attribute N	Target
1				
2				
…				

Fig. A.1: *representation of a crisp diagnostic database (training database).*

The effect of discretization consists in the fact that the values of the N attribute fields are integers. In fact, the j-th attribute is associated to V_j values; therefore, each integer number in the j-th field is the index which identifies which value in $\{1,\ldots, V_j\}$ is associated to which record.

The effect of fuzzyfication involves an increment of the number of fields in the database. Considering, e.g., the j-th field (attribute), if it has been fuzzyfied in f fuzzy sets, the correspondent fuzzy database comprises f fields pertinent to the j-th attribute, each one containing real values ranging in [0, 1], which specify the membership of each record to a specific fuzzy set for that attribute.

Record	…	Attribute j				…	Target
		A_{j1}	A_{j2}	…	A_{jf}		
1							
2							
…							

Fig. A.2: *representation of a fuzzy diagnostic database (training database).*

Appendix B

Some code about fuzzy and AI tools

(The code reported in the following was written with Matlab 6.1 and is directly linked to the matter discussed in the third section of this thesis.

```
% CONTENTS Artificial Intelligence tools.
%
% FuzzyTools.
%   crisp2fuzzy         - Convert scalar into fuzzy vector.
%   Ftabella2           - Generate multiple subscripts array for a given array size.
%   F_eval_Gen2         - Derive indexes and memberships to sub-domains from crisp input vectors.
%   fuzzyengine2        - Associate crisp input vectors to the output space given knowledge and reference matrixes.
%   examples2matrix     - Build knowledge matrix from training dataset (crisp examples vectors provided with targets).
%   updaterulematrix    - Updates a rule matrix with respect to a specific training example.
%   IBLn1               - Evaluate a sub-domain (query instance) generalizing beyond prior knowledge by means of distances.
%   assigndistances     - Determine distances in discretized space with respect to query instances (specific sub-domains).
%   extendknowledge     - Extend knowledge matrix according to IBLn1.
%   IBLn2               - Like IBLn1, but returns distance matrix as output.
%
% DecisionTrees.
%   ID3n3               - Construct decision tree from a training dataset (it allows pre-pruning).
%   ID3n4               - Like ID3n3, but introduces random variables (genetic mutations).
%   runtree             - Evaluate a crisp input vector with respect to a decision tree.
%   dataseteval         - Evaluate a crisp input dataset with respect to a decision tree.
%   sortparameters      - Sorts columns of a training dataset (continuous-valued attributes) with respect to information gain.
%   findthreshold       - For a specific continuous-valued attribute, finds the value for which the inf. gain is maximum.
%   findthreshold2      - Like sortparameters, but evauates only relative peaks of information gain (more efficient).
%   gain                - Information gain of a training dataset with respect to a given attribute.
%   entropy             - Entropy of a training dataset.
%   indexsubsets        - Indexes of repeated elements in a vector.
%   efficiency          - Evaluate the efficiency of a decision tree with respect to a training dataset.
%   treefitness         - Evaluate the fitness of a decision with repect to efficiency and tree size.
%   squaretree          - Post-pruning algorithm for decision trees.
%   issquare            - True if in a decision tree every parameter is associated to the same number of nodes.
%   desumetargets       - Find target vector for which a decision tree has efficiency 1 with respect to a given dataset.
%   mostra              - Graphical representation of a decision tree.
%   msclbias            - Evaluate misclassified examples with respect to the dataset used to train the decision tree.
%   node2leaf           - Convert a node into a leaf (decision tree pruning process).
%   selecttarget        - Associate a leaf to a target value in pre-pruning process.
%   usedpars            - Indexes of parameters used in a decision tree as columns of the dataset used for training.
%   tree2rules          - Convert a tree into a rule set; fuzzyfies the input parameters and evaluates rule seeds.
%   plotfuzzyref        - Graphical representation of fuzzy references.
%   translatetseeds     - Display tree rules in a linguistic form.
%   tseeds2rseeds       - Convert tree seeds into rule seeds.
%
% ConceptLearning.
%   CEn1                - Candidate-Elimination algorithm, modified to catch outliers.
%   CEn2                - Return necessary and sufficient conditions (rules, hypotheses) for each output category.
%   discretizepar       - Discretize a crisp scalar input with respect to a set of crisp references.
%   discretizeparvec    - Discretize a crisp array input with respect to a set of crisp references.
%   translatecondition  - Displays conditions (hypotheses) in a linguistic form.
%   comparehypotheses   - Compare hypotheses with respect to partial general-to-specific ordering.
%   comparestatements   - Compare hypotheses along a specific dimension (attribute).
%   generalizehyp       - Minimal generalization of a hypothesis with respect to an instance.
%   generatesubsets     - Generate the set of all hypotheses obtainable by the conjunction of generic subsets of attribute
values.
%   isconsistent        - True if a hypothesis is consistent with a given instance.
%   rminconsistenthyps  - Remove inconsistent hypotheses from a given set.
%   specifyhyp          - Minimal specializations of a hypothesis with respect to an instance.
```

```
%
```

FUZZY TOOLS

```
function mi = crisp2fuzzy(rif,x)
%
% mi = crisp2fuzzy(rif,x)
%
% prende in ingresso il vettore dei riferimenti e il valore
% calcolato del parametro fuzzyficato (x); restituisce
% il vettore delle membership (se ad esempio si considerano 5
% insiemi fuzzy (B, MB, M, MA, A), length(rif)=10 e length(mi)=5,
% con il vincolo che sum(mi)=1); x è il valore del parametro
% che si intende valutare
%

FR = length(rif);
if (FR/2)~=round(FR/2)
  disp('ERRORE: i riferimenti devono essere in numero pari')
  help crisp2fuzzy
  return
end
mi = zeros(1,(FR/2));
rif = sort(rif);
for k=1:2:(FR-1)
  if x>=rif(k) & x<=rif(k+1)
    mi((k+1)/2) = 1;
  end
end
for k=2:2:(FR-2)
  if x>rif(k) & x<rif(k+1)
    mi(k/2) = (rif(k+1)-x)/(rif(k+1)-rif(k));
    mi((k/2)+1) = 1 - mi(k/2);
  end
end
```

****

```
function T = ftabella2(F,m)
%
% T = ftabella2(F,m)
%
% F: vector of natural numbers.
% m: scalar natural value; if this second argument is passed,
%    F is set equal to ones(1,F(1))*m
%
% T: array with r = prod(F) rows and c = length(F) columns.
%    Rows of T represent all possible vectors of c natural
%    numbers, where the k-th element can assume values in
%    [0:F(k)-1]. Rows are sorted in ascending order, as if
%    were c-digits numbers, being the k-th number expressed
%    in base F(k).
%
if nargin==2
   F = ones(1,F(1))*m;
elseif nargin~=1
  disp('Warning: wrong number of inputs')
  help Ftabella2
end
T = [];
```

```
for k=1:length(F)
   f = F(k);
   La = prod(F(k:end));
   C = costruiscicolonna(prod(F)/La,La,F(k));
   T = cat(2,T,C);
end

% ------------------------------------------------

function C = costruiscicolonna(Na,La,Ne)
% C = costruiscicolonna(Na,La,Ne)
a = [];
for k=1:Ne
   for s=1:La/Ne
      a = cat(1,a,k-1);
   end
end
C = [];
for k=1:Na
   C = cat(1,C,a);
end
```

****

```
function IDstatus = F_eval_Gen2(Fmat)
%
% IDstatus = F_eval_Gen2(Fmat)
%
% Fmat: cell array (p x 1), Fmat{k} if a vector (1 x F(k)), containing
%       the membership of the k-th parameter to its F(k) fuzzy sets,
%       where p is the number of parameters.
%
% IDstatus: array (2xN), with N number of elementary status activated;
%           IDstatus(1,:) contains status indexes, IDstatus(2,:) contains
%           their memberships. Rows are sorted in descending order of membership.
%           The number of status simultaneously activated will range in
%           [1,z^p], where z is the number of fuzzy sets that overlap
%           (for each parameter), thus that can be activated at the same time.
%           The total number of elementary status is prod(length(Fmat{k}), with
%           k=1:length(Fmat).
%
% See also: Ftabella2
%

p = size(Fmat,1);
f = zeros(1,p);
for k=1:p
   f(k) = length(Fmat{k});
end

M0 = zeros(f);
coords = cell(1,p);
for k=1:p
   coords(k) = {find(Fmat{k})};
end
D.type = '()';
D.subs = coords;
M1 = subsasgn(M0,D,1);
index = find(M1);
IDstatus = cat(1,index',zeros(1,length(index)));

T = fliplr(Ftabella2(f) + 1); % call Ftabella2
```

```
for k=1:length(index)
   RowkT = T(index(k),:);
   Vmi = zeros(1,p);
   for s=1:p
      Fmat_s = Fmat{s};
      Vmi(s) = Fmat_s(RowkT(s));
   end
   IDstatus(2,k) = prod(Vmi);
end
IDstatusS = sortrows(IDstatus');
IDstatus = transpose(flipud(IDstatusS));
```

****

```
function [Vout,Fmat,IDstatus] = fuzzyengine2(Vin,Mreferences,VRules)
%
% [Vout,Fmat,IDstatus] = fuzzyengine2(Vin,Mreferences,VRulrs)
%
% Vin: input array (1 x p), containing crisp values of the the p input quantities.
% Mreferences: array (p x 2*f), with f fuzzy sets.
% VRules: array (f^p x m) with m output categories, such that
%          sum(VRules(k,:))=1 for any k in [1:f^p].
% Vout: array (1 x m) of the memberships to the output categories.
% Fmat: array (p x f) containing fuzzyfied values of the the p input quantities.
% IDstatus: array (2 x N), being N the number of elementary status activated;
% the first row contains the status indexes, the second row contains their memberships.
% The total number of elementary status (Ntot) is f^p.
%
% See also: examples2matrix  IBLn1  extendknowledge  ID3n3  tree2rules  CEn2
%          F_eval_Gen2  crisp2fuzzy
%

if ne(nargin,3)
   disp('Error in input variable assignment')
   help fuzzyengine2
   return
end
p = size(Mreferences,1);
f = zeros(1,p);
for k=1:p
   f(k) = length(Mreferences{k})/2;
end
[Ntot,m] = size(VRules);
if not(isequal(Ntot,prod(f)))
   disp('Warning: size of VRulrs and of Mreferences are not consistent')
   help fuzzyengine2
   return
end
if not(isequal(length(Vin),p))
   disp('Warning: size of Vin and of Mreferences are not consistent')
   help fuzzyengine2
   return
end
%
Fmat = cell(p,1);
for k=1:p
   Fmat(k) = {crisp2fuzzy(Mreferences{k},Vin(k))}; % call crisp2fuzzy
end
IDstatus = F_eval_Gen2(Fmat); % call F_eval_Gen2
Nstatus = size(IDstatus,2);
Mout = zeros(Nstatus,m);
for k=1:Nstatus
```

```
   Mout(k,:) = VRules(IDstatus(1,k),:) * IDstatus(2,k);
end
Vout = sum(Mout,1);
```

****

```
function [VR,F] = examples2matrix(X,T,Ref,FuzzyFlag,SaturationOn)
%
% [VR,F] = examples2matrix(X,T,Ref,FuzzyFlag,SaturationOn)
%
% ( Converts a training dataset into a rule matrix, both in the crisp (discrete)
%   and in the fuzzy case, developed by Marco Conti, Bologna, 29/01/2004 )
% examples2matrix allows the construction of a matrix to store prior knowledge contained
% in a training dataset. Its output can be used by functions fuzzyengine2 and IBLn1,
% allowing instance-based identification and learning.
%
% F: vector (1 x p), with p number of input parameters (obtained as size(X,2) or length(Ref));
%    its k-th element indicates the number of sets (fk) for the k-th parameter.
% VR: array (Nout x prod(F)); Nout is the number of output categories (obtained as length(unique(T)));
%     prod(F) is the total number of elementary sub-domains provided by
%     the partition of each parameter in (fuzzy) sets. The k-th row of VR indicates
%     the vector of membership to the output categories assigned to that sub-domain.
%     VR is subjected to partial normalization, so that sum(VR(k,:))<=1, for every row index k.
%
% X: array (N x p), set of N training instances (examples), each one containing
%    values of p attributes.
% T: vector (N x 1), target vector. T(j) belongs to {1,...,Nout}, with
%    Nout number of output categories.
% FuzzyFlag: logic constant; 0 (default) refers to the crisp case, 1 to the fuzzy case.
% Ref: cell array (p x 1). Cref{k} is a vector containing Nrefk discretization references.
%       Nrefk is equal to F(k)+1 in the crisp case, 2*F(k) in the fuzzy case.
% SaturationOn: logic constant (default is 0); if true, training examples which refers to the same
%               sub-domain and are pertinent to the same output category increase the vale of the
%               correspondent output until it reaches value 1 (saturation threshold value); otherwise,
%               the saturation threshold value is set to the largest real number.
%
% Observe that, if SaturationOn is false, in the generation of the rule matrix VR more weight is given
% to those categories which are more represented in the training database, i.e. the ones that have a
% larger number of positive examples.
%
% See also: extendknowledge  updaterulematrix  IBLn1  ID3n3  tree2rules  CEn2  Fuzzyengine2
%

if nargin==3
   FuzzyFlag = 0;
   SaturationOn = 1;
elseif nargin==4
   SaturationOn = 1;
elseif ne(nargin,5)
   disp('Attenzione: numero di ingressi errato')
   help examples2matrix
   return
end
if SaturationOn
   ThS = 1;
else
   ThS = realmax;
end
[N,p] = size(X);
if length(Ref)~=p
   disp('Errore nell''assegnazione dei riferimenti')
   help examples2matrix
```

```
   return
end
categories = unique(T);
Nout = length(categories);
T2 = zeros(size(T));
for k=1:Nout
   T2(find(T==categories(k))) = k; % make sure categories are addressed
end                        % to by means of the first Nout natural numbers
F = zeros(1,p);
for k=1:p
   if FuzzyFlag
      F(k) = length(Ref{k})/2;
   else
      F(k) = length(Ref{k})-1;
   end
end
VR = zeros(prod(F),Nout);

if FuzzyFlag% fuzzy case
   for k=1:N
      Vin = X(k,:);
      Fmat = cell(p,1);
      for z=1:p
         Fmat(z) = {crisp2fuzzy(Ref{z},Vin(z))}; % call crisp2fuzzy
      end
      IDstatus = F_eval_Gen2(Fmat); % call F_eval_Gen2
      for z=1:size(IDstatus,2)
         val = VR(IDstatus(1,z),T2(k));
         VR(IDstatus(1,z),T2(k)) = min(ThS,IDstatus(2,z)+val);
      end
   end
else%        crisp case
   for k=1:N
      xD = discretizeparvec(X(k,:),Ref); % call discretizeparvec
      coordsk = zeros(1,p);
      for z=1:p
         coordsk(z) = find(xD{z});
      end
      indk = sub2ind_p(coordsk,F); % call sub2ind_p (see assigndistances)
      val = VR(indk,T2(k));
      VR(indk,T2(k)) = min(ThS,val+1);
   end
end
for k=1:prod(F)
   z = VR(k,:);
   if sum(z)>1
      VR(k,:) = z/sum(z); % normalization
   end
end
```

****

```
function [VR,F] = updaterulematrix(VR,x,t,Ref,FuzzyFlag,SaturationOn)
%
% [VR,F] = examples2matrix(VR,x,t,Ref,FuzzyFlag,SaturationOn)
%
% ( Updates a rule matrix with respect to a specific training example,  both in the crisp (discrete)
%   and in the fuzzy case, developed by Marco Conti, Bologna, 09/02/2004 )
%
% F: vector (1 x p), with p number of input parameters (obtained as size(X,2) or length(Ref));
%    its k-th element indicates the number of sets (fk) for the k-th parameter.
% VR: array (Nout x prod(F)); Nout is the number of output categories (obtained as length(unique(T)));
```

```
%     prod(F) is the total number of elementary sub-domains provided by
%     the partition of each parameter in (fuzzy) sets. The k-th row of VR indicates
%     the vector of membership to the output categories assigned to that sub-domain.
%     VR is subjected to partial normalization, so that sum(VR(k,:))<=1, for every row index k.
%
% x: vector (1 x p), training instance (example), containing values of p attributes.
% t: scalar target. t must belong to belong to {1,...,Nout}, with Nout number of output categories.
% FuzzyFlag: logic constant; 0 (default) refers to the crisp case, 1 to the fuzzy case.
% Ref: cell array (p x 1). Cref{k} is a vector containing Nrefk discretization references.
%       Nrefk is equal to F(k)+1 in the crisp case, 2*F(k) in the fuzzy case.
% SaturationOn: logic constant (default is 0); if true, training examples which refers to the same
%               sub-domain and are pertinent to the same output category increase the value of the
%               correspondent output until it reaches value 1 (saturation threshold value); otherwise,
%               the saturation threshold value is set to the largest real number.
%
% Observe that, if SaturationOn is false, the output associated to the input instance and target is
% increased only if its current value is smaller than 1
%
% See also: examples2matrix extendknowledge  IBLn1  ID3n3  tree2rules  CEn2  Fuzzyengine2
%

if nargin==4
   FuzzyFlag = 0;
   SaturationOn = 1;
elseif nargin==5
   SaturationOn = 1;
elseif ne(nargin,6)
   disp('Attenzione: numero di ingressi errato')
   help updaterulematrix
   return
end
if SaturationOn
   ThS = 1;
else
   ThS = realmax;
end
p = length(x);
if length(Ref)~=p
   disp('Errore nell''assegnazione dei riferimenti')
   help updaterulematrix
   return
end
Nout = size(VR,2);
t = ceil(t);
if t>Nout
   disp('Errore nell''assegnazione del target')
   help updaterulematrix
   return
end
F = zeros(1,p);
for k=1:p
   if FuzzyFlag
      F(k) = length(Ref{k})/2;
   else
      F(k) = length(Ref{k})-1;
   end
end
if FuzzyFlag% fuzzy case
   Fmat = cell(p,1);
   for z=1:p
      Fmat(z) = {crisp2fuzzy(Ref{z},x(z))}; % call crisp2fuzzy
   end
```

```
   IDstatus = F_eval_Gen2(Fmat); % call F_eval_Gen2
   for z=1:size(IDstatus,2)
      val = VR(IDstatus(1,z),t);
      VR(IDstatus(1,z),t) = min(ThS,IDstatus(2,z)+val);
   end
else%        crisp case
   xD = discretizeparvec(x,Ref); % call discretizeparvec
   coordsk = zeros(1,p);
   for z=1:p
      coordsk(z) = find(xD{z});
   end
   indk = sub2ind_p(coordsk,F); % call sub2ind_p (see assigndistances)
   val = VR(indk,T2(k));
   VR(indk,T2(k)) = min(ThS,val+1);
end
for k=1:prod(F)
   z = VR(k,:);
   if sum(z)>1
      VR(k,:) = z/sum(z); % normalization
   end
end
```

****

```
function [Vout,Md] = IBLn1(xq,F,VR,N,Lthresh)
%
% [Vout,Md] = IBLn1(xq,F,VR,N,Lthresh)
%
% ( Instance Based Learning algorithm, by Marco Conti, Bologna 29/01/2004 )
% IBLn1 classifies a query instance on the basis of prior knowledge stored in a matrix;
% it averages the outputs of knownd instances according to their distance from the
% query instance.
%
% xq: query instance. It can be either a vector of indexes (1 x p) or a scalar index.
%     In the former case xq(k) is the value assumed by the k-th parameter in the query instance;
%     in the latter case the index refers to the matrix of all instances.
% F: vector (1 x p), with p number of input parameters (dimensions); F(k) is the size of the k-th
%    dimension, i.e. the number of values that the k-th parameter may assume.
% VR: array (prod(F) x Nout), with Nout number of output categories. Its k-th row specifies the
%     output vector associated to the k-th elementary sub-domain (instance). VR stores the prior knowledge.
% N (optional): maximum number of order of neighborhood to be considered. Default is max(F)
% Lthresh (optional): likelihood threshold. The algorithm stops as soon as the likelihood (sum(Vout))
%                     exceeds Lthresh. Default is 1.
%
% Vout: output vector (1 x Nout)
% Md: matrix of distances (such that size(Md)=F) evaluated with respect of xq on the basis of the equation:
%     distance(xk) = sum(abs(xk - xq)), where instances xk and xq are expressed as vectors of indexes in
%     a matrix of size F.
%
% See also: examples2matrix  extendknowledge  ID3n3  tree2rules  CEn2
%

if nargin==4
   Lthresh = 1;
elseif nargin==3
   Lthresh = 1;
   N = max(F);
elseif ne(nargin,5)
   disp('Attenzione: numero delle variabili in ingresso non corretto')
   help IBLn1
   return
end
```

```
if length(xq)==1
   xq = ind2sub_p(xq,F); % call ind2sub_p (see assigndistances)
end
Vout = zeros(1,size(VR,2));
Md = zeros(F);
Ntot = prod(F);
for k=1:Ntot
   xk = ind2sub_p(k,F); % call ind2sub_p
   Md(k) = sum(abs(xk - xq)); % calculate distance
end
Nmax = max(reshape(Md,1,Ntot));
N = min(N,Nmax);

flag = 1;
order = 1; % order of neighborhood
L = 0; % likelihood
w = 1; % weight
while flag
   IndOrd = find(Md==order);
   VRneighbor = VR(IndOrd,:);
   Vout = Vout + mean(VRneighbor,1) * w; % the outputs of those instances which have
   L = sum(Vout);  % a distance equal to 'order' from xq are averaged and weighted.
   if L>=Lthresh | order==N
      flag = 0;
   else
      order = order + 1;
      w = w * (1 - mean(sum(VRneighbor,2))); % update weight
   end
end
```

****

```
function Md = assigndistances(F,xq)
%
% Md = assigndistances(F,xq)
%

Md = zeros(F);
Ntot = prod(F);
for k=1:Ntot
   xk = privat_ind2sub_p(k,F);
   Md(k) = sum(abs(xk - xq)); % distance
end

% --------------------------------------------------

function indvec = privat_ind2sub_p(ind,siz)
% indvec = ind2sub_p(ind,siz)
vc = cell(size(siz));
try
   [vc{:}] = ind2sub(siz,ind);
   indvec = cat(2,vc{:});
catch
   disp(lasterr)
   help ind2sub_p
end
```

****

```
function VRextended = extendknowledge(VR,F,speed,Lth)
%
% VRout = extendknowledge(VR,F,speed,Lth)
%
% ( Instance Based Learning algorithm, by Marco Conti, Bologna 01/02/2004 )
% Expands the knowledge contained in a rule matrix, on the basis of distances,
% either in the crisp or in the fuzzy case.
%
% F: vector (1 x p), with p number of input parameters; its k-th element indicates the
%    number of sets (fk) for the k-th parameter.
% VR: array (Nout x prod(F)); Nout is the number of output categories;
%     prod(F) is the total number of elementary sub-domains provided by
%     the partition of each parameter in (fuzzy) sets. The k-th row of VR indicates
%     the vector of membership to the output categories assigned to that sub-domain.
% speed (optional): natural scalar number (default is 1); represents the number of orders of
%                   neighborhoods taken into accout at each step.
% Lth (optional): likelihood threshold; the algorithm ends when every sub-domain is associated
%                 to the output space with a likelihood >= lth. Default is 0.1.
%
% VRextended: array of the same size of VR, such that every sub-domain is associated
%             to an output vector; in this way every possible query instance is identified
%             with a likelihood at least equal to Lth (i.e. it is true that all(sum(VRextended,2)>=Lth)).
%
% See also: examples2matrix  updaterulematrix  IBLn1  ID3n3  tree2rules  Fuzzyengine2
%

if nargin==2
   speed = 1;
   Lth = 0.1;
elseif nargin==3
   Lth = 0.1;
elseif ne(nargin,4)
   disp('Wrong number of inputs')
   help extendknowledge
   return
end

speed = ceil(speed);
Lth = min(max(Lth,0),1);
N = size(VR,1);
flag = ~all(sum(VR,2)>=Lth);
while flag
   for k=1:N
      yk = VR(k,:);
      sk = sum(yk);
      if sk<1% Lth (in alternativa)
         Vout = IBLn1(k,F,VR,speed,1); % call IBLn1
         VR(k,:) = yk + Vout*(1-sk);
      end
   end
   flag = ~all(sum(VR,2)>=Lth);
end
VRextended = VR;
```

****

DECISION TREES

```
function Tree = ID3n3(X,T,pref,N,k,Z,Tree)
%
% Tree = ID3n3(X,T,pref)
%
% ( Decision Tree construction algorithm, by Marco Conti, Bologna, 2003 )
% ID3n3 searches for a decision tree which completely separates continuous-valued
% training examples of matrix X with respect to target vector T (pref is optioal
% and regulates a pre-pruning process of the tree).
%
% X: array (N x na), set of N training instances, each one containing
%    values of na attributes.
% T: vector (N x 1), target vector. T(j) belongs to {1,...,Nout}, with
%    Nout number of output categories.
% pref: reference value for pre-pruning. It corresponds to the maximum value
%       of efficiency reduction for the tree which is accepted, at each node,
%       to end the current branch growth and put a leaf.
%       Default value: pref = 0 (no pre-pruning).
%
% Tree: structure (1 x Z), representing a decision tree with Z nodes (including leaf nodes).
%       Fields:
%            - Parameter: column index in X indicating the parameter
%                         pertinent to that node
%            - Threshold: value of the parameter such that the database is
%                         partitioned according to (parameter <= threshold)
%            - ParentNode: index in Tree of the node which has in output the
%                          current node
%            - ParentBranch: indicates which output of the parent node is connected
%                            to the current node (0 if parameter > threshold, 1 othewise)
%            - IsLeaf: 1 if current node is a leaf, 0 otherwise
%            - AssociatedRecords: if IsLeaf=1 is the indexe vector (rows of X) indicating
%                                 the records pertinent to that leaf, otherwise is empty.
%            - AssociatedTarget: if IsLeaf=1 is the value of the target associated to that
%                                leaf, otherwise is empty
%            - Level: natural number inicating the level of the current node in the tree,
%                     i.e. 1 + the number of nodes above the curent one along any branch.
%
% See also: sortparameters tree2rules ID3n4 runtree dataseteval desumetargets efficiency
%           treefitness squaretree issquare usedpars msclbias mostra CEn2 Fuzzyengine2
%

%nargchk(2,6,nargin);
if nargin<4 % first call
   X = cat(2,[1:size(X,1)]',X); % attach row indexes to training set
   N = 0; k = 0; Z = T; Node = 1; % initialization
   Tree = struct('Parameter',[],'Threshold',[],'ParentNode',[],'ParentBranch',[],...
      'IsLeaf',0,'AssociatedRecords',[],'AssociatedTarget',[],'Level',[]);
   if nargin<3
      pref = 0; % pre-pruning unabled
   end
else
   Node = length(Tree) + 1;
end

Xp = X(:,2:end);
[Nexamples,Npars] = size(Xp);
[I,Thresholds] = sortparameters(Xp,T); % call sortparameters
A = Xp(:,I(1)); % select attribute with maximum information gain
Threshold = Thresholds(1,I(1)); % select Threshold with maximum information gain
```

```
N = N + 1; % go down one level
Tree(Node).Level = N; %     update Tree structure
Tree(Node).Parameter = I(1);
Tree(Node).Threshold = Threshold;
Tree(Node).ParentNode = findparent(Tree,N-1);
Tree(Node).ParentBranch = k;
Tree(Node).IsLeaf = 0;

Ad = A<=Threshold; % evaluate 'A <= Threshold' expression ==> 2 branches out of each node
CA = indexsubsets(Ad);  % call indexsubsets

for k=1:length(CA)
   Ik = CA{k};
   Tk = T(Ik);
   if pref>0
      [flag,AssociatedTarget] = prepruning(Z,Tk,pref);  % call prepruning (privat function)
   else % pre-pruning is not active
      flag = 1;
      AssociatedTarget = unique(Tk);
   end
   if entropy(Tk) & flag
      Tree = feval(mfilename,X(Ik,:),Tk,pref,N,k,Z,Tree); % recursive call
   else
      Node = length(Tree) + 1;
      Tree(Node).Level = N+1; %        update Tree structure
      Tree(Node).ParentNode = findparent(Tree,N); % call findparent (privat function)
      Tree(Node).ParentBranch = k;
      Tree(Node).IsLeaf = 1;
      Tree(Node).AssociatedRecords = X(Ik,1);
      Tree(Node).AssociatedTarget = AssociatedTarget;
   end
end


% ------------------------ END OF ID3n3 --------------------------

function Parent = findparent(Tree,L)
% Parent = findparent(Tree,L)
if L==0 % root
   Parent = 0;
else
   leafs = cat(1,Tree(:).IsLeaf);
   NodesInd = find(leafs==0);
   Level = cat(1,Tree(:).Level);
   ParentCandidates = intersect(NodesInd,find(Level==L));
   Parent = ParentCandidates(end);
end

% ---------------------------------------------------------------

function [flag,AssociatedTarget] = prepruning(T,Tcurrent,pref)
% [flag,AssociatedTarget] = prepruning(T,Tcurrent,pref)
if nargin==2
   pref = 0.05;
end
[AssociatedTarget,LeafResultS] = selecttarget(T,Tcurrent); % call selecttarget
c = size(LeafResultS,1);
p = sum(LeafResultS(1:c-1,1)); % quote of misclassified examples (decrement of relative efficiency)
Deff = p/c; % efficiency reduction
flag = Deff > pref;
%               flag=1 ==> go ahead with current branch;
```

```
%                flag=0 ==> put a leaf below current branch.


                                        ************************

function [C,N] = runtree(ParSet,Tree)
%
% [OutputCategory,LeafIndex] = runtree(ParSet,Tree)
%
% ParSet: vector (1 x n) of input parameters to be evaluated.
%         n is the number of columns of the dataset X used to train Tree;
%         in general, n<p (p number of input parameters). To see which of the n
%         attributes contained in X are used as input parameters, see 'usedpars'
% Tree: tree structure; length(ParSet) must not exceed the highest value in the
%       field Parameter of the structure Tree.
%
% OutputCategory: value of the output category associated to ParSet
%                 (identification process), with respect to the target
%                 vector (T) used to construct Tree (T=desumetargets(Tree)).
% LeafIndex: index that identifies the leaf ending the evaluated
%            branch, i.e. the rule used to achieve identification.
%

[a,ParTable] = issquare(Tree);
if length(ParSet)<max(ParTable(:,1))
   disp('Warning: ParSet is not suitable for Tree');
   help runtree
   return
end

Parents = cat(1,Tree(:).ParentNode);
BranchesC = indexsubsets(cat(1,Tree(:).ParentBranch));
flag = 0;
N = 1;
while ~flag
   Parameter = Tree(N).Parameter;
   Threshold = Tree(N).Threshold;
   Branch = (ParSet(Parameter)<=Threshold) + 1;
   N = intersect(find(Parents==N),BranchesC{Branch+1});
   flag = Tree(N).IsLeaf;
end
C = Tree(N).AssociatedTarget;


                                        ************************

function [Tevluated,Icatched,Imissed] = dataseteval(X,Tree,T)
%
% [Tevluated,Icatched,Imissed] = dataseteval(X,Tree,T)
%
% X: array (m x p), constituting a dataset with m records, each one
%    providing values pertient to p attributes.
% Tree: decision tree derived from Xtrain by assigning a target vector (Ttrain).
%       Number of columns of X must be greater or equal to that of Xtrain.
% T: vector (m x 1) of targets specified for X (optional input).
%
% Tevluated: new target vector, obtained by running Tree for each one of the m records.
% Icatched: (available only if T is provided as input) indexes of records assigned
%           by Tree to the output cateories specified by T.
% Imissed: (available only if T is provided as input) indexes of records assigned
%           by Tree to the output cateories other than those specified by T.
```

```
%

msg = nargchk(2,3,nargin);
if not(isempty(msg))
   disp(msg)
   help dataseteval
   return
end
m = size(X,1);
Tevluated = zeros(m,1);
for k=1:m
   Tevluated(k) = runtree(X(k,:),Tree);
end
if nargin>2 & nargout>1
   Icatched = find(Tevluated==T);
   if nargout>2
      Imissed = setdiff([1:m]',Icatched);
   end
end


                                        ************************
function [I,Thresholds,Parade] = sortparameters(X,T,n)
%
% [I,Thresholds,Parade] = sortparameters(X,T,n)
%
% X: array (Nexamples x Npars), containing Nexamples values
%    for each of the Npars parameters.
% T: target vector (Nexamples x 1)
% n: number of returned values, for each parameter, that
%    maximize information gain for that parameter.
%
% I: index that sorts parameters according to their information gain,
%    in descending order.
% Thresholds: array (n x Npars) containing values that maximize
%            information gain for each parameter. Values are sorted
%            in descending order of information gain with row index.
% Parade: array (Npars x 2); the first column contains indexes, in such
%         a way that its i-th element is the rank of i-th parameter;
%         in the second column the i-th value of the maximum gain of the
%         i-th parameter.
%
% See also: findthreshold2  gain  entropy  indexsubsets
%

if nargin==2
   n = 1;
elseif ne(nargin,3)
   disp('Error: wrong number of inputs');
   help sortparameters
   return
end

[Nexamples,Npars] = size(X);
G = zeros(Npars,1);
Thresholds = zeros(n,Npars);
for k=1:Npars
   [Thresh_k,gainvec] = findthreshold2(T,X(:,k),n); % call findthreshold2
   G(k) = gainvec(1);
   Thresholds(:,k) = Thresh_k;
end
[G,I] = sort(G);
```

```
G = flipud(G);
I = flipud(I);
z = zeros(Npars,1);
z(I) = [1:Npars]';
Parade = cat(2,z,G(z));
```

****

```
function [Thresholds,G] = findthreshold2(T,P,n)
%
% [Thresholds,G] = findthreshold2(T,P,n)
%
% T: target vector
% P: attribute vector (continuous-valued).
% n: number of returned values of the attribute, for which information gain is maximum,
%    sorted for decreasing values of gain (default is 1).
%
% Thresholds: values of P that maximize the information gain, by evaluating
%             the condition P<value(k).  Thresholds is an array (n x 1).
% G: vector containg all calculated values of the information gain.
%    Length of G is equal to the number of variations in the T vector sorted
%    for increasing values of P.
%

if nargin==2
   n = 1;
elseif ne(nargin,3)
   disp('Error: wrong number of inputs');
   help findthreshold2
   return
end

[Ps,I] = sort(P);
Ts = T(I);
Tdiff = Ts(1:end-1) - Ts(2:end);
Iboundaries = I(find(Tdiff));
X = P(Iboundaries);
Nb = length(Iboundaries);
G = zeros(Nb,1);
for k=1:Nb
   A = P<=X(k);
   G(k) = gain(T,A);
end
[Gs,Imaxgain] = sort(G);
n2 = min(n,Nb);
Imaxgain = flipud(Imaxgain);
Thresholds = X(Imaxgain(1:n2));

if nargout==0
   figure
   h = bar(P(Iboundaries),G,'Linewidth',3);
   set(h,'Linewidth',3)
   set(gca,'XTick',unique(sort(X)))
else
   G = flipud(Gs);
End
```

****

```
function G = gain(S,A)
%
% G = gain(S,A)
%
% S: arrey of doubles (target vector)
% A: arrey of doubles, with the same number of elements as S;
%    A represents the values of a certain attribute.
% G (information gain): expected reduction in entropy obtained
%                       by partitioning the examples with
%                       target S according to attribute A.
%

CS = indexsubsets(S);
CA = indexsubsets(A);
cS = length(CS);
cA = length(CA);
n = length(S);
Gv = zeros(cA,1);

for k=1:cA
   Ik = CA{k};
   nk = length(Ik);
   Gv(k) = (nk/n)*entropy(S(Ik));
end
G = entropy(S) - sum(Gv);
```

****

```
function E = entropy(S)
%
% E = entropy(S)
%
% S: arrey of doubles (target vector)
% E: entropy of S. Consider S as reshape(S,prod(size(S)),1)
%    If S contains c different elements, and the i-th element is repeated
%    ni times, E is obtained adding c elements like -pi*log2(pi) ,
%    where pi is ni/prod(size(S)).
%
% Examples:
%
% S = cat(2,zeros(1,n),ones(1,n))*m ==>
% E = 1 for every n natural, m real
%
% S = ones(1,n)*m ==>
% E = 0 for every n natural, m real
%
% S = [1:n] ==>
% E > 1 (larger values of E correspond to larger values of n).
%

C = indexsubsets(S);
c = length(C);
n = prod(size(S));
Ev = zeros(c,1);
for k=1:c
   p = length(C{k})/n;
   Ev(k) = -p*log2(p);
end
E = sum(Ev);
```

****

```
function C = indexsubsets(S)
%
% C = indexsubsets(S)
%
% S: arrey of doubles
% C: cell array which contains indexes of the positions
%    of each element of S. If none of the elements of S
%    is repeated iside S, C is the same as num2cell([1:length(S)]').
%
% Example:
% S = [1 4 4 3 1 1 3 4 1] ==>
% C = {[1 5 6 9];[4 7];[2 3 8]}
%

S = reshape(S,prod(size(S)),1);
Sshrinked = unique(S);
n = length(Sshrinked);
C = cell(n,1);
for k=1:n
   C(k) = {find(S==Sshrinked(k))};
End
```

****

```
function [eff,eff_rel,effm] = efficiency(X,Tree,T)
%
% [eff1,eff_rel,eff2] = efficiency(X,Tree,T)
%
% X: array (m x p), constituting a dataset with m records, each one
%    providing values pertient to p attributes.
% Tree: decision tree derived from Xtrain by assigning a target vector (Ttrain).
%       Number of columns of X must be greater or equal to that of Xtrain.
% T: vector (m x 1) of targets specified for X (optional input).
%
% eff (efficiency): 1 - M/M, M = total number of misclassified records,
%                           N = total number of records
% eff_rel (relative efficiency): vector (1 x Nout). Nout be the number of output
%                              categories. eff_rel(k) = 1 - mk/nk
%                              mk = number of misclassified records for the k-th output category
%                              nk = number of  records for the k-th output category
% effm (mean efficiency): sum(eff_rel)/Nout
%

if ne(nargin,3)
   disp('Wrong number of inputs')
   help efficiency
   return
end

[Tevluated,Icatched,Imissed] = dataseteval(X,Tree,T);
CT = indexsubsets(T);
Nout = length(CT);
[n,m,eff_rel] = deal(zeros(1,Nout));
for k=1:Nout
   Ck = CT{k};
   n(k) = length(Ck);
   m(k) = length(intersect(Ck,Imissed));
end
```

```
eff_rel = 1 - m./n;
eff = 1 - length(Imissed)/length(T);
effm = sum(eff_rel)/Nout;
```

****

```
function y = treefitness(eff,n,a,b)
%
% y = treefitness(eff,n,a,b)
%
% y: value of the fitness function for a tree algorythm
%       y = eff^a * n^b
%
% eff: tree efficiency (see function 'efficiency')
% n: number of nodes (excluding leafs)
%
% Default values for a and b: a = 1  ; b = -0.2
%

if nargin==2
   a = 1;
   b = -0.2;
elseif ne(nargin,4)
   disp('Error: wrong number of inputs')
   help treefitness
   return
end

y = eff.^a .* n.^b;
```

****

```
function [Tree,Success] = squaretree(Tree)
%
% TreeOut = squaretree(Tree)
%
%

T = desumetargets(Tree);
Levels = cat(1,Tree(:).Level);
L = max(Levels)-1;
[flag,ParTable] = issquare(Tree);
N = min(ParTable(:,2));
IndNodes = find(cat(1,Tree(:).IsLeaf)==0);

while ~flag & L>1

   Levels = cat(1,Tree(:).Level);
   IndNodes = find(cat(1,Tree(:).IsLeaf)==0);
   ParsToBeReduced = ParTable(find(ParTable(:,2)>N),1);
   I = intersect(find(Levels==L),IndNodes);
   for k=1:length(I)
      if any(ParsToBeReduced == Tree(I(k)).Parameter)
         Tree = node2leaf(Tree,I(k),T);
      end
   end
   [flag,ParTable] = issquare(Tree);
   L = L - 1;
```

```
end
if nargout>1
   Success = issquare(Tree);
End
```

****

```
function [LogicConstant,ParTable] = issquare(Tree)
%
% [LogicConstant,ParTable] = issquare(Tree)
%
% Tree: decision tree (see ID3n3)
%
% LogicConstant: 1 if all parameters used in Tree are associated to
%                the same number of nodes.
% ParTable: array(N x 2), N is the number of parameters used in Tree;
%          the first column indicates the parameter idexes (columns
%          of the matrix X used to grow the tree). The second column
%          specifies how many times each parameter is used in Tree.
%

ParTable = parinfo(Tree);
LogicConstant = all(ParTable(:,2)==ParTable(1,2));

% ------------------------------------------------------------
function ParTable = parinfo(Tree)
% ParTable = parinfo(Tree)

ParVec = cat(1,Tree(:).Parameter);
ParC = indexsubsets(ParVec);
Parameters = unique(ParVec);
Nparameters = length(Parameters);
ParTable = zeros(Nparameters,2);
for k=1:Nparameters
   ParTable(k,:) = [Parameters(k),length(ParC{k})];
end
```

****

```
function T = desumetargets(Tree)
% T = desumetargets(Tree)
%

Leafs = find(cat(1,Tree(:).IsLeaf));
T = [];
for k=1:length(Leafs)
   Ik = Tree(Leafs(k)).AssociatedRecords;
   T(Ik) = Tree(Leafs(k)).AssociatedTarget;
End
```

****

```
function mostra(Tree)
%
% mostra(Tree)
%

parents = cat(2,Tree(:).ParentNode);
leafs = cat(2,Tree(:).IsLeaf);
```

```
[x,y,h,s] = treelayout(parents);

figure
treeplot(parents)
H = get(gca,'Children');
set(H(2),'MarkerSize',10,'LineWidth',2);

for k=1:length(Tree)
   if leafs(k)
      target = Tree(k).AssociatedTarget;
      strk = sprintf('c=%d',target);
      text(x(k)-0.022,y(k)-0.04,strk)
   else
      parameter = Tree(k).Parameter;
      threshold = Tree(k).Threshold;
      strk = sprintf('P=%d \n th=%1.2f',parameter,threshold);
      text(x(k)+0.013,y(k),strk)
   end
end
title(strcat('Layout of tree "',inputname(1),""))
```

****

```
function [p,Icatched,Imissed] = msclbias(Tree,T)
%
% [p,Icatched,Imissed] = msclbias(Tree,T)
% (misclassification bias)
%
% Tree: decision tree.
% T: target vector of the original dataset.
%
% p: ratio between the number of miscalssified examples and
%    the total number of examples in the original dataset.
% Icatched: indexes of records assigned by Tree to the output cateories specified by T.
% Imissed: indexes of records assigned by Tree to the output cateories
%          other than those specified by T.
% Note that function 'dataseteval' also provides the number of
% misclassified recors, but refers to a generic dataset; instead,
% msclbias refers to the training dataset.
%

IndLeafs = find(cat(1,Tree(:).IsLeaf));
N = length(T);
Teval = zeros(N,1);
for k=1:length(IndLeafs)
   I = Tree(IndLeafs(k)).AssociatedRecords;
   Teval(I) = Tree(IndLeafs(k)).AssociatedTarget;
end
ImissBool = (T~=Teval);
Nmissed = sum(ImissBool);
p = Nmissed / N;
if nargout>1
   Imissed = find(ImissBool);
   Icatched = find(not(ImissBool));
End
```

****

```
function Tree = node2leaf(Tree,N,T)
```

```
% Tree = node2leaf(Tree,N,T)
%
if nargin<3
   T = desumetargets(Tree);
end
Children = find(cat(1,Tree(:).ParentNode)==N);
if all(cat(1,Tree(Children).IsLeaf))
   I = [];
   for k=1:length(Children)
      I = cat(1,I,Tree(Children(k)).AssociatedRecords);
   end
   Tree(N).AssociatedRecords = sort(I);
   Tree(N).AssociatedTarget = selecttarget(T,T(sort(I)));
   Tree(N).IsLeaf = 1;
   Tree(N).Parameter = [];
   Tree(N).Threshold = [];
   Tree(Children) = [];
   for k=1:length(Tree)
      x = Tree(k).ParentNode;
      if x>N
         Tree(k).ParentNode = x - 2;
      end
   end
end
```

****

```
function [Target,LeafResultS] = selecttarget(T,Tcurrent)
%
% [Target,LeafResultS] = selecttarget(T,Tcurrent)
%
% T: target vector relevant to the original dataset.
% Tcurrent: Target vector relevant to the current sub-set
%          (at a given step of the tree construction).
%
% Target: output category that would be assigned to the current
%        node in case of pre-pruning, on the basis of the
%        minimization of misclassified examples.
% LeafResultS: array (c x 2), where c is the number of different
%             output categories present in Tcurrent. The second
%             column contains the c output categories; the first
%             column contains the ratio between the number of examples
%             belonging to that category in Tcurrent and the number
%             of examples belonging to the same category in T.
%

CT = indexsubsets(Tcurrent);
c = length(CT);
LeafResult = zeros(c,2);
for k=1:c
   Tk = Tcurrent(CT{k});
   tk = unique(Tk);
   nk = sum(T==tk);
   LeafResult(k,:) = [length(Tk)/nk,tk];
end
LeafResultS = sortrows(LeafResult,1);
Target = LeafResultS(c,2);
```

****

```
function [Parameters,I] = usedpars(Tree)
%
% [Parameters,I] = usedpars(Tree)
%
% Tree: decision tree.
% Parameters: vector (p x 1); contains indexes of the parameters used
%             in Tree, referring to training dataset; it is sorted
%             in ascending order.
% I: vector (N x 1), where N is the total number of Tree nodes (N = length(Tree));
%    the k-th element of I returns the parameter used in the k-th node, returns
%    0 if the k-th node is a leaf.
%

I = zeros(length(Tree),1);
for k=1:length(Tree)
   park = Tree(k).Parameter;
   if ~isempty(park)
      I(k) = park;
   end
end
Xpar2 = I(find(I));
Parameters = unique(Xpar2);


                                        ************************

function [TreeSeeds,Mref,VR,RuleSeeds] = tree2rules(Tree,Limits,T)
%
% [TreeSeeds,Mref,VR,RuleSeeds] = tree2rules(Tree,Limits,T)
%
% ( Conversion of a decision tree into a rule set, developing of a rule matrix,
%   fuzzyfication of the discrete input space, by Marco Conti, Bologna, 2003 )
%
% Tree: input decision tree.
% Limits: array (p x 2), where p is the number of parameters used in Tree (see function 'usedpars');
%         Limits(k,1) and Limits(k,2) represent the lower and upper limit,
%         respectively, for the k-th parameter.
% T: vector (N x 1), where N is the number of examples (records) used to train Tree;
%    it is the target vector for the training dataset. If T is not specified, it is
%    set equal to the target vector associated to Tree (see function 'desumetargets'),
%    that is, the outputs that Tree would return if run on the training dataset
%    (note that Tree may be pruned).
%
% TreeSeeds: scalar structure with fields
%           - 'Parameters': vector (p x 1) of parameters used in Tree, as (column)
%                           indexes which refer to the trainind dataset.
%           - 'CrispReferences': cell (p x 1); its k-th element is a vector (1 x fk+2), containing the
%                               crisp references for the k-th parameter (fk is the number of nodes in Tree
%                               which use the k-th parameter).
%           - 'Rules': cell (Nrules x 1), its k-th element is a cell (p x 1), and is pertinent to the k-th rule;
%                      the i-th element of such a sub-cell is a row vector (1 x fi), where fi is the number of
%                      sets for the i-th parameter, and contains 1 in correspondence of the sets used in that
%                      rule for that parameter, 0 otherwise.
%           - 'AssociatedTargets': vector (Nrules x 1); its k-th element is the value of target
%                                  (output category index) associated to the k-th rule.
% Mref: cell array (p x 1); its k-th element is a row vector of fi elements, that is,
%       the fi fuzzy references for the k-th parameter (fi = 2*(fk+1), fk+1 is the number of (fuzzy sets)).
% VR: array (Ncategories,prod(F)); prod(F) is the total number of elementary status provided by
%     the partition of each parameter in (fuzzy) sets. The k-th row of VR indicates
%     the vector of membership to the output categories assigned to that elementary status.
% RuleSeeds: cell vector (Nrules x 1); the k-th element is pertinent to the k-th rule.
%           Each sub-cell is a structure with fields
```

```
%          - OutputCategory: vector (1 x Ncategories); the k-th element is 1 if the k-th category is associated
%                       to that rule, 0 otherwise
%          - conditions: cell (p x p); each element either is empty or is a structure with fileds
%                              - Parameter: scalar indicating the parameter (j-th) to which the
%                                      condition belongs (indexes the vector of used parameters);
%                              - FuzzySets: vector (1 x fj), where fj is the number of fuzzy sets
%                                      of the j-th parameter; contains indexes of the fuzzy sets
%                                      used for the j-th parameter in the k-th rule.
%          Elements of the sub-cell that are in the same row represent conditions to be put in OR;
%          Elements of the sub-cell that are in different rows represent conditions to be put in AND;
%          OR operators shall be evaluated first, then AND operators.
%
% See also: ID3n3  CEn2  plotfuzzyref  translatetseeds  tseeds2rseeds  IBLn1  Fuzzyengine2
%

if nargin==2
   T = desumetargets(Tree);
elseif ne(nargin,3)
   disp('Warning: wrong number of inputs');
   help tree2rules
   return
end

[Parameters,Xpar] = usedpars(Tree); % call usedpars
p = length(Parameters);

AssociatedTargets = cat(1,Tree(:).AssociatedTarget);
Ncategories = length(unique(AssociatedTargets));

Xleaf = cat(1,Tree(:).IsLeaf);
IndLeafs = find(Xleaf);
IndNodes = setdiff([1:length(Tree)],IndLeafs);
Nrules = sum(Xleaf);
UP = {p,Parameters,Xpar,Nrules,IndLeafs};

[F,Mrefcrisp,RefInfoC,RulesInfo] = crisprefinfo(Tree,T,Limits,UP); % call crisprefinfo
TreeSeeds = struct('Parameters',[],'CrispReferences',[],'Rules',[],'AssociatedTargets',[]);
TreeSeeds.Parameters = Parameters;
TreeSeeds.CrispReferences = Mrefcrisp;
TreeSeeds.Rules = RulesInfo;
TreeSeeds.AssociatedTargets = AssociatedTargets;

N = length(T);
Nmissed = msclbias(Tree,T) * N; % call msclbias
if Nmissed==0 & nargout>1
   disp('Warning: fuzzyfication is not expected when tree efficiency equals 1')
end
FD = Nmissed / N; % set FuzzynessDegree
% Observe: it may be desirable to have FD higher for smaller values of N (and always > 0),
% e.g. FD = (Nmissed/N) + N^k , with k<0

if nargout>1
   [Mref,Fcheck] = findfuzzyref(Mrefcrisp,RefInfoC,FD); % call findfuzzyref
   if nargout>2
      VR = dorulematrix(F,Ncategories,Nrules,AssociatedTargets,RulesInfo,p); % call dorulematrix
      if nargout>3
         RuleSeeds = tseeds2rseeds(TreeSeeds); % call tseeds2rseeds
      end
   end
end
```

```
% ----- END TREE2RULES ----------------------------------------------------------------
% ----- TREE2RULES PRIVAT FUNCTIONS: --------------------------------------------------

function [f,Fcrisp,RefInfoC,RulesInfo] = crisprefinfo(Tree,T,Limits,UP)
% [F,Mrefcrisp,RefInfoC,RulesInfo] = crisprefinfo(Tree,T,Limits,{p,Parameters,Xpar,Nrules,IndLeafs})
%
% F: vector (1 x p); its k-th element indicates the number of sets (fk) for the k-th parameter.
% Mrefcrisp: cell(p x 1); its k-th element contains the fk+1 crisp references for the k-th parameter.
% RefInfoC: cell (p x 1); each element is a structure (1 x f), with fields 'Catched', 'Missed';
%           each scalar structure contains the elements (not the indexes) of T assigned to that target,
%           through a rule which involves that set (structure index) of that parameter (cell index).
% RulesInfo: cell(Nrules x 1), its k-th element is a cell (p x 1), and is pertinent to the k-th rule;
%            the i-th element of such a sub-cell is a row vector (1 x fi), where fi is the number of
%            sets for the i-th parameter, and contains 1 in correspondence of the sets used in that
%            rule for that parameter, 0 otherwise.
%
[p,Parameters,Xpar,Nrules,IndLeafs] = deal(UP{:});
[f,Fcrisp,M2,MCheck,RefInfoC] = deal(cell(p,1));
f = zeros(1,p);
RulesInfo = cell(Nrules,1);
for k=1:p
   I = find(Xpar==Parameters(k));
   fk = length(I);
   f(k) = fk+1;
   x = zeros(1,fk);
   for s=1:fk
      x(s) = Tree(I(s)).Threshold;
   end
   x2 = cat(2,Limits(k,1),sort(x),Limits(k,2));
   Fcrisp(k) = {x2};
   M2(k) = {( x2(1:end-1) + x2(2:end) )/2};
   MCheck(k) = {ones(1,f(k))};
   RefInfoS = struct('Catched',[],'Missed',[]);
   RefInfoS(1:fk+1) = RefInfoS;
   RefInfoC(k) = {RefInfoS};
end
for k=1:Nrules
   v = Tree(IndLeafs(k)).AssociatedRecords;
   Tk = T(v);
   Target = Tree(IndLeafs(k)).AssociatedTarget;
   MCheck2 = MCheck;
   N = IndLeafs(k);
   while N>1
      side = Tree(N).ParentBranch;
      N = Tree(N).ParentNode;
      Threshold = Tree(N).Threshold;
      Parameter = Tree(N).Parameter;
      pI = find(Parameters==Parameter);
      switch side
      case 1
         MCheck2(pI) = {MCheck2{pI}.*(M2{pI} > Threshold)};
      case 2
         MCheck2(pI) = {MCheck2{pI}.*(M2{pI} <= Threshold)};
      end
   end
   RulesInfo(k) = {MCheck2};
   for s=1:p
      RefInfo = RefInfoC{s};
      I = find(MCheck2{s});
      for z=1:length(I)
         Catched = RefInfo(I(z)).Catched;
         Missed = RefInfo(I(z)).Missed;
```

```
            RefInfo(I(z)).Catched = cat(1,Catched,Tk(find(Tk==Target)));
            RefInfo(I(z)).Missed = cat(1,Missed,Tk(find(Tk~=Target)));
        end
        RefInfoC(s) = {RefInfo};
    end
end

% ------------------------------------------------------------------------------
function [Mref,Fcheck] = findfuzzyref(Fcrisp,RefInfoC,FD)
% [Mref,Fcheck] = findfuzzyref(Fcrisp,RefInfoC,FD)
%
% Fcheck: array (p x 1); its k-th element is 1 if the correspondentfuzzy
%         references are in ascending order, 0 otherwise.
%
Mref = cell(size(Fcrisp));


for p=1:length(Fcrisp)

    Fcrispk = Fcrisp{p};
    Fcheck = zeros(size(Fcrisp));
    RefInfo = RefInfoC{p};
    Lmin = Fcrispk(1);
    Lmax = Fcrispk(end);
    Ip = Lmax - Lmin;
    X = Fcrispk(2:end-1);
    f = length(Fcrispk) - 1; % number of fuzzy sets
    M = zeros(1,f); % number of misclassified for each set
    for k=1:f
        M(k) = length(RefInfo(k).Missed);
    end
    M = M + eps; % avoid zeros
    R = zeros(1,2*f); % vector of fuzzy references - initialization
    R(1) = Lmin; R(end) = Lmax;
    Na = 2*(f-1);
    AC = zeros(Na); % matrix of coefficients - initialization
    b = zeros(Na,1); % column of constants - initialization

    if FD>0
        for k=1:f-1
            a = zeros(1,Na);
            a(2*k) = 1; a(2*k-1) = 1;
            row = 2*k-1;
            AC(row,:) = a;
            b(row) = ((M(k)+M(k+1))/(2*sum(M)-M(1)-M(end)))*Ip*FD; % potrebbe essere 0? NO!
            row = 2*k;
            a = zeros(1,Na);
            a(2*k-1) = 1; a(2*k) = -M(k)/M(k+1);
            AC(row,:) = a;
        end
        ap = AC\b; % solve linear equation system
    elseif FD==0
        ap = zeros(1,Na);
    end

    t = zeros(1,Na);
    for k=1:Na
        z = ceil(k/2);
        s = sign(0.5 - rem(k,2));
        t(k) = X(z) + s*ap(k);
    end
    R(2:end-1) = t;
```

```
   Mref(p) = {R};
   Fcheck(p) = isequal(t,sort(t)); % che cazzo faccio se non e' vero?

end

%---------------------------------------------------------------------
function VR = dorulematrix(F,Ncategories,Nrules,AssociatedTargets,RulesInfo,p)
% VR = dorulematrix(F,Ncategories,Nrules,AssociatedTargets,RulesInfo,p)
%
VR = zeros(prod(F),Ncategories);
M = zeros(F);
for k=1:Nrules
   D.type = '()';
   coords = cell(1,p);
   Rulek = RulesInfo{k};
   for s=1:p
      cond_s = Rulek{s};
      coords(s) = {find(cond_s)};
   end
   D.subs = coords;
   M1 = subsasgn(M,D,1);
   index = find(M1);
   category = AssociatedTargets(k);
   VR(index,category) = 1;
End
```

****

```
function plotfuzzyref(refvec,ParStr)
% plotfuzzyref(refvec)
%

f = length(refvec)/2; % number of fuzzy sets
X = zeros(f,2*f);
for k=1:f
   X(k,[2*k-1 , 2*k]) = 1;
end
colours = 'brygmpck';
Xticks = zeros(2*f,1);
for k=1:2*f
   z = sprintf('%3.2f',refvec(k));
   Xticks(k) = str2num(z);
end
figure
set(gca,'NextPlot','Add','LineWidth',1);
for k=1:f
            plot(refvec,X(k,:),colours(k))
end
H = get(gca,'Children');
set(H,'LineWidth',2);
set(gca,'NextPlot','Replace','XTick',Xticks,'YTick',[0 0.5 1]);
axis([min(refvec) max(refvec) 0 1.2])
if nargin>1
   title(strcat('Fuzzy references for parameter ',ParStr))
end
```

****

```
function Rstr = translatetseeds(TreeSeeds,ParStr,CategoriesStr)
%
% Rstr = translatetseeds(TreeSeeds,ParStr,CategoriesStr)
%
% TreeSeeds: summary of the results of a decision Tree (see tree2rules).
% ParStr: cell of strings (p x 1) containing parameters names (delault is
%         'Par#1','Par#2',...)
% CategoriesStr: cell of strings (Ncategories x 1) containing output categories
%                 names (delault is 'Category#1','Category#2',...)
%
% Rstr: cell array of strings (Nrules x 1), providing a literary expression for
%       rules specified in TreeSeeds. Each rule would be like
%       IF { ( Par#1: Set2 ) AND ( Par#2: Set3 ) AND ( Par#3: Set1 or Set2 ) }==> Category#3
%

Parameters = TreeSeeds.Parameters;
p = length(Parameters);
Rules = TreeSeeds.Rules;
Nrules = length(Rules);
Mrefcrisp = TreeSeeds.CrispReferences;
F = zeros(p,1);
for k=1:p
   F(k) = length(Mrefcrisp{k})-1;
end
AssociatedTargets = TreeSeeds.AssociatedTargets;
Ncategories = length(unique(AssociatedTargets));

if nargin<3
   CategoriesStr = cell(1,Ncategories);
   for k=1:Ncategories
      CategoriesStr(k) = {sprintf('Category#%d',k)};
   end
   if nargin<2
      ParStr = cell(1,p);
      for k=1:p
         ParStr(k) = {sprintf('Par#%d',k)};
      end
   end
end

Rstr = cell(Nrules,1);
Sincipit = 'IF { '; Send = ' }';
for k=1:Nrules
   rkStr = Sincipit;
   Rule_k = Rules{k};
   nr = length(Rule_k);
   for z=1:nr
      condkz = find(Rule_k{z});
      Skz = getconditionstr(condkz,ParStr,z);
      rkStr = [rkStr Skz];
      if z<nr
         rkStr = [rkStr ' AND '];
      else
         rkStr = [rkStr Send];
      end
   end
   rkStr = [rkStr ' ==> ',CategoriesStr{AssociatedTargets(k)}];
   Rstr(k) = {rkStr};
end


% --------------------------------------------------------------------
```

```
function Skz = getconditionstr(condkz,ParStr,z)
% Skz = getconditionstr(condkz,z)
Skz = ['( ',ParStr{z},': '];
n = length(condkz);
for k=1:n
   s = sprintf('Set%d',condkz(k));
   Skz = [Skz s];
   if k<n
      Skz = [Skz ' or '];
   else
      Skz = [Skz ' )'];
   end
end
```

****

```
function Mseeds = tseeds2rseeds(TreeSeeds)
%
% Mseeds = tseeds2rseeds(TreeSeeds)
%
% TreeSeeds: scalar structure with fields
%          - 'Parameters': vector (p x 1) of parameters used in Tree, contains
%                    indexes which refer to the trainind dataset.
%          - 'CrispReferences': cell (p x 1); its k-th element contains the
%                          fk+1 crisp references for the k-th parameter.
%          - 'Rules': cell (Nrules x 1), its k-th element is a cell (p x 1), and is pertinent to the k-th rule;
%                    the i-th element of such a sub-cell is a row vector (1 x fi), where fi is the number of
%                    sets for the i-th parameter, and contains 1 in correspondence of the sets used in that
%                    rule for that parameter, 0 otherwise.
%          - 'AssociatedTargets': vector (Nrules x 1); its k-th element is the value of target
%                          (output category index) associated to the k-th rule.
%
% Mseeds: cell vector (Nrules x 1); the k-th element is pertinent to the k-th rule.
%        Each sub-cell is a structure with fields
%        - OutputCategory: vector (1 x Ncategories); the k-th element is 1 if the k-th category is associated
%                    to that rule, 0 otherwise
%        - conditions: cell (p x p); each element either is empty or is a structure with fileds
%                          - Parameter: indiating the parameter (j-th) to which the condition belongs
%                          - FuzzySets: vector (1 x fj), where fj is the number of fuzzy sets
%                                  of the j-th parameter; contains indexes of the fuzzy sets
%                                  used for the j-th parameter in the k-th rule.
%        Elements of the sub-cell that are in the same row represent conditions to be put in OR;
%        Elements of the sub-cell that are in different rows represent conditions to be put in AND;
%        OR operators shall be evaluated first, then AND operators.
%

Parameters = TreeSeeds.Parameters;
p = length(Parameters);
Rules = TreeSeeds.Rules;
Nrules = length(Rules);
Mrefcrisp = TreeSeeds.CrispReferences;
F = zeros(p,1);
for k=1:p
   F(k) = length(Mrefcrisp{k})-1;
end
AssociatedTargets = TreeSeeds.AssociatedTargets;
Mout = zeros(Nrules,length(unique(AssociatedTargets)));
for k=1:Nrules
   Mout(k,AssociatedTargets(k)) = 1;
end
Mseeds = cell(Nrules,1);
```

```
for k=1:Nrules

   seed = struct('conditions',[],'OutputCategory',[]);
   [CC,DD] = deal(cell(p,p));
   Rule_k = Rules{k};

   for s=1:length(Rule_k)
      Z = struct('Parameter',[],'FuzzySets',[]);
      cond = find(Rule_k{s});
      Z.Parameter = s; % Parameters(s) se volessi rispettare
      % l'ordine dei parametri del training dataset.
      Z.FuzzySets = cond;
      DD(s,1) = {Z}; % colonna sempre 1 perche' tutte le
       % condizioni sono in AND
   end

   seed.OutputCategory = Mout(k,:);
   seed.conditions = DD;
   Mseeds(k) = {seed};

End
```

CONCEPT LEARNING

```
function [LearnedConcepts,Xd] = CEn2(X,T,Cref)
%
% [LearnedConcepts,Xd] = CEn2(X,T,Cref)
%
% ( Candidate Elimination algorithm modified by Marco Conti 21.01.04 )
% This algorithm induces a partial ordering (from specific to general) in the
% hypotheses space. Let p be the number of input parameters (size(X,2)), and f a vector (1xp),
% such that the k-th parameter is discretized in f(k) sets. Therefore, the input space
% consists of a number D=prod(f) of elementary sub-domains. A hypothesis is a
% sub-set of the input space, associated to an output category. A Sub-set of the input space is
% represented as a cell array (1 x p), such that its k-th element is a boolean vector (1 x f(k)).
% CEn2 returns, for each output category, two hypotheses, constituiting necessary
% and sufficient conditions to identify such a category. The necessary condition
% is a hypothesis consistent with all positive examples of that category (but not
% necessarily consistent with the negative ones); the sufficient condition is
% a hypothesis consistent with all negative examples of that category and obtained
% through a minimal generalization process from positive examples. Therefore, in
% a generic case (partial convergence) the necessary condition includes the sufficient
% one (it is more general); in case of convergence the two conditions coincide.
% CEn2 also returs, for each output category, a list of examples which should be
% excluded (as outliers) to have a total or partial convergence.
%
% X: array (N x p), set of N training instances (examples), each one containing
%    values of p attributes.
% T: vector (N x 1), target vector. T(j) belongs to {1,...,Nout}, with
%    Nout number of output categories.
% Cref: cell array (p x 1). Cref{k} is a vector (1 x f(k)+1), which specifies the (crisp) references
%       to discretize the examples contained in X --> see TreeSeeds obtained by means of function tree2rules.
%
% LearnedConcepts: structure (1 x Nout), (note: Nout = length(unique(T))) with fields:
%                     - AssociatedTarget: specifies which output category is associated to the current element.
%                     - VersionSpace: structure (1x1), with fields:
%                                 o NecessaryCondition: hypothesis correspondent to the necessary condition
%                                                       for current category;
%                                 o SufficientCondition: hypothesis correspondent to the sufficient condition
%                                                        for current category;
%                     - CrispReferences: Cref
%                     - Outliers: vector of indexes specifying the examples which should be excluded to obtain
%                               a total or partial convergence.
%
% Xd: cell array (N x 1), containing the the input vectors (rows of X) discretized according to Cref,
%    i.e. represented as instances of the discretized input space. Each element of Xd is a cell array (1 x p),
%    the k-th element of such a cell array is a boolean vector (1 x f(k)); note that each one of these boolean
%    vectors has one element equal to 1 and all others equal to 0.
%
% See also: translatecondition tree2rules ID3n3 Fuzzyengine2 discretizeparvec comparehypotheses isconsistent
%          generatesubsets generalizehyp
%

if ne(nargin,3)
   disp('Wrong number of inputs')
   help CEn1
   return
end
[N,p] = size(X);
if ne(length(Cref),p)
   disp('Dimensions of X and Cref are not consistent')
```

```
    help CEn2
    return
end
F = zeros(1,p);
Gmax = cell(1,p);
for k=1:p
    dims = length(Cref{k})-1;
    F(k) = dims;
    Gmax(k) = {ones(1,dims)};
end
Gmax = {Gmax};
hyp = cell(1,p);

Xd = cell(N,1);
for k=1:N
    Xd(k) = {discretizeparvec(X(k,:),Cref)}; % call discretizeparvec
end

categories = unique(T);
Ncategories = length(categories);
LearnedConcepts = struct('AssociatedTarget',[],'VersionSpace',[],...
    'CrispReferences',[],'Outliers',[]);

for k=1:Ncategories

    category = categories(k);
    OutliersVec = [];
    Tpos = find(T==category);
    Tneg = find(T~=category);
    Tk = {Tpos, Tneg};
    Scell = cell(2,1);

    for z=[1 2]

        Tkz = Tk{z};
        N = length(Tkz);

        for r=1:N
            d = Xd{Tkz(r)};
            if r==1
                S = {d}; % initialization
            else
                Sc = generalizeS(S,d); % call generalizeS
                Sc = rmhyp(Sc,-1); % call rmhyp
                if isequal(Sc,Gmax) | isempty(Sc{1})
                    OutliersVec = cat(1,OutliersVec,r);
                else
                    S = Sc;
                end
            end
        end
        Scell(z) = {S};

    end

    CondNecess = Scell{1};
    CondNecessNeg = Scell{2};
    CN = CondNecess{1};
    CNN = CondNecessNeg{1};
    CondSuff = cell(size(CN));
    for z=1:length(CN);
        CondSuff(z) = { CN{z} & ( xor(CN{z},CNN{z}) ) };
```

```
    end

    VH = struct('NecessaryCondition',{CN},'SufficientCondition',{CondSuff});
    LearnedConcepts(k).AssociatedTarget = category;
    LearnedConcepts(k).VersionSpace = VH;
    LearnedConcepts(k).CrispReferences = Cref;
    LearnedConcepts(k).Outliers = OutliersVec;

end



% ------------------------------------------------------------------------------------------
function S = generalizeS(S,d)
% S = generalizeS(S,d,G)
ns = size(S,1);
for k=1:ns
    hs = S{k};
    if not(isconsistent(hs,d,1)) % 1 => positive example
        hs2 = generalizehyp(hs,d);
        S(k) = {hs2}; % hs is single hyp
    end
end

% ------------------------------------------------------------------------------------------
function H = rmhyp(H,segno)
%H = rmhyp(H,segno)
if size(H,1)==1; return; end;
[I,It] = deal([1:size(H,1)]);
flag = 1;
while flag
    h1 = H{I(1)};
    for k=2:length(I)
        hk = H{I(k)};
        CMP = comparehypotheses(h1,hk);
        if not(isempty(CMP))
            switch CMP
            case segno % segno=1 => tengo l'ipotesi piu' generale
                %It(I(k)) = [];
                It = setdiff(It,I(k));
            case -segno
                %It(I(1)) = [];
                It = setdiff(It,I(1));
            end
        end
    end
    I(1) = [];
    if length(I)==1 | length(It)==1
        flag = 0;
    end
end
H = H(It);



                                        ************************

function xD = discretizepar(x,Vref)
%
% xD = discretizepar(x,Vref)
%
% x: scalar value
```

```
% Vref: vector (1 x n), sorted in ascending order.
% xD: logical vector (1 x n-1); xD(k) is 1 if x <= Vref(k+1)
%      and >= Vref(k), zero otherwise.
%      xD is all zeros if x is out of range.
%

if ne(nargin,2)
   disp('Wrong number of inputs')
   help discretizepar
   return
end

xD = ( x <= Vref(2:end) ) .* ( x > Vref(1:end-1) );
if x==Vref(1)
   xD(1) = 1;
end
```

****

```
function vxD = discretizeparvec(vx,Cref)
%
% vxD = discretizeparvec(vx,Cref)
%
% vx: vector (1 x p)
% Cref: cell vector (p x 1); the k-th cell contains a vector (1 x nk)
%       sorted in ascending order. Cref: matrix of crisp references.
% vxD: cell vector (p x 1); the k-th cell contains a logical vector xDk (1 x nk-1); being Vrk=Cref{k},
%      xDk(j) is 1 if vx(k) <= Vrk(j+1) and >= Vrk(j), zero otherwise (see function discretizepar).
%

if ne(nargin,2)
   disp('Wrong number of inputs')
   help discretizeparvec
   return
end
p = length(Cref);
if ne(length(vx),p)
   disp('Wrong input assignment: z and Cref must have the same lengtht')
   help discretizeparvec
   return
end

vxD = cell(1,p);
for k=1:p
   vxD(k) = {discretizepar(vx(k),Cref{k})};
end
```

****

```
function [Rstr,index] = translatecondition(Condition,ParStr,CategoriesStr)
%
% Rstr = translatecondition(Condition)
%
%
% Rstr: string providing a literary expression for
%       rules specified in TreeSeeds. Each rule would be like
%       IF { ( Par#1: Set2 ) AND ( Par#2: Set3 ) AND ( Par#3: Set1 or Set2 ) }==> Category#3
%
% index: index vector of activated subdomains (status)
```

```
%

p = length(Condition);
if nargin<3
   CategoriesStr = 'Category#x';
   if nargin<2
      ParStr = cell(1,p);
      for k=1:p
         ParStr(k) = {sprintf('Par#%d',k)};
      end
   end
end

Sincipit = 'IF { '; Send = ' }';
rkStr = Sincipit;
F = zeros(1,p);
for z=1:p
   L = Condition{z};
   F(z) = length(L);
   condkz = find(L);
   if not(all(L) | all(~L))
       Skz = getconditionstr(condkz,ParStr,z);
       rkStr = [rkStr Skz];
       if z<p
          rkStr = [rkStr ' AND '];
       else
          rkStr = [rkStr Send];
       end
   end
end
Rstr = [rkStr ' ==> ',CategoriesStr];
M = zeros(F);
D.type = '()';
coords = cell(1,p);
for s=1:p
   cond_s = Condition{s};
   if sum(cond_s)==0; cond_s = ~cond_s; end;
   coords(s) = {find(cond_s)};
end
D.subs = coords;
M1 = subsasgn(M,D,1);
index = find(M1);



% ------------------------------------------------------------------
function Skz = getconditionstr(condkz,ParStr,z)
% Skz = getconditionstr(condkz,z)
Skz = ['( ',ParStr{z},': '];
n = length(condkz);
for k=1:n
   s = sprintf('Set%d',condkz(k));
   Skz = [Skz s];
   if k<n
      Skz = [Skz ' or '];
   else
      Skz = [Skz ' )'];
   end
end
```

****

```
function flag = comparehypotheses(h1,h2)
%
% flag = comparehypotheses(h1,h2)
%
% h1, h2: cell arrays of the same length (n); their correspondent elements
%         must be logical vectors of the same length (statements).
% flag: 0 if all correspondent statements are the same;
%       1 if none of h1 statements if more specific then the correspondent h2 statement,
%       and at least one is more general;
%       -1 if none of h1 statements if more general then the correspondent h2 statement,
%       and at least one is more specific;
%       [] if at least two correspondent statements are non comparable.
% flag 0  ==> equal hypotheses
% flag 1  ==> hypothesis 1 more general than hypothesis 2
% flag -1 ==> hypothesis 1 more specific than hypothesis 2
% flag [] ==> hypotheses are not comparable
%

flag = [];
n = length(h1);
vc = [];
for k=1:n
   vc = cat(2,vc,comparestatements(h1{k},h2{k}));
end
if length(vc)==n
   if not(any(vc))
      flag = 0;
   elseif all(vc>=0)
      flag = 1;
   elseif all(vc<=0)
      flag = -1;
   end
end
```

****

```
function flag = comparestatements(s1,s2)
%
% flag = comparestatements(s1,s2)
%
% s1 , s2: logical vectors of the same length
% flag: 0 if s1(k)=s2(k), otherwise 1 if s1(k)>=s2(k),
%       -1 if s1(k)<=s2(k) for every k,
%       otherwise flag is emptymatrix.
% flag 0  ==> equal statements
% flag 1  ==> statement 1 more general than statement 2
% flag -1 ==> statement 1 more specific than statement 2
% flag [] ==> statements non comparable
%

if isequal(s1,s2)
   flag = 0;
elseif  all(s1>=s2)
   flag = 1;
elseif all(s1<=s2)
   flag = -1;
else
   flag = [];
```

```
end
```

****

```
function Hs = generalizehyp(h,d,G)
%
% Hs = generalizehyp(h,d,G)
%
%

nc = length(h);
hs = cell(1,nc);
for k=1:nc
   hs(k) = { h{k} | d{k}};
end
if nargin==2 | isok(hs,G)
   Hs = hs;
else
   Hs = [];
end


% ------------------------------------------------
function MoreGeneral = isok(hs,G)
ng = size(G,1);
flag = 1;
MoreGeneral = 0;
m = 1;
while flag
   q = comparehypotheses(G{m},hs);
   if not(isempty(q)) & q==1
      flag = 0;
      MoreGeneral = 1;
   elseif m==ng
      flag = 0;
   end
   m = m + 1;
end
```

****

```
function SubSets = generatesubsets(F)
%
% SubSets = generatesubsets(F)
%
% F: vector (1 x n) of natural numbers.
% SubSets: cell vector (1 x n); being fk=F(k), the k-th element of SubSets contains
%        a logical array (2^fk-1 x fk); the rows of these arrays represent
%        all possible sub-sets of fk elements, except the empty sub-set.
%
% Example: F = [3 4]; SubSets{1} will be A1: array (7 x 3);
%        let A1(j,:) be [1 0 1]; such a vector represents the
%        sub-set containing the first and the third element
%        out of a set of three elements.
%

n = length(F);
SubSets = cell(1,n);
for k=1:n
```

```
   T = ftabella2(F(k),2);
   SubSets(k) = {T(2:end,:)}; % exclude null hypotheses
End
```

****

```
function SubSets = generatesubsets(F)
%
% SubSets = generatesubsets(F)
%
% F: vector (1 x n) of natural numbers.
% SubSets: cell vector (1 x n); being fk=F(k), the k-th element of SubSets contains
%        a logical array (2^fk-1 x fk); the rows of these arrays represent
%        all possible sub-sets of fk elements, except the empty sub-set.
%
% Example: F = [3 4]; SubSets{1} will be A1: array (7 x 3);
%        let A1(j,:) be [1 0 1]; such a vector represents the
%        sub-set containing the first and the third element
%        out of a set of three elements.
%

n = length(F);
SubSets = cell(1,n);
for k=1:n
   T = ftabella2(F(k),2);
   SubSets(k) = {T(2:end,:)}; % exclude null hypotheses
End
```

****

```
function flag = isconsistent(h,d,t)
% flag = isconsistent(h,d,t)
%

c = comparehypotheses(h,d);
if not(isempty(c)) & c>=0
   incluso = 1;
else
   incluso = 0;
end
switch t
case 1
   flag = incluso;
case -1
   flag = ~incluso;
end
```

****

```
function H = rminconsistenthyps(H,d,t)
%
% H = rminconsistenthyps(H,d)
%

I = [];
for k=1:length(H)
   h = H{k};
```

```
    if isconsistent(h,d,t)
        I = cat(1,I,k);
    end
end
H(I) = [];
```

****

```
function Hg = specifyhyp(h,d,S,SpecifyToSingleHyp)
%
% Hg = specifyhyp(h,d,S,SpecifyToSingleHyp)
%
%

nc = length(h);
Hg = {};
for k=1:nc
    hg = h;
    hg(k) = { h{k} & (xor(h{k},d{k})) };
    if isok(hg,S)
        Hg = cat(1,Hg,{hg});
    end
end
Hg
if SpecifyToSingleHyp & not(isempty(Hg))
    nHg = size(Hg,1);
    L = zeros(nHg,nc);
    for k=1:nHg
        Hgk = Hg{k};
        for z=1:nc
            L(k,z) = length(Hgk{z});
        end
    end
    L = sum(L,2);
    ind = find(L==max(L));
    ind = ind(1);
    Hg = Hg{ind};
end

% ------------------------------------------------
function MoreSpecific = isok(hg,S)
ns = size(S,1);
flag = 1;
MoreSpecific = 0;
m = 1;
while flag
    q = comparehypotheses(S{m},hg);
    if not(isempty(q)) & q==-1
        flag = 0;
        MoreSpecific = 1;
    end
    if m==ns
        flag = 0;
    end
    m = m + 1;
end
```