Kernel Machines for Disruption Prediction and Novelty Detection at JET.

Advisor
Prof. Alessandra Fanni

PhD Dissertation of
Rita Sabrina Delogu
Acknowledgements

I would like to thank all the people that supported me to complete this dissertation.

I would particularly like to mention all those that had to read the whole draft and gave important suggestions and advices, Prof. Alessandra Fanni, Doct. Augusto Montisci, Doct. Maria Katiuscia Zedda.

Part of this dissertation has been completed at JET science centre (Culham, UK), thanks to all the people without their work this dissertation could no be completed.

I also thank my family and friends that with their patient have supported me during the last three years of my Ph.D.
I.1 Introduction

Chapter 1  Energy and Nuclear Fusion.................................................................3
References ........................................................................................................13

Chapter 2 - Disruptions prediction – State of the Art. ........................................14
References ........................................................................................................18

Chapter 3 - Kernel Machines – Conventional approaches ....................................20
References ........................................................................................................29

Chapter 4 - Kernel Machines – Geometrical Approach ........................................30
  4.3.1 Neural model formalization .................................................................33
  4.3.2 Neural Network Synthesis .................................................................34
  4.3.2.1 Input Space Mapping ................................................................34
  4.3.2.2 Theorem 1 – Input space mapping .............................................37
  4.3.2.3 Feature Space Mapping .............................................................39
  4.3.2.4 Generalization to K-classes problems .......................................42
  4.4 Results .....................................................................................................43
Reference ........................................................................................................51

Chapter 5  Novelty Detection ............................................................................53
References ........................................................................................................61

Chapter 6  Disruption prediction at JET .............................................................64
References ........................................................................................................70

Chapter 7  Disruption predictors .................................................................71
References ........................................................................................................81

Chapter 8  Conclusions and comments ............................................................82
I.1 Introduction

The interest on nuclear fusion devices as power generators has been growing since the last 25 years. As new competitors enter in the energy market and natural resources become not enough to satisfy the demand, the answer for the long term period is certainly the nuclear fusion.

Although none of the operating nuclear devices has produced energy, as they operate as experimental machines, it is fundamental to study and understand the process in the light of future developments like ITER.

One of the drawbacks of those experimental devices is the occurrence of disruptive events inside the plasma which pose serious limits to both machine's lifetime and its economical feasibility. During a disruption the plasma looses its confinement and the plasma-facing components are then subject to forces of several tons.

In order to avoid or to mitigate the disruptive events a number of neural techniques have been developed, in many of those techniques neural networks approach are used and they seems to be the most suitable to predict the event or, more precisely, to build an impending disruption warning indicator.

In this dissertation we try to give an overview on the techniques used to predict disruption in tokamak devices. An application of SVM and of a new approach, based on geometrical synthesis of MLP neural network, are presented and applied on a sub-set of the entire JET disruption database.

The aim prediction is then reached classifying patterns coming from JET database 100 ms before the disruption event in order to give the mitigation system enough time to operate.

An important aspect of neural networks to take into account is the aging of the net, once completely new patterns are presented to them, their performance decade miserably. Those new patterns can come from unexplored device's region, so a chance to overcome this drawback is to map those new regions with the technique of novelty detection.

This thesis is then organized as follows.

The Chapter 1 gives an overview of energy market and of the most important operating nuclear devices. In the Chapter 2 the state of the art of disruption prediction is given, describing what produced in literature up till now.

The Chapter 3 presents all the most important kernel machines used for classification and prediction purposes, while in the Chapter 4 a new approach based on geometrical synthesis of MLP neural network is presented and applied on some benchmarks present in literature.

The novelty detection state of the art is presented in the Chapter 5, where all the most important papers are illustrated.
The Chapter 6 gives information about the database used to train and test SVM and the geometrical approach, where all the signal used to build the database are machine independent.

An application of SVM and geometrical approach, on the database showed in the Chapter 6, is then given in the Chapter 7, where the performance are reported in terms of False Alarms and Missed Alarm.

The Chapter 8 reports the conclusion and further comments on performance achieved with the technique presented in this dissertation.
Chapter 1 Energy and Nuclear Fusion.

1.1 Introduction.

The global demand for energy continues to grow year by year as the world population expands and society becomes more and more dependent on energy supplies. The need to find new sources of energy becomes increasingly important as environmental concerns mount over the emission of CO$_2$ from burning fossil fuels [1].

The energy debate also should take into account that current energy demand is covered by 41% oil, 22% gas, 16% coal, 15% nuclear and 6% renewables, as we can see from Fig.1 [2]. If nothing is done, the total energy picture in 2030 will continue to be dominated by fossil fuels (see Fig.2).

![Energy Demand](image)

**Figure 1 - Current Energy Demand**

Hence, the emission of CO$_2$ represents a serious burden for all the world.
At the moment the scientist community is involved in finding different solutions to this problem, especially focusing on renewable energy, hydrogen systems and solar energy. Although renewable energy sources have the potential to play a great role, all the targets already achieved in energy production, around 30% over the last 15 years, are nothing in absolute terms. For what concern hydrogen systems all problems are related to storage and transportation, while solar plants have no supply problems but are not a valid option to generate a large amount of energy.

The option we are taking into account in this thesis is to generate electricity from nuclear fusion, a secure and everlasting source of energy. Of course this is not a short-term solution, all the forces and resources involved to develop a nuclear fusion plant are not even comparable to all the others considered up till now.
The next research reactor ITER (International Thermonuclear Experimental Reactor) could be operating by 2050, but even if it is a very long term plan it is worth to spend resources to develop something useful for future generations.

1.2 The nuclear fusion process.

The nuclear fusion process consists of forcing together two light nuclei (i.e. Hydrogen) and obtaining a combination whose mass will be less than the sum of the masses of the individual nuclei. The decrease in mass comes off in the form of energy according to the Einstein relationship [3].

In the 50’s some fusion reactions were individuated to be suitable to get a thermonuclear fusion. The most promising was the one that involved Deuterium and Tritium, isotope 2 and 3 of Hydrogen:

\[ _2^1 D + _1^3 T \rightarrow _2^4 He + n + 17.59 \text{MeV} \]  \hspace{1cm} (1.1)

Deuterium and Tritium are easy to get, Deuterium can be extracted from sea water by electrolysis, while Tritium is obtained by the fixing reaction of Lithium:

\[ _3^6 Li + n \rightarrow _1^3 T + _1^4 He + 4.8 \text{MeV} \] \hspace{1cm} (1.2)
\[ _3^6 Li + n \rightarrow _1^3 T + _1^4 He + n - 2.5 \text{MeV} \]
As we can see from eq. 1.1 and 1.2 a little amount of these fuels is able to generate an enormous quantity of energy.

In addition, to provide a sufficiently high temperature to enable the particles to overcome the Coulomb barrier, that temperature must be maintained for a sufficient confinement time and with a sufficient density in order to obtain a net yield of energy from a fusion reaction. The overall conditions, which must be met for a yield of more energy than is required for the heating of the plasma, are usually stated in terms of the product of ion density $n_e$, confinement time $\tau_E$ and temperature $T$, a condition called Lawson’s criterion [4]:

$$n_e T \tau_E \geq \frac{12 k_B T^2}{E_{ch} \langle \sigma v \rangle}$$

(1.3)

where $k_B$ is the Boltzmann constant, $\sigma$ is the fusion cross section, $v$ is the relative velocity, and $\langle \rangle$ denotes an average over the Maxwellian velocity distribution at the temperature $T$. $E_{ch}$ is the energy of the charged fusion products, which, for a D-T reaction, is about 3.5 MeV. When eq. 1.3 is verified then the ignition starts; to be more precise, the condition is reached under which a plasma can be maintained by fusion reactions without external energy input.

Confinement time in nuclear fusion experiments is defined as the time the plasma is maintained at a temperature above the critical ignition temperature. To yield more energy from the fusion than has been invested to heat the plasma, the plasma must be held up to this temperature for some minimum length of time. For D-T reaction:

$$T_{D,T} = 4.5 \cdot 10^7 k$$

(1.4)

At that temperatures the gas becomes a plasma, an ionized gas which atoms have lost almost all their electrons. The plasma is charge balanced, all the ions generated by the high temperature are balanced by free electron lost during ionization.

To control the plasma at this high temperature two methods have been developed:

- inertial confinement
- magnetic confinement

Inertial confinement is a process where D-T reactions are initiated by heating and compressing a fuel target, typically in the form of a pellet. To compress and heat the fuel, energy is delivered to the outer layer of the target using laser beams, ion beams, or X-ray radiation.
Chapter 1 - Energy and Nuclear Fusion

This hot outer layer then explodes, producing a reaction force against the remainder of the target, accelerating it inwards, and sending shock waves into the center. A sufficiently powerful set of shock waves can compress and heat the fuel at the center so much that fusion reactions occur. The energy released by these reactions will then heat the surrounding fuel, which may also begin to undergo fusion. The aim of inertial confinement is to produce a condition known as "ignition", where this heating process causes a chain reaction that burns a significant portion of the fuel. Typical fuel pellets are about the size of a pinhead and contain around 10 milligrams of fuel: in practice, only a small proportion of this fuel will undergo fusion, but if all this fuel was consumed it would release the energy equivalent to burning a barrel of oil.

Magnetic confinement uses magnetic fields to confine the plasma, it is one of two major branches of fusion energy research, the other being inertial confinement fusion mentioned above. The magnetic approach is more highly developed and is usually considered more promising for energy production. Within the nuclear fusion devices, magnetic fields are used to contain the charged particles that compose the hot plasma and keep it away from the chamber walls. Magnetic confinement rests upon the property that charged particles, like those in a plasma, will travel along the lines of a magnetic field. By arranging magnetic fields in just the right way, it is possible to “trap” the plasma within the fields.

![Figure 3 - Magnetic field lines in a “magnetic mirror”.
(courtesy of General Atomics)](image)

While the plasma is held, it can be heated through a combination of microwaves, particle beams, and the heating generated from currents flowing through the plasma. Currently there are two types of magnetic confinement systems: the mirror (open, see Fig.3) and the toroidal (closed, see next section). The primary toroidal method we will be looking at is the tokamak, although there are other toroidal confinement techniques, including reversed-field pinch, stellarator, and others undergoing research.
1.3 Magnetic Confinement Devices.

An early attempt to build a magnetic confinement system was the stellarator, introduced by Lyman Spitzer in 1951. Essentially the stellarator (see Fig.4) consists of a torus that has been cut in half and then attached back together with straight "crossover" sections to form a figure-8. This has the effect of propagating the nuclei from the inside to outside as it orbits the device, thereby canceling out the drift across the axis, at least if the nuclei orbit fast enough. Newer versions of the stellarator design have replaced the “mechanical” drift cancellation with additional magnets that “wind” the field lines into a helix to cause the same effect.

![Figure 4 - Stellarators Wendelstein 7-X](image)

The most promising shape for magnetic confinement is the TOKAMAK (Toroidalnaya Kamera i Magnitnaya Katushka), (see Fig.5) presented in 1968 by russian researchers [5]. Since then the majority of effort in magnetic confinement has been based on the tokamak principle. In the tokamak a current is periodically driven through the plasma itself, creating a field “around” the torus that combines with the toroidal field to produce a winding field in some ways similar to that in a modern stellarator (see Fig.4) at least in that nuclei move from the inside to the outside of the device as they flow around it.

![Figure 5 - Toroidal and Poloidal field](image)
As we can see from Fig. 5 the resultant field generated by poloidal and toroidal combination is helicoidally shaped.

Not only does the plasma current play an important role in the confinement but also for ohmic heating the plasma itself, however the resistivity decreases when the temperature rises, the ohmic heating is not enough to reach the plasma ignition.

Other, not less important, magnetic confinements such as linear are not presented because are out of the aim of this work.

### 1.3.1 Joint European Torus

The Joint European Torus, also known as JET, it is the biggest fusion reactor in the world. It started operating in 1983 and was the first fusion facility in the world to achieve a significant production of controlled fusion power (nearly 2MW) with a Deuterium-Tritium experiment in 1991 [1].

As we can see from Fig. 6, the toroidal component of the magnetic field on JET is generated by 32 large D-shaped coils with copper windings, which are equally spaced around the machine. The primary winding (inner poloidal field coils) of the transformer, used to induce the plasma current, which generates the poloidal component of the field, is situated at the centre of the machine.

![Figure 6 - JET Magnetic field configuration](image)

Coupling between the primary winding and the toroidal plasma, acting as the single turn secondary, is provided by the massive eight limbed transformer core. Around the outside of the machine, but
within the confines of the transformer limbs, is the set of six field coils (outer poloidal field coils)
used for positioning, shaping and stabilizing the position of the plasma inside the vessel.

![Image of JET cross section]

**Figure 7 - JET cross section.**

The use of transformer action for producing the large plasma current means that the JET machine
operates in a pulsed mode. Pulses can be produced at a maximum rate of about one every twenty
minutes, and each one can last for up to 60 seconds in duration. The plasma is enclosed within the
doughnut shaped vacuum vessel which has a major radius of 2.96m and a D-shaped cross section of
4.2m by 2.5m (see Fig.7). The original main design parameters are detailed in Tab.1.

In addition to ohmic heating, three additional heating systems have been installed: Ion Cyclotron
Radio Frequency (ICRF), Neutral Beam Injection (NBI) and Lower Hybrid Current Drive (LHCD).
The NBI is based on the injection of powerful beams of neutral atoms into ohmically pre-heated
plasma. The beam atoms carry a large uni-directional kinetic (motional) energy. In the plasma,
beam atoms loose electrons due to collisions, i.e. they get ionized (electrically charged) and as a
consequence are captured by the magnetic field of tokamak. These new ions are much faster than
average plasma particles. In a series of subsequent ion-ion, ion-electron and electron-electron
collisions, the group velocity of beam atoms is transferred into an increased mean velocity of the
chaotic motion of all plasma particles.

Although the LHCD system has an inefficient heating effect it is useful to drive electric current of
several MA. The electromagnetic wave is generated in klystrons - tubes that can produce the above
frequencies by resonant modulation of an electron beam. At JET, 24 klystrons are installed in 6
independent modules. The electromagnetic wave is then transmitted to the LHCD antenna by a complex system of waveguides. Waveguides are hollow rectangular metallic conductors with cross-section size that corresponds to the transmitted wavelength. The LHCD antenna is of a very sophisticated design, called “multi-junction grill” in order to allow for a correct phasing of the wave before it is launched into the plasma. The correct phasing of LHCD waves is hampered by propagation in vacuum, therefore it is required that LHCD antenna is mounted directly in the JET inner wall, as close to the plasma as possible.

Tab. 1 - Jet Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plasma major radius</td>
<td>2.96m</td>
</tr>
<tr>
<td>Plasma minor radius</td>
<td>2.10m (vertical) - 1.25m (horizontal)</td>
</tr>
<tr>
<td>Flat-top pulse length</td>
<td>20s</td>
</tr>
<tr>
<td>Weight of the iron core</td>
<td>2800t</td>
</tr>
<tr>
<td>Toroidal Field Coil Power (Peak On 13s Rise)</td>
<td>380MW</td>
</tr>
<tr>
<td>Toroidal magnetic field (on plasma axis)</td>
<td>3.4T</td>
</tr>
<tr>
<td>Plasma current</td>
<td>3.2MA (Circular plasma)</td>
</tr>
<tr>
<td></td>
<td>4.8MA (D-Shape plasma)</td>
</tr>
<tr>
<td>Volt-seconds to drive plasma current</td>
<td>34Vs</td>
</tr>
<tr>
<td>Additional heating power</td>
<td>25MW</td>
</tr>
</tbody>
</table>

ICRF is routinely applied on JET. It is resonant with the second harmonic (i.e. double) frequency of ion gyration of main JET plasma ions (deuterium) or with a base frequency of gyration of a minority species (e.g. tritium, helium...).

The available resonant frequencies at JET are in the range of 23-57 MHz (megahertz, or million of oscillations per seconds), which correspond to length of the vacuum electromagnetic wave from 13m (at 23 MHz) down to 5m (at 57 MHz). This is a “shortwave” frequency, which is not very popular in the air due to many fades, blackouts and interferences (the FM radio frequencies are just a bit higher, around 100 MHz). In total, the installed power of JET ICRF system is as much as 32 MW, and in practice only part of this potential is sufficient for the JET experiments. This is a huge power compared to radio or TV broadcast, where a 50 kW transmitter is already considered as a powerful one.

Amplifier chains generate the ICRF electromagnetic waves, each chain with a powerful (2MW output) tetrode tube in final stage. Transmission lines that conduct ICRF waves from the generators to the JET tokamak are low loss coaxial cables. Coaxial cables consist of a conducting outer metal tube enclosed and insulated from a central conducting core. Such cables are generally used in any high-frequency transmission - e.g. signal from the TV aerial or satellite dish is transferred to the TV set by a coaxial cable. However, at JET due to high powers involved the ICRF output coaxial cables look rather like 'pipelines' with 20cm diameter of the outer metal tube. Several hundreds meters of
these transmission lines are installed at JET. The transmission lines terminate in 4 ICRF antennas that are installed within the JET inner wall and that are slotted in the front. Each antenna consists of four conductors (straps), and each strap is fed by a separate generator. The ICRF electromagnetic waves cannot propagate in the JET vessel vacuum (their wavelength being too long) so that the antenna must be as close to the plasma as possible [1].

### 1.3.2 Confinement at JET.

In a machine like JET the confinement is realized by a closed magnetic surface generated by two magnetic fields. One is due to the plasma current while the other one is generated by an external magnetic field.

![Figure 8 - Limiter and X Point Configurations.](image)

That magnetic surface could be limited by a solid surface determining a configuration known as ‘Limiter’, otherwise if the magnetic surface is completely limited by magnetic fields that configuration is called ‘X-Point’. Fig.8 shows both the X-Point and Limiter configurations.

With elongated plasmas there is a special magnetic surface called a *separatrix* which separates two different types of magnetic topology. Surfaces inside the separatrix form closed, nested magnetic surfaces, as needed for good confinement. Outside the separatrix the surfaces are open and at moderate elongation that surfaces lie outside the vacuum vessel, but if a new high elongation occurs the separatrix enters the vessel with X-Points just inside the vacuum vessel. An X-point
configuration is more stable if compared to the Limiter one but the plasma is smaller, yet, with the Limiter configuration the plasma is more sensible to impurities.

The X-point configuration seems to be the choice for next Tokamaks.
References


Chapter 2 - Disruptions prediction – State of the Art.

2.1 Introduction.

Disruptions are an endemic and likely unavoidable (at least on an occasional basis) aspect of tokamak operation [1]. To satisfy the power balance of the reactor the confinement time must be long enough, so, the plasma has to be sufficiently large. But larger plasma lead to an increased current carrying capacity and this is the cause for strong instabilities. One of the most serious and frequent instability is the so called disruption.

During a disruption a sudden loss of plasma energy occurs and the discharge rapidly terminates, large current are induced into the vacuum vessel and the vessel itself could be seriously damaged, the plasma facing component surface area is highly subject to localized melting, vaporization and erosion. Electromagnetic (EM) loads, on electrically conducting first wall and vacuum vessel structures, result in structural loadings that are typically higher than the loadings that arise in these components owing to their vacuum/pressure containment and gravity/seismic support functions. As result of major disruption, plasma energy confinement is lost, the plasma is cooled to sub-keV temperatures in less than 1 ms, and global contamination of the plasma by products of chamber wall desorption and erosion ensues. These factors result in a dramatic increase in plasma electrical resistance and an ensuing rapid current quench, at rates that can approach 1000 MA/s. Modification of in-vessel component designs to fully withstand disruption thermal and/or EM loading have been required in many ‘modern’ tokamaks, including JET, JT-60U and PBX [1].

As just mentioned, disruption avoidance is a critical issue for tokamak machines, and although several studies have been conducted it is still an open issue. The next section presents the most relevant works related to disruption prediction including both neural techniques and MHD studies.

2.2 State of the Art for disruption prediction.

In literature several papers described the operational limits of a tokamak and the theoretical stability limits of the plasma: the Greenwald plasma density limit [2], the high-\(\beta\) limit [3][4], the \(l_i-q_\psi\) diagram [5], the ratio of the radiated power to the input power [6] and MHD plasma instabilities [7]. Nevertheless none of them have led to the development of a reliable predictive model of disruptions as the large number of parameters involved. For this reason in the last 15 years, there have been
several studies for the prediction of disruptions using neural networks. In particular some papers are focused on predicting the proximity of the plasma to disruption by using an artificial output. The first application of neural networks for tokamak real-time control is reported in [8]. The net is trained with a database representing the plasma position of a COMPASS tokamak experiment. The result showed that the network responds in close agreement with the signals obtained from the device. In [9] the authors propose an approach base on a Bayesian probability assessment of the disruptive phenomenon, the Bayesian probability is modeled by a MLP neural network. The model try to investigate if a set of parameters is useful as a coming disruption indicator. An artificial neural network was used in [10] to estimate the high-β disruption boundary in the DIII-D tokamak. The authors combined signals from a large number of plasma diagnostics, demonstrating that the network, so trained, provided a much more accurate prediction of the disruption boundary than that provided by traditional Troyon limit. In [11] a feedforward neural network is used to forecast major and minor disruptive instabilities in TEXT tokamak discharges. The net is trained with soft x-ray signals related to just one plasma disruptive discharge. The authors suggest to use soft x-ray signals because better equipped than magnetic ones as indicators of the possible origin of disruptive instability. A time series prediction method is employed in [12] using diagnostics signals from Mirnov probes, soft x-ray and Hα monitor. The neural network trained with past signals offers a prediction up to 8 ms earlier to the event and it is well suited for on-line application in the ADITYA device. In [13] the predictions using high quality discharge equilibrium reconstructions are tested against the observations for the principal limiting phenomena in DIII–D. Ideal eigenfunctions are then used to predict stabilization using active feedback. In [14] the authors compute the remaining time-to-disruption to indicate the stability level of the discharge. The neural network is specified, trained and then implemented within the real-time plasma control system. In [15] an on-line predictor of the time to disruption installed on the Asdex Upgrade tokamak is presented. The prediction system uses a neural network trained on eight plasma parameters and some of their time derivatives extracted from 99 disruptive discharges. The non disruptive phase was defined as the L-mode phase following the H-mode phase before the disruption, or the phase starting just before a MARFE (a multifaceted asymmetric radiation from the edge) and ending with a disruption, for plasma which has been in L-mode for more than 0.8 s. The system was implemented and tested for real-time mitigation, showing satisfactory prediction capability. However the authors highlight the deterioration of the network performance on on-line tests, due to
the slight difference between the real-time signal and the stored ones. Moreover, it has been shown that new experiments, which belong to operational spaces different from those used for training, are not well predicted in the on-line implementation, thus presenting the so-called ‘ageing’ of the neural network.

The aim of Versaci and Morabito in [16] is to develop a processing system, a model with two-factor time series, able to predict correctly the “time to disruption” by using a fuzzy time series approach for ASDEX-Upgrade machine.

The major disruptions caused by the density limit, the plasma current ramp-down with high internal inductance, the low density locked mode, and the $\beta$-limit in JT-60U, have been investigated in [17]. The concept of ‘stability level’, proposed in the paper is calculated from nine plasma parameters by a MLP, and the occurrence of a major disruption is predicted when the stability level decreases to a certain level, named the ‘alarm level’. In particular, the onset of the major disruption is determined as the start of the positive current spike followed by the plasma current quench.

The authors in [18] combine multiple plasma diagnostic signals to provide a composite impending disruption warning indicator. To take into account the disruption precursor appearing in different time instants for different pulses, an off-line clustering procedure automatically select the training set samples.

The work presented in [19] has been performed on flat-top JET scenarios characterized by a single null plasma. The authors trained a MLP to forecast disruptive events at JET, up to 100 ms in advance. Pulse samples have been selected in a temporal window of 400 ms. For disruptive pulses, the window is constituted by the last 400 ms of the discharge and the artificial output is a sigmoid representing the risk of disruption. In the paper a saliency analysis is also presented to validate the suitability of the selected input signals.

The $\beta$-limit disruption prediction performance presented in [17] has been improved in [20] with a cascade of specialized MLPs. Since no clear precursor are generally observed a few tens of ms before the $\beta$-limit disruption, a neural network is trained to output the value of the $\beta_{\text{N}}$ limit every 2 ms.

In [21] two neural approaches (Self Organizing Maps and Support Vector Machines) are used to determine the novelty of the output of the neural disruption predictor. The novelty detector is used to assess the reliability of the network output, i.e., samples having a low confidence have to be discarded and used off line to update the disruption predictor.

For what concern [22], a Support Vector Machine has been developed to realize a predictor and a novelty detector in a unique system. This approach try to overcome the drawback of neural networks ageing analyzing the Support Vector decision function. It is widely recognized that a
neural network deteriorates its performance when completely new patterns are presented to it. With this approach the authors found that the novelty detector justifies many of the missed alarms of the predictor as they are recognized to belong to unexplored regions of the operational space.

A new criterion for disruption prediction on HL-2A tokamak is presented in [23], almost of the disruptions present in HL-2A can be forecasted introducing an integral of poloidal magnetic field over time as precursor. The success on prediction is on more than 95% of disruptive events in HL-2A device.

In [24] different pattern recognition techniques have been tested in order to implement an automatic tool for disruption classification in a tokamak experiment. The aim of prediction can be performed by classifying patterns coming from the device, forecasting the event at least 100 ms before the disruption.
References


Chapter 2 – Disruption prediction – State of the Art


Chapter 3 - Kernel Machines – Conventional approaches

3.1 Introduction

As mentioned in the last section, a black box approach with neural networks, for prediction purposes, or more in general with kernel machines, could be a complementary to more conventional ones. All methods based on kernel machines give a systematic principled approach to training learning machines and the good generalization performance achieved can be readily justified using statistical learning theory or Bayesian arguments [1].

In the present dissertation, an original approach to the synthesis of MLPs has been used to develop a disruption prediction system, and its prediction performance has been compared with that of different disruption prediction systems based on more conventional kernel machines: Multi layer Perceptron (MLP) neural networks, Self Organizing Maps (SOM), and Support Vector Machines (SVM). In this chapter it will be described how to use kernel methods for classification focusing on MLP neural networks, SOM, and SVM. The next chapter will be devoted to a detailed description of the constructive algorithm for MLP neural networks, and, in the chapter 7, how to use it as a disruption predictor.

As we will see in chapter 6, the aim of the prediction for disruption events is achieved by classifying patterns, coming from a selection of JET database, at least 100 ms in advance with respect to the time to disruption.

Hence, it is very important to reach the best classification/prediction rate in order to prevent the disruption or to allow the mitigation system to intervene in time.

3.2 Multi Layer Perceptron

An MLP is the most widely used type of neural network. It is both simple and based on solid mathematical backgrounds. Input quantities are processed through successive layers of “neurons”, see Fig.1. There is always an input layer, with a number of neurons equal to the number of variables, and an output layer, where the perceptron response is made available, with a number of neurons equal to the desired number of quantities computed from the inputs. The layers in between are called “hidden” layers. With no hidden layer, the perceptron can only perform linear tasks (for example a linear discriminant analysis, which is already useful).
From mathematical point of view a MLP with a single hidden layer provides a non-linear combination of the inputs:

\[
y_k(x) = f \left( \sum_{i=1}^{m} w_{ik} g \left( \sum_{j=1}^{n} w_{ij} x_j + \theta_i \right) + \theta_k \right) = f(\text{net}_k)
\]

where \( y_k \) is the \( k \)th output neuron, \( m \) is the number of hidden neurons, \( n \) is the number of input neurons, \( x_j \) is the \( j \)th input vector component, \( w_{ij} \) is the connection weight between the \( i \)th and the \( j \)th neuron while \( \theta_i \) and \( \theta_k \) are the bias values and \( f \) and \( g \) are the activation functions of the hidden and output neurons respectively.

The most common activation function for this kind of neural network are:

- Log-sigmoidal
- Hyperbolic-tangent
- Linear

Training a MLP neural network means to find an optimal set of connection weights in order to minimize the output error. The most used training algorithms work are based on Error Back Propagation (EBP) technique. The error, the difference between the actual output and the desired one, is then minimized sending the error backwards through the hidden layer as far as the input layer [2]. After randomly initializing the connection weights \( w_{ij} \), the \( N \) couple input/output, belonging to a certain set

\[
D = \{ x^{(p)}, d^{(p)} \} \quad p=1, \ldots, N,
\]

they iteratively feed the net until the error function reaches one of its lowest values. Let us define the error function as the mean squared error:
Chapter 3 – Kernel Machines – Conventional approaches

\[ E = \frac{1}{K \cdot N} \sum_{p=1}^{N} \sum_{k=1}^{K} (d_k^{(p)} - y_k^{(p)})^2 \]  \hspace{1cm} (3.3)

where \( y_k^{(p)} \) is the output of the \( k \)th neuron, corresponding to the input \( x(p) \), \( d_k^{(p)} \) is the desired output and \( K \) the number of output neurons.

The weight \( w \) is modified by an amount \( \Delta w \) proportional to the gradient of the error function, in a vector form it is given by the following:

\[ \Delta w = -\eta \cdot g \]  \hspace{1cm} (3.4)

where \( g \) is the gradient at the \( i \)th iteration, \( \eta \) is the learning rate.

Another training algorithm, belonging to the global optimization approaches, is proposed by Levenberg-Marquardt (LMA)[3]. It interpolates between the Newton algorithm (NA) and the method of gradient descent. The LMA is more robust than the NA, which means that in many cases it finds a solution even if it starts very far off from the final minimum. On the other hand, for well-behaved functions and reasonable starting parameters, the LMA tends to be a bit slower than the NA.

The LMA is a very popular curve-fitting algorithm; most software that provide a generic curve-fitting tool provide an implementation of it.

For LMA:

\[ H \cong 2 \cdot J^T \cdot J \]  \hspace{1cm} (3.5)

and the gradient of the error function is:

\[ g = J^T \cdot e \]  \hspace{1cm} (3.6)

\( J \) is the Jacobian matrix of the mean squared error with respect to the connection weights, while \( e \) is the vector of the errors of the net.

More in general for LMA:

\[ \Delta w = -\left( J^T \cdot J + \mu \cdot I \right)^{-1} \cdot g \]  \hspace{1cm} (3.7)

Another important aspect of learning algorithm is how to stop the algorithm. A common criterion is to stop it when the error itself is less than a certain threshold. Sometimes this is not the best way as the overfitting can occur, or rather the input and the output seems to be related by a relation that does not actually exist. Some authors suggest to use a stop-learning criterion based upon cross-
validation [2], where the input set is divided in training set and validation set, the algorithm is applied on the training set while the lowest error is reached for the validation set.

For what concern the problem of classification, where a class $C_k$ is associated to an input vector $x^{(p)}$, let us consider a MLP neural network with $n$ inputs, $K$ output neurons, an hidden layer with $m$ neurons. A training set $D = \{x^{(p)}, d^{(p)}\} \quad p=1,...,N$, where the $k^{th}$ output is:

$$y_k^{(p)} = F_k(x^{(p)}) \quad k = 1,2,...,K. \quad (3.8)$$

Then, once the network is trained, the classification is then achieved associating an output $y_k^{(p)}$ to each pattern $x^{(p)}$, and the classification error is done by comparing the results with the actual class the pattern belongs to.

### 3.3 Self Organizing Map

The SOM is a subtype of artificial neural networks. It is trained using unsupervised learning to produce low dimensional representation of the training samples while preserving the topological properties of the input space. This makes SOM reasonable for picturing low-dimensional views of high-dimensional data, akin to multidimensional scaling. It can be used to clarify relations in a complex set of data, by revealing some inherent order.

The model was first described by the Finnish professor Teuvo Kohonen and is thus sometimes referred to as a Kohonen map [4].

A set of vectors is input repeatedly to a map consisting of units, associated with each unit is a weight vector $w_i$, see Fig.3, initially consisting of random values. Units respond more or less to the input vector according to the correlation between the input vector and the unit's weight vector. The unit with the highest response to the input is allowed to learn, as well as some units in the
neighborhood. The neighborhood decreases in size during the training period according to the exponential decay function:

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\lambda}\right) \quad t=1,2,3,\ldots$$

(3.9)

where $\sigma_0$ denotes the width of the grid at time $t_0$ and $\lambda$ is a time constant, $t$ is the current time-step.

The learning is done by adjusting the weights of the units by a small amount every time-step according to the following equation:

$$w(t+1) = w(t) + L(t)(v(t) - w(t))$$

(3.10)

where $w(t)$ is the weight at the time-step $t$, $L$ is the learning rate which decreases with time, $v$ is the input vector. The decay of $L$ follows (3.9).

In order to take into account the shrinking of the neighborhood the 3.10 becomes:

$$w(t+1) = w(t) + \Theta(t)L(t)(v(t) - w(t))$$

(3.11)

$\Theta$ represents the amount of influence a node’s distance from the winning node has on its learning, $\Theta(t)$ is given by equation:

$$\Theta(t) = \exp\left(-\frac{d}{2\sigma^2(t)}\right) \quad t=1,2,3$$

(3.12)

where $d$ is the distance a node is from the winning node, while $\sigma$ is the width of the neighbourhood function as calculated by 3.9 $\Theta$ also decays over time.

The SOM differs considerably from the feed-forward back propagation neural network, both in how it is trained and how it recalls a pattern. The SOM does not use any sort of activation function and it does not use any sort of a bias weight. When a pattern is presented to a SOM one of the output neurons is selected as a “winner”. This “winning” neuron is the output from the SOM. More precisely, during the training each weight is set to random value, when input arrives neuron with pattern of weights most similar to input gives largest response, causing all the units in “neighborhood” of winner unit to learn.

The result of the training is that a sort of organization emerges in the map. Different units learn to respond to different vectors in the input set, and units closer together will tend to respond to input vectors that resemble each other.

When the training is finished, the set of input vectors is applied to the map once more, marking for each input vector the unit that responds the strongest (is most similar) to that input vector [5].
The SOM are also an useful tool in “correlation hunting”, that is, in inspecting the possible correlation between vector components in the input data. In practice the SOM is sliced to component planes and the value of one component in each node of the SOM using gray-level or color representation, as shown in Fig.4. However, an accurate comparison of color coded values on the planes is difficult or even impossible [6][7].

The most significant difference between the SOM and the feed-forward back propagation neural network is that the SOM is trained in an unsupervised manner. This means that the SOM is presented with data, but the correct output that corresponds to that data is not specified, furthermore this data can be classified into groups. About the limitations of the SOM it is important to remember that not having hidden neurons, once defined a cluster, the SOM is not able to split up the
cluster in smaller sets. There is no official way to calculate the error for a SOM, and the error itself is not used to modify the weights, as in the back propagation algorithm.

Despite of all drawbacks mentioned above, it is worth remembering the SOM is a relatively simple network to construct and it can be trained very rapidly [8].

Anyway, to profit of SOM ability on clustering pattern, it has been used to recognize disruption precursors on a subset of JET disruptive pulses. Each pulse has been clustered with respect of the euclidean distance from the one representing the disruption event. That cluster contains disruption precursors information and it has been demonstrated to have less than 3 clusters neighbors if compared to not disruptive ones (see chapt.7 for further details).

3.4 Support Vector Machines

SVMs has been firstly proposed by Vapnik [9] over the past decade. The original optimal hyperplane algorithm proposed by Vladimir Vapnik in 1963 was a linear classifier. However, in 1992, Bernhard Boser, Isabelle Guyon and Vapnik suggested a way to create non-linear classifiers by applying the kernel trick (originally proposed by Aizerman) to maximum-margin hyperplanes. The resulting algorithm is formally similar, except that every dot product is replaced by a non-linear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in the transformed feature space. The transformation may be non-linear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space it may be non-linear in the original input space.

If the kernel used is a Gaussian radial basis function, the corresponding feature space is a Hilbert space of infinite dimension. Maximum margin classifiers are well regularized, so the infinite dimension does not spoil the results.

Hence, the SVM is built by detecting an optimal separating hyper-plane, which maximizes the margin between itself and closest training data points. An optimal separating hyper-plane separates the data in two classes. The support vectors are those patterns that determine that margin; they are informally the hardest pattern to classify and the most informative ones for designing the classifier [10].

From the perspective of statistical learning theory the motivation for considering binary classifier as Support Vector Machines, SVM, comes from theoretical bounds on the generalization. Firstly the error bound is minimized by maximizing the margin, i.e., the minimal distance between the hyperplane separating the two classes and the closest data points to the hyperplane itself, see Fig.5. Secondly the upper bound on the generalization error does not depend on the dimension of the space.
In case of nonlinear decision functions, the SVM projects the input vector $x$ into a high-dimensional feature space $H$ and constructs the optimal separating hyper-plane in this space. The optimal hyper-plane is obtained by solving the following optimization problem:

$$\begin{align*}
\text{minimize} & \quad E(w) = \frac{1}{2} w^T w + C \sum_{i=1}^{I} \xi_i \\
\text{subject to} & \quad (y^i \cdot (w^T \Phi(x^i) + b)) \geq 1 - \xi_i \\
& \quad \xi_i \geq 0 \quad i=1,\ldots,I
\end{align*}$$ (3.13)

where $w$ and $b$ are the coefficients of the optimal hyper-plane, $C$ is an error penalty, and $\xi_i$ are parameters handling no separable inputs. The index $i$ labels the $I$ training cases; $x^i$ is the $i^{th}$ training input pattern, and $y^i \in \{-1, +1\}$ is the corresponding class label. The kernel $\Phi$ is used to transform data from the input to the higher dimensional feature space.

There are different kernels that can be used in SVM models, e.g.:

- polynomial
  $$\Phi(x^i, x^j) = (y^i x^i \cdot x^j + \theta)^p$$ (3.14)

- gaussian radial basis
  $$\Phi(x^i, x^j) = e^{-\|x^i - x^j\|^2 / \sigma^2}$$ (3.15)

There exist also linear and sigmoid kernel functions, not considered in this context as the performance reached were not satisfactory for our purposes.

As said above, if the kernel function is a radial basis function, the corresponding feature space is a Hilbert space of infinite dimension [10]. Remember that an Hilbert space is an inner product space, which means that it has notions of distance and of angle (especially the notion of orthogonality or
perpendicularity), it allows simple geometric concepts, like projection and change of basis to be applied to infinite dimensional spaces, such as function spaces.

Hence, the images of the input points are always linearly separable in the feature space.

It should be noted that all the parameters given to SVM, $C$ and $\xi_i$, should be carefully chosen to avoid over fitting.

Once an SVM is trained, the class membership of a given test pattern $x$ is determined depending on the sign of the following decision function $f(x)$:

$$f(x) = \mathbf{w}^T \cdot \Phi(x) + b$$

(3.16)

The feature making SVMs very attractive is that classes nonlinearly separable in the original space can be linearly separated in the higher dimensional feature space. Thus, SVM is capable to solve complex nonlinear classification problems, but to chose a proper mapping function the designer should know very well the problem domain.

Being the generalization capability of SVMs the correctness in classifying patterns not belonging to the training set, defined by the following:

$$R(\alpha) = \int L(p, f(x, \alpha)) dP(x)$$

(3.17)

where $x$ is the vector of patterns, $d$ is the class of the patterns, $P$ is the cumulative probability distribution, $L$ is a generic loss function and $\alpha$ is a set of parameters.

$R(\alpha)$ is also known as expected risk. The goal of a learning machines is to reduce that risk.

Defining the empirical risk $R(\alpha)_{EMP}$ as the measured mean error rate on the training set, we have:

$$R(\alpha) \leq R(\alpha)_{EMP} + \Omega(h)$$

(3.18)

where $h$ is a non-negative integer called the Vapnik Chervonenkis (VC) dimension, known as the ability of the machines to learn any training set with no errors, while $\Omega(h)$ is called VC confidence.

Hence, in order to reduce $R(\alpha)$, the empirical risk and also VC have to be minimized.

SVMs are able to reduce VC during the training phase, in this way also the error on the training set is reduced, while once a MLP neural network has been trained there is no chance to modify VC.
References


4.1 Introduction

In this section a new constructive algorithm to design Multilayer Perceptron networks used as classifiers is presented. The resulting networks are able to classify a finite set of patterns defined in a real domain. The proposed procedure allows us to automatically determine both the number of neurons and the synaptic weights of networks with a single hidden layer. The approach is based on linear programming. It avoids the typical local minima problems of Error Back Propagation (more in general, the issue of local minima is a drawback of whatever deterministic method, even of Levenberg-Marquard algorithm, for instance) and assures convergence of the method. In this section the performances of the proposed algorithm have been tested on some benchmark problems, and they have been compared with those of different approaches. In the chapter 7 the application to disruption prediction at JET using this kernel machine will be presented.

4.2 State of the Art.

In several application fields of neural networks, the black-box approach, which is typical of supervised learning, has not produced the hoped results, either because of a wrong choice of training patterns or network architecture, or because the learning phase has been unsuccessful. In order to extend the field of application of the neural approach and to improve the performance of neural systems, several researchers paid a lot of attention to the internal organization of acquired information [1]. It is always possible to train a neural network, which correctly classifies a given training set, but it is in general a difficult task to determine the network, which performs the desired classification, and at the same time has the minimal size. To address this issue many authors have proposed methods, which overcome the traditional black-box approach of the neural networks, by considering the task performed by each single part of the network.
Among the different kinds of network architectures, the most commonly used is the Multilayer Perceptron (MLP). A suitably trained MLP with a single hidden layer is able to correctly classify any set of patterns [2], [3]. The problem in synthesizing such networks is to determine both the
number of nodes in the unique hidden layer and the weights of the synapses between the layers. The former part of the problem is usually solved by a trial and error procedure, i.e., by training several networks of increasing or decreasing size on the same training patterns, while the latter part is solved by using learning algorithms based on error minimization to find the connection weights. Among the different learning algorithms the most commonly used is the Error Back Propagation (EBP). Such algorithm, and the several improvements of the original version, is based on the evaluation of the gradient of an error function in order to find its minimum. A common problem, which the network trainer has to handle, is the entrapment in local minima of the error function, because of the null gradient. Several methods have been proposed to overcome this problem, but it is impossible to tell a priori whether an error function minimum is local or not.

Some authors proposed different interpretations for neural networks, deducing useful information to develop the learning strategy, and in some cases they have approached the problem in a completely different way with regard to the classical error correction method, which is typical of the EBP. In particular, a geometrical approach for the learning has been proposed by some authors, which is alternative with respect to the classic black-box approach. Following the geometrical interpretation, each element of the net has a well-defined task and the consequences of removing a part of it can be forecasted with precision.

In [4] the geometric interpretation of MLP networks is proposed to solve the pattern classification task. Each neuron behaves as a linear separator and the coefficients of the hyper-plane that defines this separator are equal to the connection weights connected to that neuron. The proposed interpretation is useful to determine the network size and the learning rate, but the procedure still requires an EBP learning algorithm to find the weights of the connections.

In [5], starting from the above method, a learning algorithm to synthesize the neural networks used for problems of binary – binary mapping is presented. The aim of the proposed algorithm was to define a set of hyper-planes that performed the desired classification of a set of patterns. The authors showed that the presented algorithm is always able to find a single hidden layer neural network that can solve the classification problem. Moreover, a comparison of these results with those obtained using the EBP algorithm shows that the geometrical approach proposed in [5] is far better. However, the limit of the method is that it can be applied to classify patterns with binary values.

In [6], [7] and [8] the authors extend the geometrical approach to the real values case. They establish important proprieties concerning the bounds on the networks size. In particular in [6] the authors fix the minimal size of the hidden layer, which permits to arbitrarily classify a set of real points. In [7] the number of hidden neurons is determined by satisfying rank conditions on the
output of the hidden layer. The weights for the hidden layer can be assigned almost arbitrarily, and
the weights for the output layer are calculated by solving linear equations. In [8] the authors define
a constructive algorithm to find a set of hyper-planes that maps the input space, yielding a
near-minimal hidden layer. The approaches in [6], [7] and [8] perform better than the EBP
algorithm, but none of them guarantees that all the training patterns are correctly classified.
In [9] the authors use neural layers to solve the image segmentation problem. To this end, a dipolar
criterion function has been minimized with a procedure similar to Linear Programming (LP)
methods. The result is a convex polyhedrons set in the feature space, which could be treated as
prototype of a class. Also this method does not guarantee the correct classification of all the training
patterns, but it introduces the concept of enveloping the training points by means of polyhedrons,
which is one of the key concepts of the method presented in this chapter.
In [10] and [11] the authors present several algorithms to design MLP and Cascade MLP [12]
networks using the Linear Programming (LP). The methods perform very well to construct neural
classifiers, but they suffer the highly irregular distribution of the patterns that can cause
misclassifications of some training examples.
In the last years, there has been a growing interest toward the Support Vector Machines (SVM)
[13]. The key idea of such method is to project the training set in a high-dimensional space by
means of a set of non-linear kernel functions, as mentioned in the section 3.4 the distance between
points and hyper-planes is set after projection. In [14] the authors, starting from the definition of
convex recursive deletion (CoRD) regions, propose an algorithm to find the architecture of a single
hidden layer MLP that implements such regions. The algorithm is able to classify any set of points
provided that the decision regions are known but no indication is given to find such regions. Thus
the classification capability of the neural network is not assured. Moreover as the attention is
focused on the feature space, rather than on the input space, the generalization capability of the
classifier is not taken under control.
This chapter is organized as in the following. In section 4.3.1 the formalization of the neural
network as classifier of real points is given. In section 4.3.2 the procedure to construct the neural
network is described and a theorem is demonstrated, which ensures that the described procedure
always leads to a size-limited MLP that correctly classifies whatever finite training-set. In section
4.5 the results are presented concerning benchmark problems and the performances of the proposed
method are compared with those of other algorithms for the synthesis of classifiers.
4.3 Geometrical Synthesis

4.3.1 Neural model formalization

Let $\mathbf{X}$ be a set of $M$ points $\mathbf{x}$ defined in $\mathbb{R}^I$. Let the points of $\mathbf{X}$ belong to $K$ classes $\omega_k$. Let $\mathbf{x}^m_k(k)$ be the generic point belonging to the class $\omega_k$, with $mk = 1, \ldots, MK$, and $C_k$ being the set of points of $\omega_k$. Let $\text{FN}(\mathbf{w}, \theta)$ be the formal neuron defined by the following decision rule:

$$
\text{FN}(\mathbf{w}, \theta; \mathbf{x}) = \begin{cases} 
1 & \text{if } \mathbf{w}^T \mathbf{x} \geq \theta \\
0 & \text{if } \mathbf{w}^T \mathbf{x} < \theta 
\end{cases}
$$

(4.1)

where: $\mathbf{w}^T \mathbf{x}$ is the inner product, $\mathbf{w}$ is the weights vector, and $\theta$ is the threshold.

Let $\text{LN}(\mathbf{W})$ be a layer, made up of $\Lambda$ formal neurons of the kind $\text{FN}(\mathbf{w}, \theta)$, connected to the same set of $I$ input neurons. The layer $\text{LN}(\mathbf{W})$ defines the following rule:

$$
\text{LN}(\mathbf{W}; \mathbf{x}) = \left[ \text{FN}(\mathbf{w}^1, \theta^1; \mathbf{x}) \quad \text{FN}(\mathbf{w}^2, \theta^2; \mathbf{x}) \quad \cdots \quad \text{FN}(\mathbf{w}^\Lambda, \theta^\Lambda; \mathbf{x}) \right] 
$$

(4.2)

From (4.2), $\text{LN}(\mathbf{W})$ associates each point $\mathbf{x}$, in the input space $\mathbb{R}^I$ to a point $\mathbf{h}$, in the feature space $\mathbb{R}^\Lambda$, that lies on a vertex of a unitary hyper-cube.

Let the neural network $\text{NN}(\mathbf{W}^1, \mathbf{W}^2)$ be made up of the following items:

A first layer $\text{LN}(\mathbf{W}^1)$ of $\Lambda$ formal neurons $\text{FN}(\mathbf{w}^\lambda, \theta^\lambda)$ that forms an association between $\mathbb{R}^I$ and $\mathbb{R}^\Lambda$: $\mathbf{h} = \text{LN}(\mathbf{W}^1; \mathbf{x})$ where: $\mathbf{x} \in \mathbb{R}^I$ and $\mathbf{h} = [h_1, h_2, \ldots, h_\lambda]$ with $h_\lambda \in \{0,1\}, \lambda = 1, \ldots, \Lambda$.

A second layer $\text{LN}(\mathbf{W}^2)$ of $O$ formal neurons $\text{FN}(\mathbf{w}^o, \theta^o)$ that forms an association between $\mathbb{R}^\Lambda$ defined by $\text{LN}(\mathbf{W}^1)$, and $\mathbb{R}^O$: $\mathbf{u} = \text{LN}(\mathbf{W}^2; \mathbf{h})$ where: $\mathbf{u} = [u_1, u_2, \ldots, u_o]$ with $u_o \in \{0,1\}, o = 1, \ldots, O$.

The neural network $\text{NN}(\mathbf{W}^1, \mathbf{W}^2)$ fulfills the function: $\mathbf{u} = \text{NN}(\mathbf{W}^1, \mathbf{W}^2; \mathbf{x})$ between points of $\mathbb{R}^I$ and vertexes of a unitary hyper-cube in $\mathbb{R}^O$. Given a binary coding $u^k$ to each class $\omega_k$, a classification problem can be defined, for which the following association must be created:

$$
\forall \mathbf{x}^m_k(k) \rightarrow u^k \quad \forall \mathbf{x}^m_k \in C_k \quad k = 1, \ldots, K
$$

(4.3)
If a neural network $NN(W_1, W_2)$ has to be used to perform the classification (4.4.3), a network synthesis problem must be solved, i.e., the number $\Lambda$ of neurons in the first layer and the two matrices of weights $W_1$ and $W_2$ must be determined, such that for the neural network the following association holds true:

$$NN(W_1, W_2; x^{mk}(k)) = u^k \quad \forall x^{mk} \in C_k \quad k = 1, \ldots, K$$

### 4.3.2 Neural Network Synthesis

The formal neuron defined in (4.3.1) associates the values 0 and 1 of its output to two half-spaces, separated by a hyper-plane of equation: $<w, x> = \theta$. It is therefore able to classify two sets of linearly separable points.

There is a one-to-one correspondence between each neuron of $LN(W_1)$ and a hyper-plane in the input space $\mathbb{R}^I$. Moreover, a one-to-one correspondence also exists between each neuron of $LN(W_2)$ and a hyper-plane in the feature space $\mathbb{R}^\Lambda$.

An MLP neural network, with a single hidden layer and one output node, can correctly classify two sets of points, whatever is their spatial location, provided that the hidden layer maps the classes in such a way that they are linearly separable in the feature space. In [2] and [3] the authors demonstrated that an MLP network with a single hidden layer is capable to solve any classification problem, but they do not provide any method to design such a network. In [6], the authors demonstrated that an MLP network with a single hidden layer of $M-1$ neurons can arbitrarily classify any set of $M$ elements. The dimension of this hidden layer is clearly an upper bound, and becomes inadmissible when $M$ increases.

Hence, we need a general criterion that allows us to define a hidden layer with a limited number of neurons, able to map points belonging to two classes into two linearly separable sets in the feature space. The generalization of the method to $K$-classes problems will be discussed in subsection 4.4.2.4.

#### 4.3.2.1 Input Space Mapping

Let us firstly describe the proposed method in the case of two-classes classification problem. In general, the two classes of training examples are not linearly separable in the input space, but through a hidden layer, they can be projected into a high-dimensional space where they are linearly separable. For analogy with the SVM theory [13], the space where the input points are projected...
The synthesis of the hidden layer consists of determining a minimal number of neurons, i.e., the dimension of the feature space, and the connections weights. This corresponds to finding a set of hyper-planes in the input space having a prefixed margin from the training points. The choice of the margin affects the generalization capability: the wider the margins the greater the generalization capability, but the wider the margins the larger the number of hyper-planes needed to perform the required mapping and then the number of hidden neurons.

**Procedure 1:** Defining the set of hyper-planes. Let $X$ be a set of points, defined in the input space $\mathbb{R}^{l}$, associated to two classes $A$ and $B$. Let $G_{i} \subset X$ be a sub-set whose points belong to the same class and which is linearly separable from the complementary set $D_{i} = \{X - G_{i}\}$ called residual set. Among the infinitive hyper-planes that separate $G_{i}$ from $D_{i}$, the optimal hyperplane [13] is considered, which maximizes the distance from the nearest point. This optimal hyper-plane will be indicated with $OH(G_{i},D_{i})$. The $OH(G_{i},D_{i})$ coefficients are used as connections weights between the input layer and the first hidden neuron $FN_{1}$, and the constant term is used as connection weight with the bias. Let us now consider a sub-set $G_{2} \subset D_{1}$, whose points belong to the same class, and which is linearly separable from its residual set $D_{2} = \{D_{1} - G_{2}\}$. The hyper-plane $OH(G_{2},D_{2})$ gives us the second hidden neuron $FN_{2}$. Let us iterate the procedure until the residual set contains points belonging to the same class. The corresponding $OH(G_{\Lambda},D_{\Lambda})$ completes the input space mapping and the associated hidden neuron is the last one of the hidden layer.

In order to define the succession of sub-sets of points $\{G_{1},G_{2},...,G_{\Lambda}\}$ we firstly consider two polytopes, each of them defined by the points belonging to one of the two classes. The facets of the polytope identify a convex polyhedron and they can be represented by a set of linear inequalities. This polyhedron will be called Polyhedron of Minimal Volume ($PMV$), and the $PMV$ that includes the points of a set $S$ will be indicated as $PMV(S)$. Determining the PMVs is a well-known problem [15]. In this work an original implementation is used in order to achieve a reduced computation complexity. Such procedure is qualitatively described in Appendix 4.A.

The knowledge of the PMVs gives us useful information on choosing the sub-sets of points $\{G_{1},G_{2},...,G_{\Lambda}\}$. In fact, if the PMVs of two sets do not intersect, the two sets are linearly separable, and, in order to search for the unique $OH$, we can simply consider the vertices of the two polyhedrons. On the contrary, if the two polyhedrons intersect themselves, more than one
hyper-plane must be used to separate the two sets. Also in this case we can refer to the PMVs to decide how to select the sub-sets of points. Fig. 4.1 graphically shows the procedure.

Let us denote $A$, $B$ the two classes and $S_A$, $S_B$ the corresponding set of points. Let $C$ be the intersection $PMV(A) \cap PMV(B)$, indicated in Fig. 1 by the chequered area. In the first step of the procedure, the minimum number of facets of $C$ is chosen that separate $C$ from all the points of $S_A \cap S_B$ that do not belong to $C$ (these facets are $a_4$-$a_5$; $b_1$-$b_2$; $b_2$-$b_3$ in the example in Fig. 1).

Each facet identifies a sub-set $G_i$ of points belonging to the same class (A or B) and that are linearly separable from the residual set. The OHs are identified referring to these subsets $G_i$. In Fig. 1 OH1 linearly separates $G_1 = \{b_1; b_3; b_4; b_5\}$ from the residual set; OH2 linearly separates $G_2 = \{a_1; a_2; a_6; a_7\}$, and OH3 linearly separates $G_3 = \{a_4; a_9; a_{10}\}$. Note that the final result is that the intersection $C$ is separated from the points that are external to $C$. The procedure has to be recursively applied to the points that fall in the intersection $C$, until two linearly separable sets $S_A$ and $S_B$ are obtained. In Fig. 1 the new sets $S_A$ and $S_B$ are $\{a_4; a_5; a_6\}$ and $\{b_2\}$ respectively, and such sets are linearly separable each other. They constitute the last subsets $G_A$ and $G_{A+1}$ thus $D_A \equiv G_{A+1}$. The determination of OH4 ends the procedure.

**Figure 4.12 – Graphical description of Procedure 1**
Remark 1. Using the facets of PMVs to determine the groups \( \{G_1, G_2, \ldots, G_\Lambda \} \) ensures to obtain a neural network whose size is less than the upper bound of \( M-1 \) hidden neurons, where \( M \) is the number of training points. Nevertheless, such choice does not guarantee to obtain the minimal number of hidden neurons. This minimal network could be calculated by considering all the possible combinations of the facets. This choice would combinatorially increase the computational cost without assuring significant reduction of the network size.

4.3.2.2 Theorem 1 – Input space mapping.

The previous Procedure 1 leads to a layer of neurons that maps the training points in a feature space, where we have supposed the classes be linearly separable. Because of the kernel function, the points in the input space correspond to vertexes of a hypercube in the feature space. The following Theorem 1, at the same time, demonstrates that the used input space mapping renders the two classes linearly separable in the feature space, and furthermore gives us a method to immediately calculate the hyper-plane in the feature space. To this end we give the following definitions.

Definition 1 – Let \( S \) be a non empty set of elements divided in two classes, and \( G \) be a non-empty sub-set of \( S \) constituted by elements that belong all to the same class. Let \( FN \) be a classifier that separates \( G \) from its complement in \( S \). \( D = [S - G] \) denotes the complement of \( G \) in \( S \). \( D \) is called residual set.

Definition 2 – Let \( \{FN_\lambda\} \) be a succession of classifiers according to Definition 1, such that the classifier \( FN_\lambda \) is applied to the residual set associated with the preceding classifier \( FN_{\lambda-1} \). Let the elements of the last residual set all belonging to the same class. The set \( \{FN_\lambda\} \) is called separating succession.

Based on the above definitions the following Theorem can be stated:

Theorem 1 – Let \( X \) be a finite set of elements divided in two classes. Any set of classifiers that fulfills a separating succession according to Definition 2 maps \( X \) into a space where the two classes are linearly separable.

Proof. Let \( \{FN_\lambda\} \) be a separating succession for the set \( X \) and let \( \Lambda \) be the number of classifiers of such succession. Let 0 and 1 be the values that the classifiers associate to the elements of the two classes. The set \( \{FN_\lambda\} \) maps the elements of \( X \) on the vertexes of a unitary hyper-cube in the space.
$\mathbf{R}^\Lambda$ defined by the coordinate system $\{h_\lambda : \lambda = 1, \ldots, \Lambda\}$. Each classifier of the succession is associated with the sub-set that it separates from the corresponding residual set. Hence, the succession is associated with the succession of $\Lambda + 1$ sub-sets $\{G_\lambda\}$. Let $H_\lambda$ be the set of points in $\mathbf{R}^\Lambda$ which corresponds to the set $G_\lambda$. We will call $H_\lambda$ the image of $G_\lambda$ in the space $\mathbf{R}^\Lambda$.

The last two sub-sets of the succession, i.e., $G_\Lambda$ and $G_{\Lambda+1}$, belong to all the residual sets from the first one to the last but one. Therefore, none of the first $\Lambda - 1$ hyper-planes separates them, hence their images $H_\Lambda$ and $H_{\Lambda+1}$ have the same value of the first $\Lambda - 1$ coordinates in $\mathbf{R}^\Lambda$. Furthermore, the region where $G_\Lambda$ and $G_{\Lambda+1}$ lie, is split only by the last hyper-plane, so each of the two parts of such region corresponds to a unique vertex of the unitary hypercube in $\mathbf{R}^\Lambda$.

$H_\Lambda$ and $H_{\Lambda+1}$ are linearly separable, e.g., by using the following hyper-plane:

$$h_\Lambda = 0.5$$ (4.4)

Let us now suppose to know the equation of the hyper-plane that separates the two classes of points belonging to the set $\Xi_{\Lambda,\Lambda+1} = \bigcup \{H_{\Lambda-\Lambda+1}, \ldots, H_\Lambda, H_{\Lambda+1}\}$:

$$f_{\Lambda-\Lambda+1}(h) = 0$$ (4.5)

The hyper-plane that separates the two classes of points belonging to the set $\Xi_{\Lambda,\Lambda-1} = \bigcup \{H_{\Lambda-\Lambda-1}, \Xi_{\Lambda,\Lambda+1}\}$ has to be determined. Let $p_{\Lambda-\Lambda} \in \{0, 1\}$ be the value that $FN_{\Lambda-\Lambda}$ gives to the points of the associated sub-set $G_{\Lambda-\Lambda}$, whose image is $H_{\Lambda-\Lambda}$. Let $\overline{p}_{\Lambda-\Lambda} = 1 - p_{\Lambda-\Lambda}$ be the complement of $p_{\Lambda-\Lambda}$. All the points of $H_{\Lambda-\Lambda}$ lie on the following hyper-plane:

$$h_{\Lambda-\Lambda} = p_{\Lambda-\Lambda}$$ (4.6)

As $\Xi_{\Lambda,\Lambda+1}$ is the image of the residual set associated with $FN_{\Lambda-\Lambda}$, all the points of $\Xi_{\Lambda,\Lambda+1}$ lie on the hyper-plane whose equation is:

$$h_{\Lambda-\Lambda} = \overline{p}_{\Lambda-\Lambda}$$ (4.7)

Let us now consider the family of the hyper-planes generated by linearly combining the planes of equations (4.5) and (4.7):

$$w_{\Lambda-\Lambda} \cdot (h_{\Lambda-\Lambda} - \overline{p}_{\Lambda-\Lambda}) + f_{\Lambda-\Lambda+1} = 0$$ (4.8)
For each $|w_{\lambda^{-1}}| < \infty$, the equation (4.8) corresponds to a hyper-plane that separates the points of different classes belonging to $\Xi_{\lambda^{-1}+1}$. In fact, for these points, equation (4.8) becomes equation (4.5), which correctly classifies the points of $\Xi_{\lambda^{-1}+1}$. In general, by setting $w_{\lambda^{-1}} = 0$, the hyper-plane (4.8) could cut the set $H_{\lambda^{-1}}$, i.e., the points of $H_{\lambda^{-1}}$ could lie in different half-spaces. Moreover, for $|w_{\lambda^{-1}}| = \infty$, equation (4.8) becomes equation (4.7), i.e., it corresponds to a hyper-plane parallel to the one where all the points of $H_{\lambda^{-1}}$ lie on, and, hence, it does not cut $H_{\lambda^{-1}}$. Therefore, a value of $|w_{\lambda^{-1}}| < \infty$ exists, such that, for any greater value, the hyper-plane (4.8) is able to separate the two classes of points that belong to $\Xi_{\lambda^{-1}}$.

By iterating the above procedure, the hyper-plane that separates the classes belonging to $\Xi_{1}$ is defined.

**Remark 2.** Theorem 1 justifies the strategy described in this section concerning the mapping of the input space. Indeed, following the cited strategy, the hidden layer represents a separating succession for the training set, according to Definition 2. Hence it maps the points of $X$ in such a way that the images of the two classes are linearly separable.

**Remark 3.** Theorem 1 can be applied even if the separating succession is made up of classifiers different from the formal neurons considered in the present chapter. In fact, the only hypothesis made on the classifiers of the separating succession is that their outputs have to be binary.

### 4.3.2.3 Feature Space Mapping

In order to calculate the connections weights between the hidden layer and the unique output neuron, we could apply the same technique we used in the input space. However, by using the proof of the Theorem 1 a direct method can also be obtained.

Equation (4.4) represents a hyper-plane that separates the two images $H_{\lambda}$ and $H_{\lambda+1}$. In particular, it locates the image with coordinate $h_{\lambda} = 0$ in the negative half-space, while the other image is located in the positive half-space. The output neuron associates these two half-spaces to the values 0 and 1, respectively.

By using equation (4.8), the hyper-plane (4.4) can be adapted step by step to perform the desired separation in the feature space. The coefficients of the hyper-plane collected at the last iteration are used as connection weights between the hidden layer and the output node. In the following, the iterative procedure will be detailed.
Let us suppose that the equation (4.5) of the hyper-plane that correctly classifies the set \( \mathbf{H}_{\Lambda, \lambda} = \bigcup \{ \mathbf{H}_{\Lambda, \lambda+1} , \ldots , \mathbf{H}_\lambda , \mathbf{H}_{\Lambda+1} \} \) is known. By using equation (4.8), it is also possible to determine the hyper-plane that keeps all the points of the image \( \mathbf{H}_{\Lambda, \lambda} \) in the correct half-space.

More precisely, if \( w_{\Lambda, \lambda} = 0 \), some points of \( \mathbf{H}_{\Lambda, \lambda} \) fall in the wrong half-space, namely the corresponding value of the left side hand of (4.8) assumes the wrong sign, negative or positive, depending on the class which \( \mathbf{H}_{\Lambda, \lambda} \) belongs to. On the other hand, the term \( (h_{\Lambda, \lambda} - \overline{p}_{\Lambda, \lambda}) \), calculated on the points of \( \mathbf{H}_{\Lambda, \lambda} \), gives a contribution that reduces the said error, as it is negative if \( p_{\Lambda, \lambda} = 0 \), while it is positive if \( p_{\Lambda, \lambda} = 1 \). This implies that, in order to correctly classify the whole set \( \mathbf{H}_{\Lambda, \lambda} \), we will need a positive value of \( w_{\Lambda, \lambda} \), and such value has to be large enough to compensate the sign error of \( f_{\Lambda, \lambda+1} \) for all the points of \( \mathbf{H}_{\Lambda, \lambda} \). In particular, if \( w_{\Lambda, \lambda} \) is chosen such that the largest error of \( f_{\Lambda, \lambda+1} \) is exactly compensated, all the points of \( \mathbf{H}_{\Lambda, \lambda} \) will fall in the right half-space, except at least one of them that lies on the hyper-plane. By assuming larger values for \( w_{\Lambda, \lambda} \), the points of \( \mathbf{H}_{\Lambda, \lambda} \) will fall in the right half-space, and the larger \( w_{\Lambda, \lambda} \), the larger their margin with respect to the hyper-plane, i.e., the distance of the two points from it.

Among the infinite values of \( w_{\Lambda, \lambda} \) for which the points of \( \mathbf{H}_{\Lambda, \lambda} \) are correctly classified, the value corresponding to the hyper-plane with the maximum distance from the nearest point is chosen. This hyper-plane has equal distance from \( \mathbf{H}_{\Lambda, \lambda} \) and from the nearest point of \( \mathbf{H}_{\Lambda, \lambda+1} \). If we know which is the nearest point of \( \mathbf{X}_{\Lambda, \lambda+1} \), we can impose that the hyper-plane has the same distance from such point and from each point of \( \mathbf{H}_{\Lambda, \lambda} \). This leads to as many conditions for \( w_{\Lambda, \lambda} \) as points of \( \mathbf{H}_{\Lambda, \lambda} \), and we will assume the largest value among them.

The nearest point of \( \mathbf{X}_{\Lambda, \lambda+1} \) is the same of \( \mathbf{X}_\lambda \) at the first iteration. In fact, at the \( \lambda \)-th iteration, the position of the hyper-plane depends on \( w_{\Lambda, \lambda} \), but this coefficient does no affect which point is the nearest to the hyper-plane. This is because all the points of \( \mathbf{X}_{\Lambda, \lambda+1} \) zeroes the term \( (h_{\Lambda, \lambda} - \overline{p}_{\Lambda, \lambda}) \) which multiplies \( w_{\Lambda, \lambda} \). Hence, the nearest point of \( \mathbf{X}_{\Lambda, \lambda+1} \) is the same chosen at the first iteration, when the hyper-plane (4.4) has the same distance from the two images \( \mathbf{H}_\lambda \) and \( \mathbf{H}_{\Lambda+1} \), located on two vertexes of the unitary hyper-cube.

For this reason, the point of \( \mathbf{X}_{\Lambda, \lambda+1} \) can always be chosen between the two points \( \mathbf{H}_\lambda \) and \( \mathbf{H}_{\Lambda+1} \) that are the nearest points at the first iteration. Between them, we will select the point that belongs to the same class which the points of \( \mathbf{H}_{\Lambda, \lambda} \) belong to. This point is indicated by \( h_r \), and has the following coordinates:
\[ h_r = (\overline{p}_1; \overline{p}_2; \ldots; \overline{p}_{A-\delta}; \overline{p}_{A-\delta+1}; \ldots; \overline{p}_{A-1}; p_{A-\delta}) \]  
\[ (4.9) \]

Moreover, the generic point of \( \mathbf{H}_{A-\delta} \) has coordinates \( h_{A-\delta} \equiv (\overline{p}_1; \overline{p}_2; \ldots; \overline{p}_{A-\delta}; \xi_{A}) \) where \( \xi_{A} \) is a binary vector of \( \lambda \) components. The vector \( \xi_{A} \) assumes in general all the \( 2^\lambda \) possible combinations. Each point of \( \mathbf{H}_{A-\delta} \) corresponds to one of such combinations; hence the maximum number of points of \( \mathbf{H}_{A-\delta} \) is equal to \( 2^\lambda \). Let us suppose \( \mathbf{H}_{A-\delta} \) have such maximum number of points. For each of them we write an equation that gives a value of \( w_{A-\delta} \). Among such values, the maximum one is chosen. The equations have the following expression:

\[
\frac{w_{A-\delta} \cdot (p_{A-\delta} - \overline{p}_{A-\delta}) + f_{A-\delta+1} (\xi_{A})}{\sqrt{w_{A-\delta}^2 + \sum_{j=1}^{\lambda-1} w_j^2}} = \frac{w_{A-\delta} \cdot (p_{A-\delta} - \overline{p}_{A-\delta}) + f_{A-\delta+1} (h_r)}{p_{A-\delta} - \overline{p}_{A-\delta}}
\]

from which:

\[
w_{A-\delta} = \frac{f_{A-\delta+1} (h_r) - f_{A-\delta+1} (\xi_{A})}{p_{A-\delta} - \overline{p}_{A-\delta}} \sqrt{w_{A-\delta}^2 + \sum_{j=1}^{\lambda-1} w_j^2}
\]

\[ (4.10) \]

The desired value of \( w_{A-\delta} \) is positive, hence it corresponds to the maximum value of the term \(-f_{A-\delta+1} (\xi_{A})/(p_{A-\delta} - \overline{p}_{A-\delta})\). The point \( \xi_{A} \) is that one corresponding to the greatest \( w_{A-\delta} \). More precisely:

\[
\begin{cases}
p_{A-\delta} = 0 & \Rightarrow & S_{A-\delta+1}^+ = \max \left( \sum_{j=1}^{\lambda} c_j w_{A-j} \right) = \sum_{j=1}^{\lambda} w_{A-j} \\
p_{A-\delta} = 1 & \Rightarrow & S_{A-\delta+1}^- = \min \left( \sum_{j=1}^{\lambda} c_j w_{A-j} \right) = 0
\end{cases}
\]

(4.11)

Taking into account (4.9), the general expression of \( w_{A-\delta} \) can be written:

\[
w_{A-\delta} = (-1)^{p_{A-\delta}} \left[ 1 + \sum_{j=1}^{\lambda} \left( w_{A-j} \cdot \overline{p}_{A-j} \right) - \overline{p}_{A-\delta} \cdot S_{A-\delta+1}^+ - p_{A-\delta} \cdot S_{A-\delta+1}^- \right]
\]

(4.12)

To complete the evaluation of the hyper-plane, the constant term, which corresponds to the bias weight, has to be calculated. This can be easily done using (4.8) and (4.10):

\[
\theta = -\frac{1}{\lambda} \sum_{j=1}^{\lambda} \left( w_{A-j} \cdot \overline{p}_{A-j} \right) - 0.5
\]

(4.13)
4.3.2.4 Generalization to K-classes problems

When there are more than two classes in the training set, the binary coding has more than one bit. A single output node is no longer sufficient to perform the desired classification. The number of output nodes depends on the binary coding chosen by the designer. A first choice is to associate each bit of binary code and, hence, each output neuron, to a class. This leads us to have as many output neurons as classes. Otherwise, it is possible to reduce the dimension of the output layer by assuming different binary coding. For example, by choosing a coding with \( \log_2 K \) bits, the minimal dimension of the output layer is obtained.

Once the coding has been given, each output neuron associates the whole training-set to only two classes. This means that the MLP classifier could be seen as made up of sub-networks, each with one output node. Each sub-network splits the points of \( X \) in two classes. The sub-networks have the same inputs, and can share hidden neurons. The choice of the sharing rate of the hidden neurons may have a great impact on the network design computational cost. The design method presented in this paper adopts a compromise between the requirements of limiting the developing time and the network size.

Each bit of the coding is associated with an output neuron corresponding to a sub-network that solves a two-classes classification problem. As mentioned above, the proposed method is based on the concept that each sub-network partially shares some hidden neurons with other sub-networks. The sub-networks are calculated in series. For the first one we apply the previously described procedure to solve the two-classes problem related to the first bit of the coding, without taking into account the problem of sharing hidden neurons. For the i-th sub-network, we evaluate if some neurons of the previous sub-networks can be useful for solving the two-classes problem related to the i-th bit of the coding. To this purpose, it is enough that a hidden neuron separates, from the current residual set, a sub-set which is all constituted by points belonging to the same class. If that happens, the neuron is shared by the i-th sub-network without correcting the position of the corresponding hyper-plane, if not, a new hidden neuron has to be added.

Remark 4. Overfitting is a well known drawback of neural networks training algorithms, and, in particular for MLPs. This phenomenon could be seen as a low generalization capability of the network when it has to handle examples not belonging to the training set. The cause of this behaviour is twofold: the network is trained to reduce the error on the training set, without taking into account the trend of the error on a validation set (overtraining), or the network has too many degrees of freedom, i.e., it has too many hidden neurons and/or connections. While the former can be easily handled, for example by means of a cross-validation technique, the latter is more difficult.
to treat, because it affects the topology of the network. Many approaches, mainly belonging to growing and pruning procedures, have been proposed in literature [16][17] to face this issue, although the pruning procedure usually shows better performances [17].

For classification problems, the effect of having too many connections and nodes can be interpreted, from a geometrical point of view, as an excessive fragmentation of the input space into contiguous regions that the network associates to different outputs. Therefore, the generalization capability of the network is affected by both the input space mapping and how the second layer classifies said regions.

Note that, training algorithms based on error function minimization do not have any chance to determine how to map the input space and to coordinate that mapping with the one of the feature space. As a consequence of this black-box approach, unsuitable solutions can not be avoided, such those associating points close to each other, and belonging to the same class, to different regions. Furthermore, the margins between the training points and the decision surfaces can not be taken under control, so validation points could be misclassified even if they are close to the training points.

In the proposed approach, the two described causes of overfitting are handled separately. Firstly, error on the validation set is taken under control by setting the margins between training points and hyper-planes in the input space, as described in the previous section 4.4.2.1

Secondly, once a group of points has been separated from its residual set (see section 4.4.2.3), a further fragmentation of the region where the group lies on does not affect the classification of the points in such region. This is a consequence of how the second layer is calculated. In fact, the hyper-plane, mapping the feature space, is calculated so that all the possible images of the region (vertices of the hyper-cube) fall on the same side of the hyper-plane. Therefore, whatever the region is partitioned by the next hyper-planes, the resulting sub-regions will be associated to the same network output.

4.4 Results

In this section the performances of the proposed procedure are evaluated and compared with those of other methods presented in literature. The results refer to benchmarks that address the main issues related to the classifiers.

The first benchmark refers to two classes of points randomly distributed in the $\mathbb{R}^2$ space. Each class has 50 points. The aim of the problem is to evaluate the capability of the algorithm to make a correct classification of the training set. The same example has been solved using the LIBSVM
toolbox available on the internet [18] and a neural network trained with the Levenberg-Marquard algorithm (LM-NN), implemented using the neural toolbox of Matlab [19]. The comparison is made in terms of computation time and classifier size. All the three methods perform a correct classification of the training set. The proposed method does not require any tuning of the network parameters, while the other two methods need a trial and error procedure to fix the best parameters. The computation time of the three methods are comparable and of the order of few seconds on a Pentium II 850 MHz machine. Fig. 2 shows how the three classifiers map the input space. Our classifier makes a less regular mapping than the other approaches, but it classifies the training set with 41 hidden neurons, whereas SVM uses 55 kernel functions and LM-NN has 55 hidden neurons.

The second benchmark refers to a well-known classification problem [12], where two sets of points are distributed along two interlocking spirals. Each class has 97 points. We use this problem to evaluate the generalisation capability of the previous methods in classifying points belonging to a set different from the training set. Also in this case we have forced the whole training set to be classified without errors.

Figure 13 - set mapping performed by the proposed method, SVM and LM algorithm.

Fig. 3 shows the mapping of the input space performed by the three classifiers. The test set has been built as in the following: for each interval between two contiguous points of the same class, we consider the middle point on the spiral, obtaining 192 points. All the methods classify the test set without errors. By using the same PC used in the previous benchmark, our method spends 5.67 seconds, the SVM spends 0.84 seconds, and the LM-NN spends 300 epochs.
Figure 14 - Interlocked spirals mapping performed by the proposed method, SVM and LM algorithm.

The sizes of the three classifiers are very different: our network has 31 hidden neurons, SVM has 155 kernel functions, and LM-NN has 80 hidden neurons. Note that a set of M-1 linear separators is enough to correctly classify any training set of M examples. In the present benchmark the upper bound of the hidden neurons (kernel functions) is M-1 = 191. Accepting to misclassify some training examples can reduce SVM classifier size.

The third benchmark refers to a problem (Adult Dataset) retrieved from the UCI Repository [20]. The aim is to forecast the income of a person on the basis of 14 features. The classes are two and the threshold that divides them is 50 k$/year. The complete dataset has 32,561 examples for the training and 16,281 examples for the test. We compared the performances of our method with those of SVMs trained with the Platt's SMO [21] and the NPA algorithms whose results are reported in [22]. For the sake of comparison, we used a Pentium II 200 MHz machine, similar to that used in [22]. The dataset has been divided in the training, validation, and test sets, according to the properties reported in Tab. I. In Tabs. II, III, and IV we compare the performances of our method (PM) with those of SMO and NPA methods. The comparison refers to the computation time, the size of the yielded classifiers, and the percentage of misclassifications on the test set. As SVMs require fixing margin values in the feature space, in Tab. II and III, results refer to three different values of the margin. Note that our method allows fixing the margin value directly in the input space and the corresponding results in Tab. II and III refer to a unique margin value equal to the low margin value used in SMO and NPA. The results suggest that the method proposed in this paper performs as well as the SVMs with respect to the generalization capability, but the sizes of our neural networks and the computation time are smaller than those of the SVMs.

The fourth benchmark refers to a real world problem (breast-cancer) retrieved from proben1 database [23].
The problem consists on classifying a tumor as either benign or malignant on the basis of a set of microscopic examinations. A total number of 699 instances are available. Three partitions of the data are given in proben1, which consist of 350 training examples, 175 validation examples, and 174 test examples. In the present experiment, the first two sets have been chosen as training set, while the test set has been chosen as it is. The input patterns have 9 continuous attributes quantized in 10 levels. The aim of this experiment is to compare the proposed method with pruning NN design techniques (MAG, OBS, OBD, NC, SKEL, EFAST), retrieved from the literature [17]. The pruning methods belong to both weights pruning and node pruning methods. The comparison has been done on the basis of error rate, number of hidden neurons, and number of connections. In Table V the results reported in [17]-Table V are compared with those of the proposed method. Note that, as the proposed network architecture has hard limit output activation functions, the classification error rate better represents the performances of the classifier. For the sake of comparison, in Table V mean squared error (mse) is also reported, as in [17].

As it can be noted, the performance of the proposed method is close to the best ones of the compared method. Note that this result has been obtained without performing a cross-validation procedure, only one training procedure is sufficient, and no trial value of training parameter has to be explored.

In order to evaluate the performance of the proposed method in classifying multi-class problems, we have also considered the real-world Abalone benchmark from the UCI Repository [20]. The Abalone problem has 4177 cases predicting the age of an abalone mollusc from several physical measurements. Each observation consists of eight input attributes and one output (the age of Abalone). This is an extremely non uniform problem, which presents a high degree of class overlapping in the input space. In the literature [24][25][26], the performance of different classification methods (Multilayer Perceptron neural networks, Support Vector Machines, Neural MinMax classifiers) have been tested on the Abalone benchmark. In these references, different training, validation, and test data are randomly generated from the Abalone database to solve the classification problem for different number of classes. Abalone was indeed originally designed to estimate a numeric value, so the original problem has been transformed into an K-classes classification problem by setting suitable thresholds in the target attribute that approximately divided the database into K classes of equal size.

Table 4.VI reports the result of an experiment we performed by assuming K equal 8 classes; the number of samples in the training, validation, and test sets, the dimension of the input vectors, as well as the error rate on the test set are shown in the same table.
For the sake of comparison, the same values are reported for the referenced methods that perform classifications with smaller number of classes [24][25][26].

The results clearly confirm the suitability of the proposed method for multi-class problems.

Table 4.I. Adult Dataset: Properties of the data sets

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Training set</th>
<th>Validation set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult-1</td>
<td>1605</td>
<td>535</td>
<td>535</td>
</tr>
<tr>
<td>Adult-4</td>
<td>4781</td>
<td>1594</td>
<td>1594</td>
</tr>
<tr>
<td>Adult-7</td>
<td>16100</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4.II. Adult Dataset: CPU time in seconds. The three values for both SMO and NPA algorithms correspond respectively to High, Middle and Low margin values.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>PM</th>
<th>PM</th>
<th>SMO</th>
<th>SMO</th>
<th>NPA</th>
<th>NPA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High Margin</td>
<td>Middle Margin</td>
<td>Low Margin</td>
<td>High Margin</td>
<td>Middle Margin</td>
<td>Low Margin</td>
</tr>
<tr>
<td>Adult-1</td>
<td>20</td>
<td>450</td>
<td>100</td>
<td>2500</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>Adult-4</td>
<td>40</td>
<td>1000</td>
<td>1000</td>
<td>44000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Adult-7</td>
<td>50</td>
<td>15000</td>
<td>7000</td>
<td>40000</td>
<td>7000</td>
<td>10000</td>
</tr>
</tbody>
</table>

Table 4.III. Adult Dataset: Sizes of the classifiers. For the neural network the values are the number of hidden neurons. For SMO and NPA the values are the number of support vectors.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>PM</th>
<th>PM</th>
<th>SMO</th>
<th>SMO</th>
<th>NPA</th>
<th>NPA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High Margin</td>
<td>Middle Margin</td>
<td>Low Margin</td>
<td>High Margin</td>
<td>Middle Margin</td>
<td>Low Margin</td>
</tr>
<tr>
<td>Adult-1</td>
<td>80</td>
<td>819</td>
<td>691</td>
<td>661</td>
<td>1602</td>
<td>1034</td>
</tr>
<tr>
<td>Adult-4</td>
<td>150</td>
<td>2204</td>
<td>1886</td>
<td>1851</td>
<td>3385</td>
<td>2999</td>
</tr>
<tr>
<td>Adult-7</td>
<td>200</td>
<td>7855</td>
<td>6014</td>
<td>6166</td>
<td>11950</td>
<td>10181</td>
</tr>
</tbody>
</table>

Table 4.IV. Adult Dataset: Percentage of misclassifications on the test set

<table>
<thead>
<tr>
<th>Data Set</th>
<th>PM Low Margin</th>
<th>SMO Low Margin</th>
<th>NPA Low Margin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult-1</td>
<td>13.2</td>
<td>13.8</td>
<td>14.6</td>
</tr>
<tr>
<td>Adult-4</td>
<td>13.6</td>
<td>14.2</td>
<td>13.4</td>
</tr>
<tr>
<td>Adult-7</td>
<td>13.5</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4.V. Breast Cancer Dataset: number of hidden nodes, number of connections, mse, and error rate
#nodes | PM | MAG | OBS | OBD | NC | SKEL | EFAST
---|---|---|---|---|---|---|---
5 | 110 | 110 | 128 | 22 | 3 | 9
#connections | 56 | 973 | 971 | 1089 | 220 | 30 | 90
Mse | 0.01724 | 0.01433 | 0.01741 | 0.02044 | 0.02723 | 0.02027 | 0.01414
classification error rate | 3/174 | - | - | - | - | - | -

Table 4.VI. Abalone Dataset: number of samples in the training, validation, and test sets; number of classes; dimension of the input vectors, mse on the test set

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Train. size</td>
<td>869</td>
<td>3133</td>
<td>1670</td>
</tr>
<tr>
<td>Valid. size</td>
<td>-</td>
<td>-</td>
<td>1670</td>
</tr>
<tr>
<td>Test size</td>
<td>3300</td>
<td>1044</td>
<td>837</td>
</tr>
<tr>
<td># Classes</td>
<td>8</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td># Features</td>
<td>5</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Mse</td>
<td>0.22</td>
<td>0.38</td>
<td>0.21</td>
</tr>
</tbody>
</table>

4.5 Conclusions and comments.

The results showed that the proposed approach is faster than the compared methods in synthesizing the classifier, in particular when the training set has a great number of examples, this will be useful once applied to JET database as a large number of patterns have to be classified and predicted as well (see chapter 7 for details). Furthermore, the yielded classifiers are smaller and the robustness in the classification can be easily fixed a priori.

**Appendix 4.A – Polyhedron of Minimal Volume (PMV)**

Let S be a finite set of points defined in an n-dimensional space $\mathbb{R}^n$. The procedure constructing the envelope of S firstly determines an initial polyhedron of $n+1$ facets (simplex) whose vertexes belong to S (see procedure 4.A.1). If some points of S do not fall into that polyhedron it will be iteratively enlarged in order to include them (see procedure 4.A.2). At each iteration an external point is included in such a way it becomes a vertex of the enlarged polyhedron. For a space
dimension $n>3$ that enlargement produces polyhedrons whose number of facets combinatorially grows with the number of vertexes. Thus not all the points have to be considered in the iterative procedure, but only those that will be vertexes of the final polyhedron $PMV(S)$. The set of such vertexes are selected as described in the same procedure 4.A.2.

A further procedure (procedure 4.A.3) is introduced to handle sets of coplanar points, namely, sets of points whose $PMV$ has a null volume.

Procedure 4.A.1: Defining the Initial Polyhedron. Let us firstly consider $n$ hyper-planes intersecting in an arbitrary point of $S$ (point 1 in Fig. 4.4.a). Secondly, let us combine $n-1$ pairs of such hyper-planes, in order to obtain as many hyper-planes, passing through the point 1 and through a second point of $S$ (point 2 in Fig. 4.4.b). The $n-1$ hyper-planes are then combined in order to capture a third point (point 3 in Fig. 4.4.c), and so on until $n$ points are captured, or all the points of $S$ fall into the intersection of the set of the hyper-planes.

If $n+1$ non-coplanar points exist in $S$, only one hyper-plane is kept. This is the first facet of the initial polyhedron, and the $n$ captured points will be vertexes of such polyhedron.

In order to obtain a closed polyhedron, $n$ facets have to be determined. To this end, a point $P$ of $S$, not coplanar with the first facet, is selected.

![Figure 15 - Definition of the first facet of the initial polyhedron in Procedure A.1](image)

By performing $n$ rotations of the first facet around its sides until the point $P$ is reached, all the facets closing the initial polyhedron are obtained.

Procedure 4.A.2: External Points Inclusion. Let us suppose the initial polyhedron does not contain all the points of $S$. Let $\{P_m \in S, m = 1, \ldots, M\}$ be the set of external points. For each $P_m$ let
\{L_{j,m} \mid j = 1,\ldots,J\} be the set of facets not fulfilled by it. For each pair \{P_m, L_{j,m}\}, the violated constraint is linearly combined with its n adjacent facets, in order to obtain n new facets that pass through \(P_m\). These new facets substitute \(L_{j,m}\) in the polyhedron. If some new facets cross the current polyhedron, they are removed. Such facets can be easily recognised because they are violated by at least one vertex of the polyhedron.

The procedure is iterated until all the points are included in the polyhedron.

Note that, the inclusion of a point can have the effect of including some other points, which will not be considered in the following iterations.

In order to reduce the computational cost, in the including procedure, only vertexes of the final PMV(S) has to be considered. To this end, at each iteration, it is sufficient to select the point with the maximum distance from the violated facets of the current polyhedron.

Since the aim of the enveloping procedure is to facilitate the identification of the optimal hyper-planes (see section III.A), it is convenient to substitute only the violated facets, which separate points belonging to different classes. That expedient further limits the computational cost of the procedure.

Procedure 4.A.3: Defining PMVs of Null Volume. If \(n+1\) non-coplanar points do not exist in \(S\), the initial polyhedron cannot be defined by means of the Procedure 4.A.1. In this case, after a certain number \(k\) of iterations of the Procedure 4.A.2, the whole set \(S\) will fall into the intersection of the following \(n-k\) hyper-planes:

\[ \langle w_i, x \rangle = \theta_i \quad \text{with} \quad i = 1,\ldots,n-k \quad (A.14) \]

These equations identify an unlimited sub-space \(\mathbb{R}^k\), where the points of \(S\) are not longer coplanar; hence Procedures 4.A.1 and 4.A.2 can be now applied to calculate the PMV(S) in the sub-space \(\mathbb{R}^k\). The same inequalities, which represent PMV in \(\mathbb{R}^k\), identify a prism in \(\mathbb{R}^n\). The PMV(S) in the space \(\mathbb{R}^n\) is given by the intersection between that prism and the sub-space \(\mathbb{R}^k\).
Reference


Chapter 4 – Kernel Machines – Geometrical approach


    http://www1.ics.uci.edu/~mlearn/MLRepository.html


Chapter 5  Novelty Detection

5.1 Introduction.

Novelty detection (ND) is the ability of an autonomous system to identify abnormal or irregular information that deviates significantly from the learnt, ‘normal’ model. There has been considerable researches in the past two decades in novelty detection and these techniques are beginning to be used in a wide range of different application domains [1].

A challenge for a ND system is to detect not only when an item is relevant to the user’s information need, but also when it contains something new the user has not seen before [2].

Almost all statistical approaches dealing with ND are based on modeling the probability density function (pdf) of the training data, this could be a problem when applying to real-world domains as a priori assumptions on pdf has to be provided.

In this thesis we concentrate on ND applied to disruption prediction and an application based on JET database will be given in the section 7. This area represents a significant challenge in the light of the next fusion reactor ITER, as a limited number of disruption can occur in that device, anyway we also try to give a comprehensive approach to ND theoretical background.

5.2 Theoretical Background.

Given a set of independent identically distributed training samples $x_1, \ldots, x_n \in X \subseteq \mathbb{R}^N$, drawn from a probability distribution in feature space, $P$, the goal of novelty detection is to determine the “simplest” subset, $S$, of the feature space such that the probability that an unseen test point $x'$, drawn from $P$, lies outside of $S$ is bounded by an a priori specified value, $\nu \in (0,1]$.

In the one-class formulation, data are first mapped into a feature space using an appropriate kernel function and then maximally separated from the origin using a hyperplane. The hyperplane parameters are determined by solving a quadratic problem, similar to the basic Support Vector Machines case [3]:

$$\min \left( \frac{1}{2} \| w \|^2 + \frac{1}{N} \sum_{i=1}^{N} \xi_i - \rho \right)$$

subject to:
(5.2)

where \( w \) and \( \rho \) are hyperplane parameters, \( \Phi \) is the map from input space to feature space, \( \nu \) is the asymptotic fraction of outliers (novelties) allowed, \( l \) is the number of training instances, and \( \xi \) is a slack variable.

During the training, feature vectors, \( q \), can be used to estimate the covariance matrix \( \Sigma \), and the mean \( \mu \), of the training data. The degree of novelty is then determined by evaluating the Mahalanobis distance:

\[
D_M(q) = \sqrt{(q - \mu)^T \Sigma^{-1} (q - \mu)}
\]  

(5.3)

between each sample in the training data and the centroid of the training set. During testing feature vectors are thresholded to produce an output according to the following:

\[
y[n] = \begin{cases} 
+1, & D_M(q_n) < K \\
-1, & D_M(q_n) \geq K 
\end{cases}
\]  

(5.4)

if the distance \( D_M \) is greater than the threshold \( K \), \( y[n] \) can be evaluated as novel.

Another model proposed in [4] regards the minimization of the number of heuristically chosen thresholds in the novelty decision process. The novel can be quantified by modelling the unconditional probability of density of a set of data and subsequently evaluating this density for all new data points. If \( P(x) \) is the unconditional density of a set of data a threshold is set to reject all new data points for which \( P(x) \) falls below this threshold. Applying a threshold means to classify all new data in one of two classes: those similar to the old set belonging to class \( C_1 \) and those which are novel denoted by class \( C_2 \). Knowing that:

\[
P(C_1) + P(C_2) = 1
\]  

(5.5)

given a new set \( S \) and assign it one of the two classes in such a way to minimize the probability of misclassification, to be more precise the set \( S \) is assigned to the class having the larger posterior probability, so that if \( S \) is assigned to the class \( C_1 \):

\[
P(C_1|S) > P(C_2|S)
\]  

(5.6)

and to the class \( C_2 \) otherwise Hence for 5.5 is:

\[
P(C_1|S) + P(C_2|S) = 1
\]  

(5.7)

For Bayes theorem, the posterior probability of the set \( S \) to belong to the class \( C_i \) is given by:
Chapter 5 – Novelty Detection

\[
P(C_i|S) = \frac{P(S|C_i)P(C_i)}{\pi(S)} \tag{5.8}
\]

where \(\pi(S)\) is the unconditional pdf given by the following:

\[
\pi(S) = P(S|C_1)P(C_1) + P(S|C_2)P(C_2) \tag{5.9}
\]

Combining 5.6 with 5.8 we obtain:

\[
p(S|C_1) > \frac{P(S|C_2)P(C_2)}{P(C_1)} \tag{5.10}
\]

and otherwise for the class \(C_2\). The eq. 5.10 represents the probability density from which the training data were drawn, while \(p(S|C_2)\) is the distribution of novel data. The condition 5.10 is equivalent to the threshold mentioned above, dividing the space into two regions (known and novel). Fig.1 shows the novel regions where the threshold is triggered for \(p(S|C_1)P(C_1) > p(S|C_2)P(C_2)\).

An improvement of 5.10 is to consider different thresholds for each class [5]. This approach is useful when probability distributions of patterns are not completely known.

The scheme presented by Parra et al [6] evolves the concept of threshold using an hyper-sphere, novel objects should ideally fall outside the hyper-sphere.

As said above, many statistical approaches dealing with ND are based on modeling the density of the training data and rejecting test patterns that falls in low density regions [7] and in some techniques no a priori assumption is done on the statistical properties of data, this is the case of k-nearest neighbor (kNN), Parzen density estimation and string matching approaches.
In [8] to perform an ND, the distribution of normal vectors is described by a small number of spherical clusters placed by the kNN technique. The novelty is assessed by measuring the normalized distance of a test sample from the cluster centres.

With the Parzen density estimation [6, 9, 25, 26, 34] kernel density approximation of a sample probability density function is given by:

$$\hat{f}_h(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x-x_i}{h}\right)$$

(5.11)

where $x_i \in \mathbb{R}$ is a sample of a random variable, $K$ is the kernel function (often a Gaussian with a mean zero and variance $\sigma^2$) and $h$ is the smoothing parameter.

In Fig.2, six Gaussians (dotted lines) and their sum (continuous lines) are shown. The Parzen window density estimate is obtained by dividing this sum by the number of Gaussians. The variance of the Gaussians was set to 0.5. Note that where the points are denser the density estimate will have higher values.

String matching approaches are based on treating data as templates by a string (vector of features) and on finding some measure of dissimilarity between training and test data.

In [10] a matching criterion considers only a certain number of matches in the string (it can be considered as a sort of threshold mentioned above).

A further extension of [4] presented in [11] regards multi-class approach. The unknown space will be then classified into multiple classes to determine the level of abnormality. The normal behavior of a system exhibits stable pattern when observed over time while abnormal ones present deviation from maximum and minimum values of the monitored patterns.

The multi-class approach can be also evaluated from the cluster partitioning point of view, where data points are divided into a number of clusters and a degree of membership to each cluster is given. That membership determines a threshold that indicates if a pattern belongs or not to a cluster, novelty can be detected when the sample does not belong to any cluster [12][13].
5.3 Neural Network approaches - State of the Art

For what concern neural networks approach if compared to statistical ones, some issues for ND are more critical such as their ability to generalize and the expense when they need to be retrained. To retrain is not compulsory when a new class is presented to the net and novel data added to the training set, this is needed only when new data is not longer representative of the environment already trained. For example, retraining of Multi Layer Perceptron (MLP) neural networks after ND means to add new output and hidden nodes and it is not clear how to best train the new configuration, while retrain a SVM after a ND means to re-cross-validate its parameters and this could have a big impact on time performance [14].

Anyway, in this section we are focusing on techniques rather than the application itself, while in the section 7 we provide an application of ND based on geometrical synthesis of MLP neural networks and SVM.

Moya et al.[15] provides three generalization criteria that can be used to asses the performance of a novelty detector. Most pattern classification algorithms and neural network fail at automatic detection of novel classes because they are discriminators rather than detectors. They often use open decision boundaries, such as hyper-planes, to separate targets from each other and fail to decide when a feature set does not represent any unknown class. The authors also tried an Adaptive Resonance Theory (ART) where hyper-spheres surround training classes and produce decision boundaries, the difference between this method and those ones mentioned below is in the number, position and size of the hyper-spheres.

Zhang and Veenker [16] introduce a new active learning paradigm enabling a neural network to adapt itself to its environment, while the latter is changing, by self-generating novel training samples with the recombination of two parent examples of the existing training set.

Bishop [17] affirms that one of the most important sources of errors in neural networks comes from novel data, so to overcome this drawback he suggests to measure the confidence of the decision the network has to take on novel patterns, refusing to make that decision if that pattern belong to a completely new class. The ND is implemented by estimating the density of training patterns and modeling its distribution to check when an input data comes from it or not. With this approach the accuracy on determining the density has to be very high, so that if a new pattern falls into a well-represented region the network will be able to classify it, on the contrary if a data falls into a low density region there is a chance for the net to recognize a class not already represented, but there is also the same chance not to recognize it.

In [18] Vasconcellos suggests how to overcome the problem of open boundaries by re-training the classifier with ‘negative’ examples chosen from those test patterns rejected by the trained net. The
rejection occurs if the output for all neurons is close to 0 or the response of more than one neuron is close to 1. The target output of a rejected pattern is set to 0.

If the activation function of a MLP is a Gaussian activation function (GMLP), the receptive field of each network’s unit corresponds to a hyper-hill in the pattern space which prunes the unit to respond only to part of the half space causing more confined regions surrounding the training data.

In the case of a Radial Basis Function (RBF) the network’s output reaches its maximum when the input pattern is close to a centroid and decreases when its distance from the centroid rises, inputs very dissimilar to train pattern tend to receive very low output. For this reason RBFs are more suitable for ND than GMLP or MLPs in practical applications [19].

In [20] a pattern should be rejected if it is significantly different from the training data and/or it lies in the overlapping region of two or more classes with the definition of a threshold. De Stefano et al. [21] extend the techniques presented in [20] to other types of neural network classifiers and determine the optimal threshold defining a cost function associated with the classifier’s correct recognition rate. They also render the threshold independent of the architecture of the classifier.

Wilson et al. [22] demonstrate that an MLP can have better ND performance by making important changes in the network’s optimization strategy. The regularization can be used by decreasing the volume of the weight space in the optimization process, the sigmoidal activation function can be substituted by a sinusoidal one (this improves the dynamic of the network and a better error-reject performance is achieved), the third strategy regards Boltzmann pruning to reduce the weight space dimension and the class error weight.

As mentioned in the previous section, SVM are based on the concept of determining optimal hyperplanes to separate data belonging to different classes. In [23][24] the authors try to solve the problem of ND by encompassing almost all the points in the data set with a sphere of minimal radius. A pattern is rejected if its distance from the centre of the sphere is greater than the radius. The advantage of this technique if compared to that proposed by Tarassenko [25] is that it does not need probability density estimation, but a drawback is, according to the author, that the method needs a large dataset, especially when high dimensional feature vectors are used.

In [26] Schölkopf et al. offer an alternative to the approach of [23][24], instead of trying to find an hyper-sphere that includes most of the data they try to separate regions containing data from those empty ones. An hyperplane separates point from the origin and a binary function returns +1 in small regions containing data , -1 elsewhere. The data is mapped in the feature space corresponding to the kernel and separated from the origin with maximum margin. All data close enough to the origin are considered as noise or outliers. The drawback of this approach is that the origin plays a crucial role as it acts as a prior for where the class abnormal instances are assumed to lie.
Campbell and Bennet [27] base their approach on the statistical analysis of data modeling the distribution with a binary function, positive in those regions of input space where most of the data lies and negative everywhere else. The separating hyperplanes in the feature space will be positive on one side and negative on the other one. The aim of the method is to find a surface that wraps around the data, anything lying outside that surface is considered novel.

In [28] a set of closed decision regions encompassing the training patterns from the various object classes is estimated. A large margin partition of the input image feature is learnt by minimizing an objective function, once the initial partition is learnt a rejection region is defined by estimating some thresholds that yield a given class conditional probability of detection on a validation set. All images lying in the rejection region will be rejected. For video sequences the novel is recognized if the image shows a degree of confusion.

In [29] the authors perform the ND by inspecting the SVM decision function value. As values of the decision function close to +1 or to -1 correspond to correct answer of the SVM predictor, two regions can be introduced to identify novel samples, if the pattern falls outside that regions it can be considered as novel. The region are chosen such that the decision function falls within that two regions for the 90% of the training samples.

For what concern Self Organizing Maps (SOM) proposed by Aeyels [30], the author provides proof and clarifies some points regarding the convergence properties of the novelty detector and novelty filter described by Kohonen [31]. The approach is unsupervised and no a priori information on class patterns is needed. Yet, the approach is similar to statistical clustering ones, the membership of a form to a certain cluster is determined by a threshold, if the form has got more than one neighbor it is supposed to belong to the class representing the neighbor itself. The author describes two types of novelty detectors: novelty detector without forgetting and novelty detector with forgetting. In the first case the system can only memorize patterns if it is frequently exposed to them, hence if a novel is presented to the SOM it will cause a reaction on the output but only if it is represented several times there is a change to be recognized. Regarding the second one, the novelty filter has got a forgetting term that forces the system to habituate and reduce its response when similar pattern are frequently shown.

In [32] the authors employ a SOM to develop a ND technique to detect faults on a fault monitoring application. The SOM provides a domain description and no statistical information of patterns is given, the topology of input is preserved by using k-means algorithm. The ND is easy to reach, once it is trained patterns from normal operation generate small distance while abnormal one generate large distance.
Another approach based on the activation of Kohonen nodes, shows that faulty features will excite different nodes than healthy ones [33], this method does not take into account the Euclidean distance between the activated neurons.

Despite of difficulties, Neural networks offer a great help when a black-box approach is needed, when the statistical distribution of data is unknown or the physics of the problem is hard to understand.
References


Chapter 6  Disruption prediction at JET

6.1 Introduction

The scientific community considers disruptions as one of the main obstacles to tokamak operation, and, even thought disruptions’ dynamics have been studied for a long time, a complete analysis has not been developed yet.
All works reported in literature are mainly based on observation of, more or less, large database and some important aspects came up as a beginning step to study the phenomenon and to classify it.
During an experimental campaign it is crucial to know disruption’s causes in order to manage the machine parameters to avoid disruption for the successive experiments. At first, in collaboration with some JET physicists, a manual classification has been provided, studying a subset of the entire JET database, and a set of diagnostic signals have been chosen to build an automatic classifier to make the recognition of disruption faster and unambiguous [1].

6.2 Disruptions characteristics and phases.

The manual classification can be divided in three main steps [2]:
- Selection of a set of disruptions
- Selection of diagnostic signals useful to identify the class membership
- Assign the disruption to a proper class membership

JET device is an experimental machine, for that reason, even though a number of similar disruptions are present in its database, the disruptions themselves cannot be grouped together in the same class.

A disruption can be described by a sequence of events summarized in four phases [3]:
Pre-precursor phase: there is an evidence of increasing of plasma density or radiated power, but sometimes the change of the condition is not identified.
Precursor phase: the conditions reach a critical point and there is a onset of magneto hydrodynamic (MHD) instability that grows with varying time intervals until everything is locked. This situation does not mean a new steady state but it is due to a Mode-Lock phenomenon.
Fast phase: once the MHD instabilities are grown enough, a new critical point is reached, during this phase a rapid flattering of the radial current profile is observed, the central plasma temperature collapses on a time scale of the order of milliseconds.

Quench phase: it is characterized by a sudden loss of the plasma energy together with a rapid drop of the electron temperature that leaves the plasma in a high resistive state. Because of that high resistance, the ohmic heating results in an enormous input power of the order of gigawatt. A large amount of energy is transferred to the vacuum vessel leading to forces of hundred tonnes.

Several different types of disruptions can be identified: Density Limit, Low q, β Limit, Vertical Displacement, Mode Lock, High Radiated Power, H-mode/L-mode transition and Internal Transport barrier.

The Density Limit is the most frequent disruption at JET, when the plasma density increases over a certain value or impurities are present in the plasma, an arising of the radiated power is observed and a subsequent drop in temperature. The resistance grows in the plasma edge and the current profile shrinks and this leads to an increased destabilizing current gradient inside the q=2 surface. Because of the inductivity of the plasma, the plasma current quench results in halo currents in the vacuum vessel wall, which lead to very large forces on the material structure of the device [4].

If the radiation is idealized to take place in a narrow layer around a radius \( a_p \) and if \( \varphi \) is the radiated power fraction, the criterion for instability is:

\[
\frac{a_p}{n} \frac{dn}{da_p} > \frac{1}{2} + \frac{1 - \varphi}{2\varphi} \frac{a_p}{a - a_p}
\]  \hspace{1cm} (6.1)

where \( 1/(a - a_p) \) represents the need for the radiating layer to be close to the boundary to conduct the power out of the plasma, \( n \) is the electron density.

For \( \varphi = 1 \) the plasma can be said to be disconnected from the boundary and in this case the criterion for instability becomes:

\[
\frac{a_p}{n} \frac{dn}{da_p} > \frac{3}{2}
\]  \hspace{1cm} (6.2)

This criterion is not easy to apply because of the uncertainty of the term \( da_p/dn \) but it shows the destabilizing effect of the layer moving into a region of higher density to produce increased radiation [3].
For *Low q* disruptions it has been demonstrated that low $q$ surfaces, and in particular $q=2$ ones, play a crucial role in the plasma stability. The surface $q=2$ moves toward the plasma edge becoming unstable with respect to the currents gradient, magnetic islands are then observed and they could be considered as disruption precursors. That magnetic islands occur when unstable magnetic surfaces interact with their neighbors. As we can see from Fig. 1 the direction of the magnetic field over and under the unstable region is opposite, for this reason the line field tears and reconnects forming new localized closed surfaces, so called magnetic islands.

![Figure 18- Magnetic Islands](image)

Knowing that the safety factor $q$ is the ratio between the toroidal and poloidal rotation a magnetic field line performs round the torus before joining up on itself. When, at the plasma surface, $q$ is close to an integer number there could be a chance of the arising of MHD instabilities with the growing of a mode that at the end locks resulting in a disruption [5].

The *High $\beta$* disruptions are related to an increase of the plasma pressure leading to a MHD instability. $\beta$ is a MHD parameter measuring the confined pressure so it is expected that $\beta$ is directly correlated to stability limitation, by the way at JET it is very hard to obtain the condition to reach the $\beta$ limit disruptions.

For what concern the *Vertical Limit* disruptions, they are related to high elongated plasmas. To have that configuration a quadrupolar field has to be applied, that field will generate a vertically unstable configuration. Although an ad hoc system to control the vertical plasma speed has been implemented, because of its limit in capability of controlling that vertical speed, sometimes it fails and a vertical disruption occurs. The plasma vertical position is then lost and it moves to the walls at a very high speed [5].

In a *Mode Lock* disruption frequent MHD perturbations are observed, due to pressure gradient or external magnetic field, which disturbs the plasma structure generating non-rotating magnetic islands. These islands spontaneously grow to large amplitude, ultimately leading to disruption. The occurrence of this type of disruption is most common during the low density startup phase of Ohmic discharges [6] [7].
The *High Radiated Power* disruption is connected to plasma impurities, the behaviour is similar to density limit case but an increase of radiated power is observed some ms before the increasing of density.

The class of disruption known as *H-mode/L-mode transition* occurs when passing from an high confinement regime (H-mode) to a low confinement one (L-mode). That transition normally happens at high density, and the birth of instabilities, which lead to a disruption, are generated by a too fast transition.

In the case of *Internal Transport barrier* when operating in H-mode confinement regime, the particles and energy transport at the plasma edge noticeably decrease. The resulting pressure gradient causes a slowdown of plasma activity, hence leading to a disruption [2],[8].

5.3 **Diagnostic signals**

JET is operated from an on-site central control room via an extensive multi-computer system (CODAS) which handles control, monitoring, data acquisition, and data archiving. It operates, as all tokamaks, in a pulsed mode with each pulse lasting up to several tens of seconds and a repeat frequency of typically one pulse every half-hour. Machine setting is handled via a high-level parameter setting tool, which includes extensive parameter consistency checks and operator guidance. During the pulse, typically 600 Mbytes of JET Pulse File (JPF) raw data are collected in local memory (see Fig.2), and then transferred after the pulse, or at the very latest by the end of the operational day, to a central data store line. As soon as the raw data collection process has concluded an automatic analysis task (Intershot Analysis) is launched and Processed Pulse File (PPF) are then produced. PPFs may exist in several versions. PPFs are also subject to a validation process organized by a Data Co-ordination Committee that may authorize for further recalculation[9].
All the signals used for the purpose of this thesis are JPFI in order to work with all developed tools in real-time way. Nine diagnostic signals have been chosen to describe the plasma regime during the discharge flat-top, according to JET physicists, validated with salience and sensitivity analysis and several previous experiences [1,2,5,7,8,10,11,12]. The choice of the signals takes into account physical considerations and the availability of real-time data. These signals have also demonstrated to be useful for classification purposes as they properly represent the plasma state. Table 1 shows the selected diagnostic signals:

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Plasma current</td>
<td>[A]</td>
</tr>
<tr>
<td>2. Locked Mode</td>
<td>[T]</td>
</tr>
<tr>
<td>3. Radiated power</td>
<td>[W]</td>
</tr>
<tr>
<td>4. Plasma Density</td>
<td>[n/m²]</td>
</tr>
<tr>
<td>5. Input Power</td>
<td>[W]</td>
</tr>
<tr>
<td>6. Internal Inductance</td>
<td>[AU]</td>
</tr>
<tr>
<td>7. Safety factor</td>
<td>[AU]</td>
</tr>
<tr>
<td>8. Poloidal Beta</td>
<td>[AU]</td>
</tr>
<tr>
<td>9. Plasma centroid vertical position</td>
<td>[m]</td>
</tr>
</tbody>
</table>

Another important aspect of those selected signals is that they represent global tokamak devices parameters, hence they do not depend on specific machines.

6.4 Data base selection

The diagnostic signals for training and testing both the SVMs and the geometrical approach have been selected in the pulse interval 47830 - 57346, produced at JET between March 1999 and October 2002. The database consists of 172 disruptive pulses and 102 non disruptive pulses. The pulses belong to the following different classes of disruption: Mode Lock, Density Limit, High Radiated Power, H-mode/L-mode transition, Internal Transport barrier [5] the remaining ones, mentioned in the last section, are not considered as they rarely occur, or do not occur at all, in JET device or they are provoked by the control system.
The discharges included in the network database satisfy the following requirements:

- Plasma current $I_{pla}>1.5$ MA;
- X-point configuration;
- Stationary (Flat-top) plasma current profile.

Discharges with $I_{pla}$ below 1.5 MA were discarded as they generally have little impact on subsequent conditioning and operation of the device.

The sampling time is 20 ms in order to allow the synchronization among different acquisition systems.

In order to train, validate and test the network performance [13], the pulses in the database have been divided in three sets: the training set, consisting of 69 disrupted pulses, the validation set consisting of 17 disrupted pulses, while the test set consisting of 86 disrupted pulses and 102 successful pulses.
References


Chapter 7 Disruption predictors

7.1 Introduction

In this section, an application regarding disruption prediction with SVM and the geometrical approach (see chapt.4 for further details) is given. The characteristics of the database used to train, validate and test the models have been already described in the last chapt. 6.

Anyway, the following considerations have to be taken into account when designing a disruption prediction system [1]:

- the prediction success rate has to be greater than those of the existing alarm systems already available;
- the prediction time has to sufficiently anticipate the starting of the disruption, in order to allow the mitigation and shut-down systems to safely intervene;
- the false alarm rate has to be limited;
- at the same time, an as low as possible missed alarm rate should be obtained;
- the prediction system has to be able to forecast different types of disruptions, characterized by different operational scenarios and dynamics;
- the prediction system has to be able to operate in real-time.

The aim of prediction is achieved by classifying patterns coming from a selection of JET database, predicting the event at least 100ms before the time to disruption.

The performance of the prediction systems are evaluated in terms of percentage of false alarms (PFA), where PFA is defined as the ratio between the number of non disruptive pulses predicted by the system as disruptive shots, and the total number of non disruptive pulses, in percent; and in percentage of missed alarms (PMA), where PMA is defined as the ratio between the number of disruptive pulses predicted as non disruptive shots, and the number of disruptive pulses, in percent.

Note that a disruption prediction is considered successful if the system is able to correctly predict the disruption at least 100 ms prior to the disruption time. Moreover, for disruptive pulses, the percentage of premature alarms (PPA) is defined as the ratio between the number of disruptive pulses predicted by the system too much in advance, and the number of disruptive shots, in percent.

In particular, for JET, the time to premature alarm is estimated around 1 s prior the time of disruption. Finally, the prediction success (PSR) rate is defined, in the present thesis, as the success rate of the predictor in correctly predicting both disruptive and non disruptive pulses.
7.2 Support Vector Machine predictor

In this section a SVM disruption predictor is presented wherein multiple plasma diagnostic signals are combined to provide a composite impending disruption warning indicator. In a SVM the analysis of the decision function value gives useful information about the novelty of an input and, on the reliability of the predictor output, during real time applications. Results show the suitability of SVM both for prediction and novelty detection tasks at JET. In particular, the SVM is trained to classify safe (1) and disrupted (-1) samples.

7.2.1 Novelty detector

As mentioned in the chapt.5 novelty detection (ND) consists of identifying new or unknown data that a machine learning system is not aware of during the training phase. Thus, ND is one of the fundamental requirements of a good classification or prediction system. In fact, actual data may contain patterns belonging to operational regions not explored when the learning system was developed. This could be the case of the disruption predictor presented in this dissertation, where new plasma configurations might present features completely different from those observed in the experiments selected for the training set. This ‘novelty’ can lead to incorrect behavior of the SVM predictor.

In the last ten years novelty detection acquired an increasing attention, and a number of techniques have been proposed and investigated to address it. In [2] [3] the authors highlighted that it is not possible to a priori identify a single best model, and the success of a novelty detection technique mainly depends on the statistical properties of data handled (see chapt. 5 for details).

Both statistic and neural clustering methods can be used for novelty detection tasks. In this thesis, the novelty detection is performed by inspecting the SVM decision function value. As values of the decision function close to +1 or to -1 correspond to correct answer of the SVM predictor, two bands $[1-\Delta_a,1+\Delta_a]$ and $[-1+\Delta_a,-1-\Delta_a]$ can be introduced to identify novel samples. In particular, a sample will be labelled as not novel if its decision function has a value belonging to that bands, and conversely.

The values $\Delta_a$ and $\Delta_b$ are chosen such that the decision function falls within that bands for the 90% of the training samples [4].
In Fig. 1, the decision function values are reported for a test pulse. In the same figure the two bands are highlighted.

The predictive system structure consists of two blocks mutually connected: a Self Organizing Map (SOM) and an SVM (see Fig. 2).

During the training phase, a SOM performs a clustering procedure and an SVM is trained to give the alarm in case of impending disruption. When the training procedure is completed, the previously introduced bands, which discriminate between novel and not novel samples, are calculated.

In order to build the training set for the SVM predictor, 86 SOMs have been constructed, one for each pulse in the training and validation sets. Each SOM is used to identify the precursor phase of the corresponding disrupted pulse, i.e., to discriminate between ‘safe’ samples and ‘disrupted’
samples containing information about the disruption proximity. Moreover, the SOM is used for data reduction, i.e., only one safe sample for each cluster is selected to build the SVM [5]. The SVM is trained to classify safe and disrupted samples respectively labelled as +1 and -1. During the on-line application, the SVM is fed with all the samples of a pulse, and, for each of them, it returns a label equal to -1 (the sample belongs to the non disruptive phase or to a non disruptive pulse) or +1 (the sample belongs to the disruptive phase and the alarm is triggered).

The answer of the SVM predictor can be validated combining it with the information supplied by the novelty detection. The presence of the ND block influences the predictor behaviour only in case of disruption alarm. In particular, if the predictor triggers the alarm for a sample considered ‘novel’ by the novelty detector, the alarm will be rejected and the sample will be used to update the SVM predictor.

### 7.2.2 Results and comments

The Table I shows the performance of the prediction system in terms of percentage of false alarms (PFA), percentage of missed alarms (PMA), Percentage of Premature Alarms (PPA), and Prediction Success Rate (PSR).

As previously cited, Training and Validation sets only contain disruptive pulses, and hence a wrong prediction can only correspond to a missed alarm or to a premature alarm. On the contrary, the test set contains both disruptive pulses (86) and safe pulses (102).

Table I shows the very good prediction performance for the test set both in terms of PFA and PPA. The performance slightly deteriorates in predicting the alarms in disruptive pulses. In particular, the system is able to correctly predict 63 pulses over the 86 disruptive pulses of the test set, and it does not trigger any alarm for 10 over the 102 non disruptive pulses of the test set.

When the predictor, presented in the previous sections, is integrated with the ND block, the performance are modified as follows:

- 14 of the 23 missed alarms are novel
- 5 of the 9 disruptive pulses triggering a premature alarm are novel
- 18 of the 63 disruptive pulses, correctly predicted by the predictor are labelled as novel by the ND.

Table II shows a comparison of the PMA, PFA, PPA and PSR obtained by the SVM predictor and by the SVM predictor with ND. In the second case the performances are calculated on a reduced test set obtained by the previous test set by discarding the pulses labelled as novel by the novelty detector.
Chapter 7 – Disruption predictors

As can be noted, the number of MAs considerably decreases, the PPA reduces, and the prediction success rate, calculated on the entire test set, goes from 83% to 91%. Unfortunately, as expected, the discrimination capability of the system in the on-line application slightly decreases.

Summarizing, the Predictor with Novelty Detector reaches better performances, if compared to the simple ones, mainly because the novelty detector is able to justify many of the missed alarms of the predictor, as they are recognized to belong to unexplored regions of the operational space.

Tab. I - SVM Predictor performance in terms of Percentage of False Alarms (PFA), and Percentage of Missed Alarms (PMA), Percentage of Premature Alarms (PPA), and Prediction Success Rate (PSR).

<table>
<thead>
<tr>
<th></th>
<th>Training Set</th>
<th>Validation Set</th>
<th>Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMA</td>
<td>0%</td>
<td>24%</td>
<td>27%</td>
</tr>
<tr>
<td>PFA</td>
<td>-</td>
<td>-</td>
<td>0%</td>
</tr>
<tr>
<td>PPA</td>
<td>0%</td>
<td>0%</td>
<td>10%</td>
</tr>
<tr>
<td>PSR</td>
<td>100%</td>
<td>76%</td>
<td>83%</td>
</tr>
</tbody>
</table>

Tab. II - SVM Predictor and Predictor/Novelty Detector performances in terms of Percentage of False Alarms (PFA), and Percentage of Missed Alarms (PMA), Percentage of Premature Alarms (PPA), and Prediction Success Rate (PSR).

<table>
<thead>
<tr>
<th></th>
<th>SVM Predictor Test Set</th>
<th>SVM Predictor/Novelty Detector Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMA</td>
<td>27%</td>
<td>18%</td>
</tr>
<tr>
<td></td>
<td>23/86</td>
<td>9/49</td>
</tr>
<tr>
<td>PFA</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>0/102</td>
<td>0/102</td>
</tr>
<tr>
<td>PPA</td>
<td>10%</td>
<td>8%</td>
</tr>
<tr>
<td></td>
<td>(9/86)</td>
<td>4/49</td>
</tr>
<tr>
<td>PSR</td>
<td>83%</td>
<td>91%</td>
</tr>
<tr>
<td></td>
<td>156/188</td>
<td>138/151</td>
</tr>
</tbody>
</table>
7.3 Prediction with geometrical approach

In this section, the performance of the geometrical approach in terms of prediction and novelty detection is presented.

As shown in the chapt.4, the geometrical approach generates an MLP neural network able to correctly classify patterns coming from the machine, with respect to SVM it does not need to tune the parameters and due to its geometrical characteristics it avoids overfitting. The only parameter to be set is the minimum distance between points and hyperplanes. Yet, as in the SVM case, once the network is trained, it is possible to test it and validate it and apply the algorithm as a novelty detector as well.

Let us suppose to have a 2 dimensions input space, as we can see from Fig.3, the separating process has divided the set into 3 groups, the normal of the separating hyperplanes will be then the axis of the reference system in the feature space (in this case we have 3 separating hyperplane that generate a 3D feature space). More precisely, the axis will be perpendicular to their correspondent separating hyperplane, the coordinate calculated on the normal becomes a coordinate in the feature space after being projected from the activation function. Due to the step activation function, the points in the input space correspond to vertices of a unitary hypercube in the feature space.

The group lying on the positive side of a hyperplane will correspond to a coordinate equal to 1 in the respective axis, 0 otherwise. More precisely, referring to Fig. 3, if we consider the group 1, it lies on the positive side of all hyperplanes then it will be projected onto the vertex (1,1,1) in the feature space, while the group 3 will be projected onto the vertex (0,1,0).

Once all the groups are projected the resulting separating hyperplane can be evaluated.
When using the proposed geometrical approach to develop a disruption prediction, the same 69 disruptive shots, considered for SVM training, have been used. Moreover, for the sake of comparison, the performance of the predictor has been calculated on the same test set considered for the SVM predictor. The validation set has not been used here, because the geometrical approach does not need any cross validation procedure.

In order to construct the training set, a different procedure is followed, with respect to that used for SVM. In particular, the safe and the disrupted samples of each disrupted pulse are labelled as 0 and 1 respectively, following an iterative procedure. That procedure enables to label all the samples using the same geometrical approach that will be used to develop the MLP predictor.

In particular, for each shot, starting from the samples selected by the clustering performed by the SOM (see the previous paragraph), the geometrical approach is used obtaining a first MLP. Note that, only a subset of samples are selected by the SOM. This first MLP is then fed with all the samples of the shot, returning a 0 or a 1. Than, the samples that the network labels as safe are added to the set of safe samples provided that they preceed the first transition of the network from 0 to 1. The procedure is iterated until only one transition occurs.

Usually, the procedure converges in few number of iterations. Such procedure is repeated for each training shot.

At the end of the procedure, all the samples of the training shots are labelled, and they are used by the geometrical procedure to build the MLP predictor.

During the on-line application, the MLP predictor is fed with all the samples of the test pulses, and, for each of them, it returns a label equal to 0 (the sample belongs to the non disruptive phase or to a non disruptive pulse) or 1 (the sample belongs to the disruptive phase and the alarm is triggered).

### 7.3.1 Novelty detector

The aim of novelty detection is reached exploring unmapped region of the hypercube in the feature space. All not mapped vertexes are novel. Yet, can be considered novel all the pattern falling out the hyper-box sourronding all the entire test set in the input space. That hyper-box is built with hyperplanes parallel to the coordinate reference planes.

As said above, during the training phase, the input set is separated into groups that will be projected on the vertexes of an hypercube, following the position they have with respect to the separating hyperplanes.

When a test set is presented to the network the same projection is provided, if a part of the test falls into an unmapped vertex of the hypercube it is considered as novel. The Fig.3 shows unmapped
regions and all free vertexes of the hypercube, these areas can be considered as novel because they belong to unexplored input region.

For what concern the wrapping hyper-box, the Fig.4 shows the details for a 3D case, all the area falling outside this ‘cover’ belong to unexplored regions, hence anything precise can be said about them.

![Figure 20 - Wrapping hyper-box and novel regions, 3D case.](image)

The performance if the predictor in terms of novelty detector is reported in Tab.IV.

### 7.3.2 Results and comments

As can be noted from Table III, the MLP predictor is able to correctly classify all the samples of the training set. Unfortunately, the MLP predictor shows not very good performance in predicting the test set, especially for what concern false alarms, and it is not comparable to what obtained with SVM approach, see Tab.III for details.

When the MLP predictor is used as ND, as described in the previous section, the performance are modified as follows:

- 12 of the 28 missed alarms are novel
- 19 of the 28 disruptive pulses triggering a premature alarm are novel
- 45 of the 89 pulses, correctly predicted by the predictor are labelled as novel by the ND.

Table IV shows a comparison of performances obtained by the MLP predictor and by the MLP ND in terms of number of missed alarms (MA), number of false alarms (FA), and number of premature alarms (PA).
As can be noted, the number of MA, of FA, and of PA considerably decreases. Unfortunatly, as expected, the discrimination capability of the system in the on-line application decreases, as for the SVM.

Tab. III - MLP Predictor performance in terms of Percentage of False Alarms (PFA), Percentage of Missed Alarms (PMA), Percentage of Premature Alarms (PPA), and Prediction Success Rate (PSR).

<table>
<thead>
<tr>
<th></th>
<th>Training Set</th>
<th>Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PMA</strong></td>
<td>0%</td>
<td>32%</td>
</tr>
<tr>
<td></td>
<td>0/69</td>
<td>28/86</td>
</tr>
<tr>
<td><strong>PFA</strong></td>
<td>-</td>
<td>42%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>43/102</td>
</tr>
<tr>
<td><strong>PPA</strong></td>
<td>0%</td>
<td>32%</td>
</tr>
<tr>
<td></td>
<td>0/69</td>
<td>28/86</td>
</tr>
<tr>
<td><strong>PSR</strong></td>
<td>100%</td>
<td>47%</td>
</tr>
<tr>
<td></td>
<td>69/69</td>
<td>89/188</td>
</tr>
</tbody>
</table>

Tab. IV - MLP Predictor and MLP Novelty Detector performances in terms of number of False Alarms (FA), of Missed Alarms (MA), and of Premature Alarms (PA).

<table>
<thead>
<tr>
<th></th>
<th>MLP Predictor Test Set</th>
<th>MLP ND Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MA</strong></td>
<td>28</td>
<td>16</td>
</tr>
<tr>
<td><strong>FA</strong></td>
<td>43</td>
<td>10</td>
</tr>
<tr>
<td><strong>PA</strong></td>
<td>28</td>
<td>7</td>
</tr>
</tbody>
</table>

7.4 Comments

The unavoidable ageing of a neural prediction system is an important problem for experimental machines, like JET, where new states of the plasma are explored. So, it is crucial to have a system able to measure the reliability of the network output and to automatically update the network in the case of plasma configurations not used during the training phase. Both the proposed Novelty
Detection techniques appear promising for enhancing predictor reliability, even if the SVM shows the best performance.
Chapter 7 – Disruption predictors

References


Chapter 8 Conclusions and comments.

The nuclear fusion is become one of the most suitable solution to supply the growing energy demand, as natural resources become not available or not enough to respond the competitors requirements.

Although all the operating nuclear devices are experimental machines, it is then fundamental to involve resources in order to study and build a device able to produce energy for market purposes.

One of the drawbacks of those experimental devices is the occurrence of disruptive events inside the plasma which pose serious limits to both machine's lifetime and its economical feasibility. During a disruption the plasma looses its confinement and the plasma-facing components are then subject to forces of several tons.

In order to avoid or to mitigate the disruptive events a number of neural techniques have been developed, many of those techniques involve neural networks approach, and they seem to be the most suitable to predict the event, or, to build an impending disruption warning indicator.

In this dissertation we gave an overview on disruption prediction with neural techniques focusing on Support Vector Machines and on a new approach based on geometrical synthesis of MLP neural networks.

The techniques, applied on a sub-set of JET database, have demonstrated to be useful to predict disruptions, and more precisely, the SVM are most suitable to predict False Alarms, while the geometrical approach gives the best performance on predicting Missed Alarms.

Yet, they have also demonstrated, when opportunely trained, to be good novelty detectors.