UNIVERSIDADE FEDERAL DE PERNAMBUCO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

SUPPORT VECTOR MACHINES AND PARTICLE SWARM OPTIMIZATION APPLIED TO RELIABILITY PREDICTION

ISIS DIDIER LINS

Orientador: Enrique Andrés López Droguett, PhD

Recife, 2009

UNIVERSIDADE FEDERAL DE PERNAMBUCO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

SUPPORT VECTOR MACHINES AND PARTICLE SWARM OPTIMIZATION APPLIED TO RELIABILITY PREDICTION

A DISSERTATION

BY

ISIS DIDIER LINS

Orientador: Enrique Andrés López Droguett, PhD

Recife, November/2009

UNIVERSIDADE FEDERAL DE PERNAMBUCO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

SUPPORT VECTOR MACHINES AND PARTICLE SWARM OPTIMIZATION APPLIED TO RELIABILITY PREDICTION

A DISSERTATION PRESENTED TO THE UNIVERSIDADE FEDERAL DE PERNAMBUCO IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MESTRE

BY

ISIS DIDIER LINS

Orientador: Enrique Andrés López Droguett, PhD

Recife, November/2009

L759s **Lins, Isis Didier.**

Support vector machines and particle swarm optimization applied to reliability prediction / Isis Didier Lins. – Recife: O Autor, 2009.
xv, 77 folhas, il : figs., tabs.

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia de Produção, 2009.

Inclui referências.

1. Engenharia de Produção. 2. *Support Vector Machines*. 3. Confiabilidade. 4. Otimização. I. Título.

UFPE

658.5 **CDD (22.ed.)** **BCTG/2009-227**

UNIVERSIDADE FEDERAL DE PERNAMBUCO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

PARECER DA COMISSÃO EXAMINADORA

DE DEFESA DE DISSERTAÇÃO DE

MESTRADO ACADÊMICO DE

ISIS DIDIER LINS

"SUPPORT VECTOR MACHINES AND PARTICLE SWARM OPTIMIZATION APPLIED TO RELIABILITY PREDICTION"

ÁREA DE CONCENTRAÇÃO: PESQUISA OPERACIONAL

A comissão examinadora, composta pelos professores abaixo, sob a presidência do(a) primeiro(a), considera a candidata ISIS DIDIER LINS **APROVADA COM DISTINÇÃO.**

Recife, 06 de novembro de 2009.

Prof. ENRIQUE ANDRÉS LÓPEZ DROGUETT, PhD (UFPE)

Prof. LEANDRO CHAVES RÊGO, PhD (UFPE)

Prof. PAULO FERNANDO FERREIRA FRUTUOSO E MELO, Doutor (UFRJ)

ACKNOWLEDGEMENTS

I would like to give my deep and sincere THANK YOU to...

- My parents Bernardete and Sóstenes, for the education they provided me with and for the continuous support in all aspects of my life. My brother Lauro, who lives miles away from us but always gives his constructive opinions and encourages me to learn different computer tools.
- Vicente, who is very special for me, for the kind support and comprehension.
- My big family, which makes me feel happy: vovó Lourdes, vovô Luiz, vovó Myriam, vovô Lauro, Nadja, Niedja, Júnior, tia Adelaide, tia Ló, tia Dulce, tia Eneida, Vanessa, Lula, Pedrinho, Arthur, Fernandinha, Mari, Adriana, Joana and João.
- My friends Sofia, Juju, Ayana, Felipe and Yuri, for the enjoyable moments we passed together.
- Professor Enrique López, for the oportunity of being part of CEERMA, for the attention, dedication and motivational words in the most critical phases of the research.
- Professor Leandro Chaves and Professor Paulo Frutuoso, for the valuable comments, which enhanced the quality of the work.
- Márcio, for the worthy discussions about the dissertation topics and also for the detailed reviews of the document. Paulo, for helping me with statistical concepts. Thiago, Evandson, Rhogi, Ricardo and Romero, for the comments in the early stages of the work.
- Ana and Juliane, who were always available to help me with the bureaucratic issues.
- CNPq, for the financial support.

ABSTRACT

Reliability is a critical metric for organizations since it directly influences their performance in face of the market competition, as well as is essential in maintaining their production systems available. The prediction of such quantitaive metric is then of great interest, as it may anticipate the knowledge about system failures and let organizations avoid and/or overcome such undesirable situations. Systems' reliability depends on the inherent aging factors as well as on the operational conditions the system is subjected to. This may render the reliability modelling very complex and then traditional stochastic processes fail to accurately predict its behavior in time. In this context, learning methods such as Support Vector Machines (SVMs) emerge as alternative to tackle these shortcomings. One of the main advantages of using SVMs is the fact that they do not require previous knowledge about the function or process that maps input variables into output. However, their performances are affected by a set of parameters that appear in the related learning problems. This gives rise to the SVM model selection problem, which consists in choosing the most suitable values for these parameters. In this work, this problem is solved by means of Particle Swarm Optimization, a probabilistic approach based on the behavior of biological organisms that move in groups. Moreover, a PSO+SVM methodology is proposed to handle reliability prediction problems, which is validated by the resolution of examples from literature based on time series data. The obtained results, compared to the ones provided by other prediction tools such as Neural Networks (NNs), indicate that the proposed methodology is able to provide competitive or even more accurate reliability predictions. Also, the proposed PSO+SVM is applied to an example application involving data collected from oil production wells.

Keywords: Support Vector Machines, Particle Swarm Optimization, Reliability Prediction.

RESUMO

Confiabilidade é uma métrica crítica para as organizações, uma vez que ela influencia diretamente seus desempenhos face à concorrência e é essencial para a manutenção da disponibilidade de seus sistemas produtivos. A previsão dessa métrica quantitativa é então de grande interresse, pois ela pode antecipar o conhecimento de falhas do sistema e permitir que as organizações possam evitar ou superar essas situações indesejadas. A confiabilidade de sistemas depende tanto dos efeitos inerentes da idade assim como das condições operacionais a que o sistema é submetido. Isso pode tornar a modelagem da confiabilidade muito complexa de forma que processos estocásticos tradicionais falhem em prever de forma acurada o seu comportamento ao longo do tempo. Nesse contexto, métodos de aprendizado como *Support Vector Machines* surgem como alternativa para superar essa questão. Uma das principais vantagens de se utilizar SVMs é o fato de não ser necessário supor ou conhecer previamente a função ou o processo que mapeia as variáveis de entrada (*input*) em saída (*output*). No entanto, seu desempenho está associado a um conjunto de parâmetros que aparecem no problema de aprendizado. Isso dá origem ao problema de seleção de modelo para SVM, que consiste basicamente em escolher os valores apropriados para esses parâmetros. Nesse trabalho, tal problema é resolvido por meio de Otimização via Nuvens de Partículas (*Particle Swarm Optimization* - PSO), uma abordagem probabilística que é inspirada no comportamento de organismos biológicos que se movem em grupos. Além disso, é proposta uma metodologia PSO+SVM para resolver problemas de previsão de confiabilidade, que é validada por meio da resolução de exemplos da literatura baseados em dados de séries temporais. As soluções encontradas, comparadas às provenientes de outras ferramentas de previsão como Redes Neurais (*Neural Networks* - NNs), indicam que a metodologia proposta é capaz de fornecer previsões de confiabilidade competitivas ou até mesmo mais acuradas. Além disso, a metodologia proposta é utilizada para resolver um exemplo de aplicação envolvendo dados de poços de produção de petróleo.

Palavras-chave: *Support Vector Machines*, Otimização via Nuvens de Partículas, Previsão de Confiabilidade.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ARIMA Autoregressive Integrated Moving Average.

BN Bayesian Network.

CBM Condition Based Maintenance.

DAG Directed Acyclic Graph.

ERM Empirical Risk Minimization.

GA Genetic Algorithm.

GRNN General Regression Neural Network.

GRP Generalized Renewal Process.

HPP Homogeneous Poisson Process.

IIR-LRNN Infinite Impulse Response Locally Recurrent Neural Network.

KKT Karush-Kuhn-Tucker.

MAPE Mean Absolute Percentage Error.

MLP-NN Multilayer Perceptron Neural Network.

MP Mechanical Pumping.

MSE Mean Square Error.

MTBF Mean Time Between Failures.

MTF Miles To Failure.

NFN Neural Fuzzy Network.

NHPP Non-Homogeneous Poisson Process.

NN Neural Network.

NRMSE Normalized Root Mean Square Error.

PCP Progressive Cavities Pumping.

PSO Particle Swarm Optimization.

RBF Radial Basis Function.

RBF-NN Radial Basis Function Neural Network.

RP Renewal Process.

SA Simulated Annealing.

SLT Statistical Learning Theory.

SMDP Semi-Markov Decision Process.

SMO Sequential Minimal Optimization.

SRM Structural Risk Minimization.

SVM Support Vector Machine.

SVR Support Vector Regression.

TBF Time Between Failures.

TTR Time To Repair.

LIST OF SYMBOLS

$\boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*$ ℓ-dimensional vectors of Lagrange multipliers.

γ Parameter involving the RBF kernel parameter σ.

δ Tolerance for consecutive global best fitness values.

ε "Radius" of the "tube" defined in the Vapnik's ε-insensitive loss function.

θ Reliability growth factor.

ϑ Number of points in the validation set.

λ Number of points in the test set.

$\boldsymbol{\xi}, \boldsymbol{\xi}^*$ ℓ-dimensional vectors of slack variabies.

ρ Angle between $\mathbf{x}_{SV,-1}$ and $\mathbf{w}$.

σ RBF kernel parameter

Φ Mapping $\mathbb{R}^n \rightarrow \mathcal{F}$.

φ Sum of PSO parameters c_1 and c_2.

χ Constriction factor

ω Angle between $\mathbf{x}_{SV,+1}$ and $\mathbf{w}$.

b Linear coefficient of H.

$bIter$ PSO iteration in which the best particle was found.

C Trade-off between training error and machine capacity.

c_1, c_2 PSO parameters.

D Training data set.

d Degree of a polynomial

$d(\mathbf{x})$ Decision function.

$\mathcal{F}$ Feature space.

$f(\mathbf{x})$ Mapping or function of $\mathbf{x}$.

f_i Validation fitness value associated with the i^{th} particle.

f_{test} Test fitness value associated with the best particle.

$g(\mathbf{x})$ Non-zero function that satisfies $\int g^2(\mathbf{x})d\mathbf{x} < \infty$.

H Separating hyperplane.

h Test point index.

H_-, H_+ Lower and upper hyperplanes that define the margin.

i Training point index; particle index.

j Auxiliar index; dimension index of a vector.

K Kernel function.

k Category index of a multi-classification problem; number of steps ahead minus one in a time series prediction; number of subsets in cross-validation.

$\mathcal{L}$ Lagrangian function.

$\mathcal{L}_d$ Dual Lagrangian function.

L Loss function.

ℓ Number of training points $(\mathbf{x}_i, y_i)$.

M Margin.

m Number of categories of a multi-classification problem.

n Dimension of $\mathbf{x}$; number of variables.

$nFSV$ Number of free support vectors.

$nIter$ Maximum number of PSO iterations.

$nLFSV, nUFSV$ Number of free support vectors lying on H_- and on H_+ (regression case).

$nNeigh$ Number of particles' neihgbors.

$nPart$ Number of particles.

nSV Number of support vectors.

p Number of lagged variables in a time series, dimension of $\mathbf{x}_t$.

$\mathbf{p}$ The best individual position a particle has found so far.

$\mathbf{p}_g$ Best position in the neighborhood of a particle.

$\mathbb{R}$ Set of real numbers.

r Number of categories associated with a categorical variable.

s Support vector index.

T Total operational time of an equipment or matrix transpose operation.

t Time index.

u_1, u_2 Random numbers generated by a uniform distribution defined in the interval [0,1].

v_{max} Maximum velocity.

$\mathbf{v}$ Velocity vector.

$\mathbf{v}^{max}$ Maximum velocity a particle can have associated with the j^{th} dimension.

w Inertia weight.

$\mathbf{w}$ Normal vector in relation to H.

x^{ind} Indicator variable.

$x_{min,j}, x_{max,j}$ Minimum and maximum training values of the j^{th} dimension of $\mathbf{x}_i$.

$\mathbf{x}$ Multidimensional input vector; current particle position

$\mathbf{x}^{min}, \mathbf{x}^{max}$ n-dimensional vectors of variables' bounds.

$\mathbf{x}_{SV,-1}, \mathbf{x}_{SV,+1}$ Support vector from the negative and positive classes.

y Output value.

$\hat{y}$ Predicted output value.

y_{min}, y_{max} Minimum and maximum training values of the output y.

1 INTRODUCTION

1.1 Opening Remarks

Reliability can be understood as the probability of a system to properly perform the tasks it was designed for, under certain conditions, during a predefined time length (RAUSAND & HOYLAND, 2004). These systems can be either a specific component or an entire system. With a slightly different interpretation, the absence of reliability means frequent system breakdowns and thus loss of productivity and increased costs, which may be associated with maintenance actions, legal penalties and also with the (bad) image of the organization in face of their costumers. It is also a critical issue in systems that entails environmental and human risks, such as oil refineries and nuclear power plants, given that their failures may incur in catastrophic events. Therefore, reliability is a key factor to production systems, since it is directly related to the competitive performance of organizations in the market share they are inserted.

The systems' reliability varies during their lifetime since it is influenced by the environmental and load conditions under which they operate. In this way, it is of great interest to identify this quantitative indicator of systems' performance in order to control it by means of appropriate maintenance actions, so as to guarantee the desired level of production and safety. According to Zio *et al.* (2008), reliability prediction modeling of an item may be conducted during various phases of its life-cycle, including the concept validation and definition, the design, operation and maintenance phases. At any stage, the reliability predictions obtained serve the purpose of anticipating the evolution of the reliability of the component so as to allow for taking the proper actions for its maintaining and, possibly, improvement. For systems design considering reliability and cost metrics, see for example Lins & Droguett (2008) and Lins & Droguett (2009).

The evaluation of the reliability behavior in time has been accomplished by means of stochastic methods, which usually involves simplifying assumptions so as to allow for the analytical treatment. The usual stochastic processes to model the reliability evolution are the Renewal Process (RP) and Non-Homogeneous Poisson Process (NHPP). If RP is chosen, the times between failures are independent and identically distributed with an arbitrary probability distribution. Besides, one assumes that the component, after a failure, is subjected to a perfect repair and returns into operation with a condition it presented when new ("as good as new"). A special case of the RP is the Homogeneous Poisson Process (HPP), in which the times between failures are modeled by identical and independent Exponential distributions and has the underlying supposition that the probability of occurrence of a failure in any time interval depends only on the length of that interval. This assumption may be true for some electronic components or in a short period of time (ROSS, 2000). On the other hand, using NHPP, the times between failures are neither independent nor identically distributed. In addition, it is supposed

that the maintenance crew makes a minimal repair in the failed component, that is, it is returned to an operational state with the same condition it had just before the failure occurrence ("as bad as old").

However, the hypothesis of minimal or perfect repairs required to utilize either NHPP or RP, respectively, are often not realistic. In practical situations, corrective maintenance actions are likely to be imperfect repairs, *i.e.*, they are intermediate actions between minimal and perfect repairs and the equipment returns into operation with a condition better than old and worse than new. In this context, Generalized Renewal Process (GRP) can be used to model failure-repair processes of components subject to imperfect repairs. In GRP, a parameter q (rejuvenation parameter) is introduced in the model and the value it assumes is related to the maintenance action effectiveness. However, this value is often considered as a constant for all interventions without taking into consideration the current state of the system. For further details in RP, HPP, NHPP and GRP, the interested reader may consult Rigdon & Basu (2000) and Rausand & Hoyland (2004).

In reality, the reliability of a system is affected by a set of time-dependent, external (operational and environmental) variables which are dependent among them. As an outcome, reliability prediction may demand sophisticated probabilistic models so as to realistically capture the complexities of the systems and components reliability behavior, which may result in burdensome mathematical formulations that in the end, may not provide the required accuracy of the reliability estimates (MOURA & DROGUETT, 2009).

In this context, learning methodologies based on data emerge and Support Vector Machine (SVM) is the one selected to be studied and applied to reliability problems in this dissertation. SVM has been developed since the years 1960's and was first introduced by Vapnik and Chervonenkis (see Vapnik (2000)). Loosely speaking, SVM is a learning method whose theory is based on statistical concepts. It incorporates the idea of learning about the phenomenon under analysis from real observations about it. The main idea is to train a "machine" with real pairs of inputs and outputs so as to allow for the prediction of future outputs based on observed inputs. The training algorithm is a quadratic optimization problem, where the objective function essentially entails a generalization error, which comprises the training error as well as the error related to the machine ability in handling unseen data. The learning problem can be either of classification, in which the outputs are discrete values representing categories, or of regression, when the outputs can assume any real value.

Competitive models to SVM are the artificial Neural Network (NN) (HAYKIN, 1999) and Bayesian Network (BN) (KORB & NICHOLSON, 2003). It is interesting to notice that, in accordance with Kecman (2005), SVM has been developed in the reverse order to the development of NN. SVM evolved from the theory to the implementation and experiments, while NN followed a more "heuristic" path, from applications to theory. The strong theoretical background of SVM did not make it widely appreciated at first. It was believed that, despite its theoretical foundations, SVM was neither suitable nor relevant for practical purposes. How-

ever, afterwards, the use of SVM in learning benchmark problems involving for example digit recognition provided excellent results and then such tool was finally taken seriously.

Haykin (1999) asserts that an NN is designed to model the way in which the brain performs a particular task or function of interest and is usually simulated in software on a digital computer. To achieve good performance, NN employs a massive interconnection of simple computing cells referred to as "neurons" or "processing units", which are basically formed by (i) a set of connecting links, each one of them characterized by a weight; (ii) an adder for summing the input signals, weighted by the respective connecting links; (iii) an activation function for limiting the amplitude of its output.

NN involves the Empirical Risk Minimization (ERM), which measures only the errors from the training step and is appropriate when there is a large quantity of training examples (VAPNIK, 2000). On the other hand, the training phase of SVM entails a convex quadratic optimization problem, whose objective function embodies the principle of Structural Risk Minimization (SRM). The general idea of the ERM is to minimize the error during the training phase, while the SRM aims at minimizing the upper bound on the generalization error. Additionally, the characteristics of the SVM training optimization problem enable the Karush-Kuhn-Tucker (KKT) conditions to be necessary and sufficient to provide a global optimum, differently from NN that may be trapped on local minima (SCHÖLKOPF & SMOLA, 2002).

According to Korb & Nicholson (2003), BNs are graphical models for reasoning under uncertainty, represented by acyclic graphs (BONDY & MURTY, 2008) whose nodes denote variables and arcs represent causal connections between the related variables. Also, a BN can model the quantitative strength of the connections between variables, allowing probabilistic beliefs about them to be updated automatically as information becomes available.

For example, Ramesh *et al.* (2003) use a hybridism of BN and SVM in order to predict the axis positioning errors in machine tools. Such errors depend on the machine temperature profile and also on the specific operating condition the machine is subjected to. Firstly, a BN approach is used to classify the error into categories associated with the different operating conditions. After that, a knowledge base of errors due to specific machine conditions is formed and combined with classification results as input to an SVM regression model for mapping the temperature profiles with the measured errors. The authors provide the following reasons for using a BN: paucity of data, expert knowledge can be incorporated when data availability is sparse and it permits the learning of causal relationships between variables. For the use of SVM, the authors note that it does not require previous knowledge of the relationship between input and output variables.

In the context of reliability prediction from time series data, in addition to the previous methods, there is also the Autoregressive Integrated Moving Average (ARIMA) and the Duane models as alternatives for SVM. Morettin & Toloi (2004) state that the ARIMA model has been one of the most popular approaches in time series forecasting and assume that predicted values are a linear combination of the previous values and errors. The Duane model, in turn,

are frequently applied on the analysis of reliability during the early stages of product design and development, which may involve reliability growth modeling. It assumes an empirical relationship whereby the improvement in Mean Time Between Failures (MTBF) is proportional to T^{θ}, where T is the equipment's total operational time and θ is the reliability growth factor (LEWIS, 1987; SMITH, 2001).

Therefore, the advantages of SVM in relation to the other methods are: (i) no requirements of previous knowledge of or suppositions about the relation between inputs and outputs; (ii) no need of a large quantity of data; (iii) incorporation of the SRM principle which provides it with a better generalization ability and (iv) the resolution of a convex optimization problem in the training stage.

Although SVM has been introduced in the sixties, its first applications in prediction based on data series occurred in the end of the years 1990's, for example see Müller *et al.* (1999). Indeed, Sapankevych & Sankar (2009) provide a literature survey of the works using SVM in time series predictions. They make a count on the number of works regarding the knowledge areas in which they are inserted (Table 1.1). It can be noticed that only three works were classified into the reliability prediction context, making it as the last field in number of related works. From this fact, it can be inferred that SVM is actually in the early stages of its applicability in the reliability prediction problems based on time series data, if compared for example with the economic field.

Table 1.1: Number of SVM time series prediction papers by application. Adapted from Sapankevych & Sankar (2009), p. 26

Application	**Number of papers**
Financial market prediction	21
Electric utility forecasting	17
Control systems and signal processing	8
Miscellaneous applications	8
General business applications	5
Environmental parameter estimation	4
Machine reliability forecasting	3

Nevertheless, the performance of SVM is influenced by a set of parameters that appear in the training problem. This fact gives rise to the model selection problem that consists in choosing suitable values for these parameters. They are actually very difficult to be manually tuned and systematic procedures may be required to perform this task. In this way, methods such as Particle Swarm Optimization (PSO) (BRATTON & KENNEDY, 2007) can be used to tackle the model selection problem. PSO is an optimization probabilistic heuristic, well-suited to deal with real variables, based on the behavior of biological organisms that move in groups such as birds. The nature concepts of cognition and socialization are translated in mathematical formulations for updating particles velocities and positions throughout the search space towards an

optimum position. There are basically two models of communication networks among particles: in one of them, the most simple, all particles are connected to each other (*gbest*) and in the other, they are able to communicate only with some of them (*lbest*).

Besides PSO, there are other probabilistic approaches to solve the model selection problem such as Genetic Algorithm (GA) (GOLDBERG, 1989). GA is a computational method usually used for optimization tasks and attempts to mimic the natural evolution process. It is based on several genetic operators such as crossover and mutation and is often computationally more expensive if compared to PSO. Other possible alternative to tackle the SVM model selection problem is the grid search method (MOMMA & BENNETT, 2002). The latter assumes the parameters as discrete values within a range and all possible combinations of them are assessed. Its main drawbacks consist in the discretization of the search space as well as in the great number of possibilities to be evaluated when there are several parameters to adjust.

In the following Section, some previous works related to SVM as well as to SVM model selection problem are presented.

1.2 Previous works

For classification tasks, Rocco & Moreno (2002) have used SVM to classify a component as operational or faulty in order to evaluate system overall reliability. The authors take advantage of the SVM velocity, which is greater than the one from the traditional discrete event simulation approach of Monte Carlo (BANKS *et al.*, 2001). Then, they couple Monte Carlo simulation with SVM. Rocco & Zio (2007), in turn, use a multi-classification SVM to categorize anomalies in components. Widodo & Yang (2007) make a review of the SVM applied to condition monitoring and fault diagnosis.

For the prediction of reliability related measures based on time series, Hong & Pai (2006) use SVM coupled with an iterative method for selecting the associated SVM parameters. They forecast time failures of an engine. Additionally, Pai (2006) and Chen (2007) propose a GA+ SVM approach to predict reliability values of an engine. GA is used as optimization tool to obtain the most suitable parameter values. No work was found relating the use of SVM with system characteristics, like temperature or the number of installed components, to predict continuous reliability metrics (*e.g.* Time Between Failures (TBF), Time To Repair (TTR)).

The methodology of PSO+SVM is presented in some works. Lin *et al.* (2008) use such approach to the model selection problem but also to the choice of the most relevant input entries (problem known as feature selection). They apply the methodology on freely available general data sets in Internet repositories. Dissolved gases contents on power transformers' oil are predicted in the work of Fei *et al.* (2009) with PSO+SVM approach. Also in the electricity context, Hong (2009) predicts the electric load by means of a PSO combined with a regression SVM. Samanta & Nataraj (2009) apply PSO with a classification SVM in the context of fault detection.

In this way, so far, no PSO+SVM approach has been proposed to tackle reliability prediction tasks based either on time series data or on specific metrics of the system under consideration. Additionally, all of them involved the most simple communication networks among particles, the *gbest* model.

1.3 Justification

As already mentioned, reliability of components and systems is a key element of production systems due to its direct relation with productivity and costs, and thus with the performance of organizations within market. Thus, reliability prediction is a subject of great interest that can be reverted in economic and competitive gains to organizations.

Besides this first motivation and reason, as can be concluded by the survey of Sapankevych & Sankar (2009), there are few works of reliability prediction based on time series using SVM as forecast tool. In addition, from the literature review, no work was found either concerning SVM regression to predict reliability metrics based on system features or relating PSO with SVM to tackle the model selection problem in this specific context. Also, all works that involved PSO+SVM used a *gbest* communication network among particles.

Also, the Condition Based Maintenance (CBM), which is a maintenance program that recommends maintenance actions based on information collected via equipment condition monitoring, may incorporate SVM. CBM consists in three main steps (JARDINE *et al.*, 2006):

- Data acquisition: data regarding equipment status is collected.
- Data processing: the acquired data is processed, analyzed and interpreted.
- Maintenance decision-making: after data interpretation, efficient maintenance policies are recommended.

SVM may take place in the second step of CBM. If the CBM approach is well established and implemented, it can provide the decrease of maintenance costs. For example, the work of Moura *et al.* (2009) assumes that the considered system is continuously monitored and that its current state is available. Then, the authors propose maintenance policies based simultaneously on availability and cost metrics of systems by means of Semi-Markov Decision Process (SMDP) and GA. Hence, they tackle the last step of a CBM program and the inclusion of SVM as a previous step would result in a more comprehensive approach of the considered problem.

1.4 Objectives

1.4.1 Main Objective

This dissertation proposes a PSO algorithm to solve the SVM model selection problem. The resulted PSO+SVM methodology is then applied to the reliability context specifically involving

regression problems such as reliability prediction problems based on time series data and/or on system specific features.

1.4.2 Specific Objectives

In order to attain the main objective, the following specific goals are established:

- Literature review of the most used methods to solve the SVM model selection problem.
- Implementation of a PSO algorithm linked with a SVM library to tackle the SVM model selection problem, resulting in a PSO+SVM approach.
- Application of the proposed PSO+SVM to reliability prediction problems based on time series data and also based on data collected from a real system.
- Performance comparison between the proposed PSO+SVM and other methodologies such as GA+SVM, as well as other time series methods such as NN and ARIMA.
- Performance comparison between *gbest* and *lbest* models for the application examples taken into account.

1.5 Dissertation Layout

Besides this introductory chapter, this dissertation comprises four more chapters, whose contents are described as follows:

- **Chapter 2**: the theoretical background is presented. Initially, the SVM methods for classification and regression tasks are detailed. Then, the related model selection problem as well as a survey of the methods that have been applied to tackle it are described. This chapter also contains the general ideas underlying PSO algorithms.
- **Chapter 3:** the PSO methodology proposed in this work to tackle the SVM model selection problem is detailed. Also, the PSO+SVM combination is commented.
- **Chapter 4:** three application examples from literature are presented and resolved by means of the proposed PSO+SVM methodology. One of them is used in two different ways, which yields four examples. The outcomes are compared to results from other tools available in literature (NN, ARIMA, among others). Also, an application example involving data collected from oil production wells is solved. Then, a comparison between the *gbest* and *lbest* PSO and a discussion about the obtained results take place.
- **Chapter 5:** a summary of the main contributions of this dissertation is provided along with some comments about its limitations. In addition, this chapter presents some topics associated with the ongoing research and suggestions for future works.

2 THEORETICAL BACKGROUND

2.1 Support Vector Machines

SVM is a learning method widely used in pattern recognition and regression problems. The applications of SVM belong to different domains, from computational biology (BEN-HUR *et al.*, 2008) to financial series forecasting (GESTEL *et al.*, 2001). In the reliability context, SVM has been used for example in CBM and fault detection (WIDODO & YANG, 2007), to classify anomalies in components (ROCCO & ZIO, 2007), to forecast equipments' reliability (HONG & PAI, 2006; CHEN, 2007), among others.

In its classical formulation, SVM is a supervised learning method, since it is based on (input, output) examples. SVM stems from the Statistical Learning Theory (SLT) and it is particularly useful when the process in which inputs are mapped into outputs is not known. According to Kecman (2005), the learning problem is as follows: there is an unknown nonlinear dependence (mapping, function) $y = f(\mathbf{x})$ between a multidimensional input vector $\mathbf{x}$ and an output y. The only information available is a data set $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, where ℓ is the number of examples in D. The data set D is called training set due to its purposeful use in training the learning machine.

Depending on the type of output y, different learning problems are defined:

- Classification problems: y assumes discrete values that represent categories. If only two categories are considered (*e.g.* $y = -1$ or $y = +1$), the problem at hand is of binary classification. Otherwise, if three or more categories are taken into account, it is the case of a multi-classification problem.

- Regression problems: y is real-valued and its relation with the input vector $\mathbf{x}$ is given by a function.

The solution of the learning problem is a decision function or a regression (target) function when, respectively, a classification or a regression is considered. For some learning machines like NN, the underlying idea to obtain the answer of the learning problem is the principle of ERM, which measures only the errors from the training step and is suitable for situations where there is a large quantity of training examples (VAPNIK, 2000). On the other hand, obtaining the learning problem solution via SVM involves a convex quadratic optimization problem, whose objective function embodies the principle of SRM. This principle entails the minimization of the upper bound of the generalization error, which is formed by two parts: one of them is associated with the machine ability to classify or predict unseen data (*i.e.*, examples that are not in D), and the other regards the training errors. In this way, there is a trade-off between model's capacity and training accuracy. Machines with low capacity have high training and generalization errors, which characterizes the situation of *underfitting* the data. On the other hand, increasing too

much the machine capacity yields small training error but the generalization error grows due to its bad performance in classifying or predicting unseen data. This latter is the case of *overfitting* the data, that is, the machine is too specialized in the training set that it is unable to work well on data not in D. The behavior of those errors in relation to the machine capacity is illustrated in Figure 2.1 and the examples of underfitting and overfitting in classifying linearly separable data are depicted in Figure 2.2. According to Kecman (2005), the SRM principle was proved

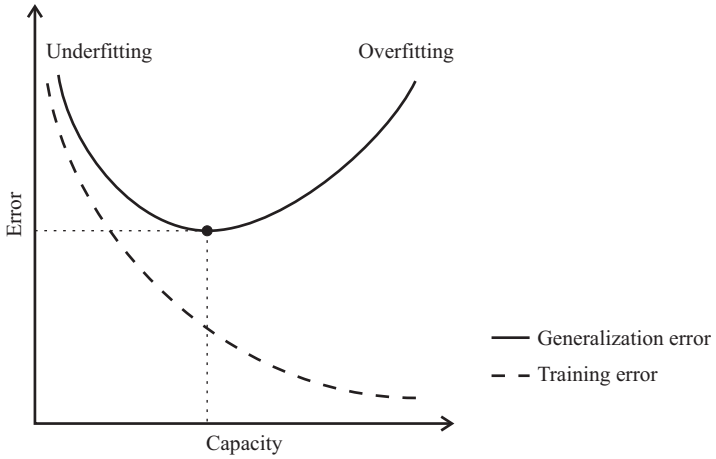

Figure 2.1: Relation between model capacity and error

to be useful when dealing with small samples and its main idea consists in finding a model with adequate capacity to describe the given training data set. For further details in ERM and SRM, see Vapnik (2000), Kecman (2001) and Kecman (2005).

An important advantage of SVM is that the training phase is accomplished by the resolution of a convex quadratic optimization problem with a unique local optimum that is also global (BOYD & VANDENBERGHE, 2004). Such problem involves well-known and established optimization theory and techniques to solve it, for example the concept of Lagrangian multipliers and KKT conditions. Besides that, the desired decision or regression function is always linear and presented by a hyperplane. Even if the relation in the input space is not linear, SVM uses kernel functions (see Schölkopf & Smola (2002)) to map the input data $\mathbf{x}$ into a feature space, often of higher dimension, in which such relation is linear. In this feature space the training procedure is executed. The following subsections introduce the main SVM classifiers as well as regression via SVM.

2.1.1 Linear Maximal Margin Classifier for Linearly Separable Data

The binary classification of linear separable data is the most simple learning problem. Due to its simplicity, it is often not applicable in practical situations. Despite that, it presents the fundamental aspects of SVM and it is useful to understand the more complex and realistic SVM approaches.

Let $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)\}$ be the training set with $\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$ and

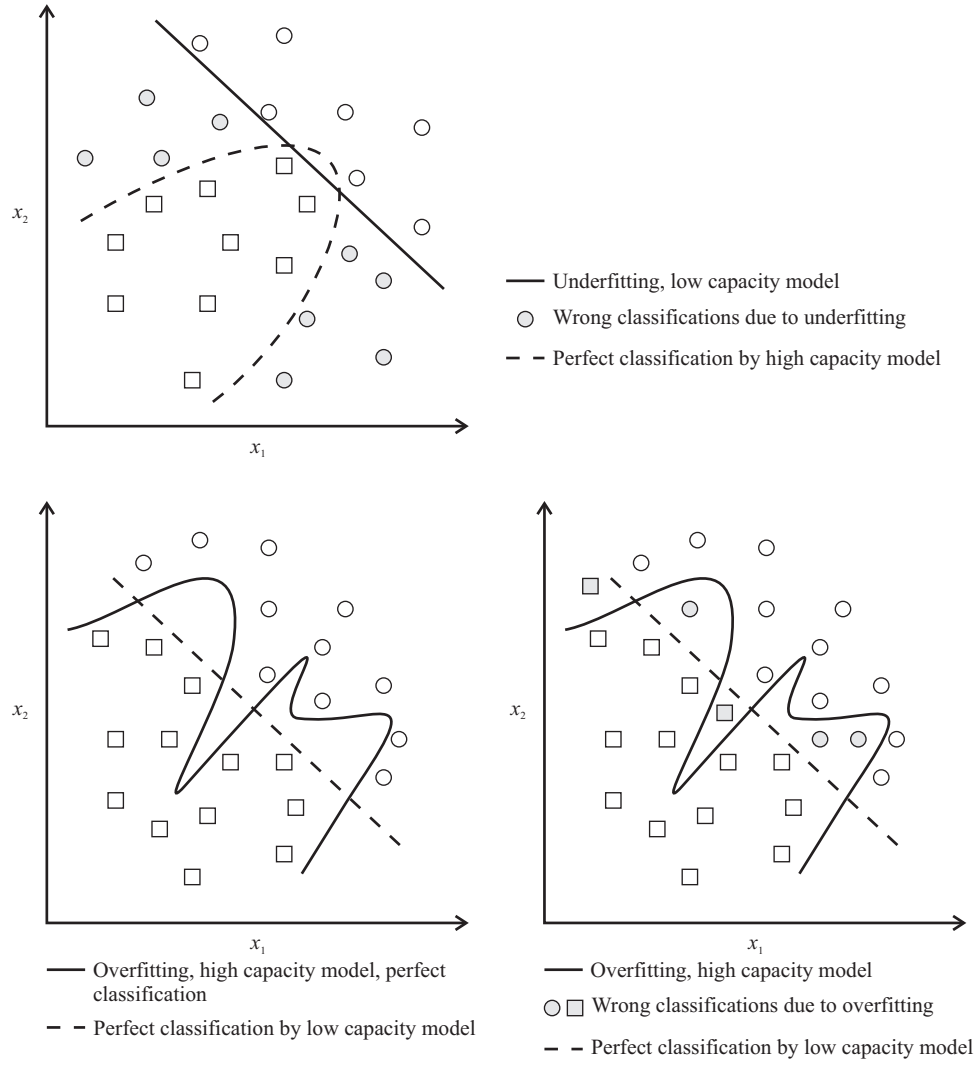

Figure 2.2: Underfitting (up) and overfitting (down) the data in the case of binary classification of linearly separable data. Adapted from Kecman (2005), p. 8

$i = 1, 2, \ldots, \ell$ denoting the i^{th} training example. Suppose that the data is linearly separable, that is, they can be perfectly separated by hyperplanes. The best of those hyperplanes is the one with maximal margin, *i.e.*, the one which maximizes the minimum distance between examples from distinct classes. If this optimal hyperplane is defined, one gets the decision function. The hyperplane equation in matrix form is given by:

$$H = \mathbf{w}^T \mathbf{x} + b = 0 \tag{2.1}$$

where $\mathbf{w}$ is the vector normal to the hyperplane, T indicates the matrix transpose operation, $\mathbf{x}$ is the input vector and b is the linear coefficient of the hyperplane.

Since the data is separable, in order to correctly classify a given example $(\mathbf{x}_i, y_i)$, the decision function should satisfy the constraints:

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1, \qquad \text{if } y_i = 1 \tag{2.2}$$

$$\mathbf{w}^T\mathbf{x}_i + b \leq -1, \qquad \text{if } y_i = -1 \tag{2.3}$$

that can be expressed by a single inequality:

$$y_i \cdot (\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \tag{2.4}$$

When (2.2) and (2.3) are active, *i.e.*, equalities hold, two hyperplanes H_+ and H_- are respectively defined and the distance between them is the margin (M). The vectors $\mathbf{x}$ from the training set which satisfy either H_+ or H_- are the so-called support vectors. As an illustration, Figure 2.3 depicts a simple two dimensional case of binary classification in which the margin, hyperplanes and support vectors are indicated.

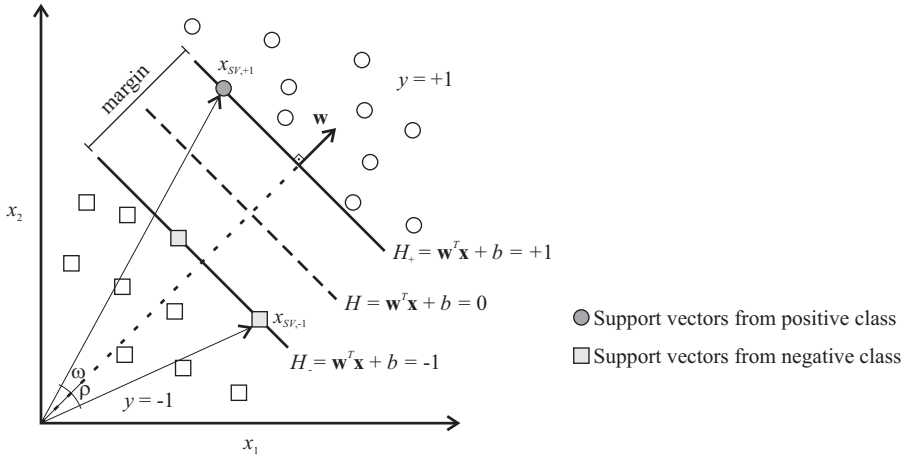

Figure 2.3: Binary classification. Adapted from Kecman (2001), p. 154

The margin is, indeed, defined by the distance between support vectors from distinct classes (*e.g.* $\mathbf{x}_{SV,-1}$ and $\mathbf{x}_{SV,+1}$) projected onto the same direction of the hyperplanes' perpendicular vector $\mathbf{w}$. That is,

$$M = (\mathbf{x}_{SV,+1} - \mathbf{x}_{SV,-1})_{\mathbf{w}} = ||\mathbf{x}_{SV,+1}||\cos(\omega) - ||\mathbf{x}_{SV,-1}||\cos(\rho) \tag{2.5}$$

in which ω and ρ are respectively the angles between $\mathbf{x}_{SV,+1}$ and $\mathbf{w}$ and between $\mathbf{x}_{SV,-1}$ and $\mathbf{w}$ (see Figure 2.3). These angles are given by:

$$\cos(\omega) = \frac{\mathbf{w}^T\mathbf{x}_{SV,+1}}{||\mathbf{w}|| \cdot ||\mathbf{x}_{SV,+1}||} \qquad \cos(\rho) = \frac{\mathbf{w}^T\mathbf{x}_{SV,-1}}{||\mathbf{w}|| \cdot ||\mathbf{x}_{SV,-1}||} \tag{2.6}$$

By replacing (2.6) in (2.5) and considering that $\mathbf{x}_{SV,+1}$ and $\mathbf{x}_{SV,-1}$ belong to H_+ and H_-, respectively, it is obtained:

$$M = \frac{\mathbf{w}^T\mathbf{x}_{SV,+1} - \mathbf{w}^T\mathbf{x}_{SV,-1}}{||\mathbf{w}||} = \frac{-b+1-(-b-1)}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||} \tag{2.7}$$

where $||\mathbf{w}|| = \sqrt{\mathbf{w}^T\mathbf{w}} = \sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}$ is the ℓ_2-norm of vector $\mathbf{w}$. For further details in margin definition consult Kecman (2001). Also, one can find information about the underlying analytic geometry concepts in Reis & Silva (1996).

It is noticeable that minimizing $||\mathbf{w}||$ is equivalent to maximize M. Moreover, to minimize $\sqrt{\mathbf{w}^T\mathbf{w}}$ is similar to minimize $\mathbf{w}^T\mathbf{w}$. In this way, the convex optimization problem to be resolved during the training step is:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & y_i \cdot (\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, \ell \end{aligned} \tag{2.8}$$

where the constant $\frac{1}{2}$ is a numerical convenience and does not change the solution. Notice that in the case of linearly separable data, there is no training error and $\mathbf{w}^T\mathbf{w}$ is associated with machine capacity (VAPNIK, 2000; KECMAN, 2005). To solve (2.8), a classic quadratic programming problem, Lagrange multipliers along with KKT conditions are used. Due to its convex nature (objective function is convex and its constraints result in a convex feasible region), the KKT conditions are necessary and sufficient for an optimum (BOYD & VANDENBERGHE, 2004). Firstly, the Lagrangian function is structured as follows:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{\ell} \alpha_i \cdot [y_i \cdot (\mathbf{w}^T\mathbf{x}_i + b) - 1] \tag{2.9}$$

in which $\boldsymbol{\alpha}$ is the ℓ-dimensional vector of Lagrange multipliers. It is necessary to find the saddle point $(\mathbf{w}_0, b_0, \boldsymbol{\alpha}_0)$ of $\mathcal{L}$, since (2.9) has to be minimized with respect to the *primal variables* $\mathbf{w}$ and b and minimized with respect to the *dual variables* $\alpha_1, \alpha_2, \ldots, \alpha_\ell$, which should be non-negative, *i.e.*, $\alpha_i \geq 0$ for all i. This problem can be resolved either in the primal or in the dual space. However, the latter approach has been adopted by a number of works due to the insightful results it provides (for example, see Vapnik (2000), Schölkopf & Smola (2002), Kecman (2005)).

In order to find the saddle point of $\mathcal{L}$, the KKT conditions are then stated:

$$\frac{\partial \mathcal{L}(\mathbf{w}_0, b_0, \boldsymbol{\alpha}_0)}{\partial \mathbf{w}} = 0, \quad \mathbf{w}_0 = \sum_{i=1}^{\ell} \alpha_{0i} y_i \mathbf{x}_i \tag{2.10}$$

$$\frac{\partial \mathcal{L}(\mathbf{w}_0, b_0, \boldsymbol{\alpha}_0)}{\partial b} = 0, \quad \sum_{i=1}^{\ell} \alpha_{0i} y_i = 0 \tag{2.11}$$

$$y_i \cdot (\mathbf{w}_0^T\mathbf{x}_i + b_0) - 1 \geq 0, \quad i = 1, 2, \ldots, \ell \tag{2.12}$$

$$\alpha_{0i} \geq 0, \quad \forall\, i \tag{2.13}$$

$$\alpha_{0i} \cdot [y_i \cdot (\mathbf{w}_0^T\mathbf{x}_i + b_0) - 1] = 0, \quad \forall\, i \tag{2.14}$$

where the first two equalities in (2.10) and (2.11) result directly by the derivatives of $\mathcal{L}$ with

respect to the primal variables being null when evaluated at the optimum $(\mathbf{w}_0, b_0, \boldsymbol{\alpha}_0)$; the inequalities in (2.12) and (2.13) require primal and dual feasibility of the solution, respectively; the last equations (2.14) are the KKT complementarity conditions which states that the product of dual variables and primal constraints must vanish at the optimum. A useful insight stems from equations (2.14): support vectors lie on either H_+ or H_- and then nullify the second term of the KKT complementarity conditions, which along with condition (2.13) implies non-negative Lagrange multipliers. According to Nocedal & Wright (2006), the constraints can be classified as:

- Inactive: if the constraint strictly satisfies the inequality and the related Lagrange multiplier is exactly 0. Then, the optimal solution as well as the optimal objective function value are indifferent to whether such constraint is present or not.
- Strongly active: if the constraint satistfies the equality and the respective Lagrange multiplier is srictly positive. Thus, a perturbation in the constraint has an impact on the optimal objective value with magnitude proportional to the Lagrange multiplier.
- Weakly active: if the constraint satisfies the equality and the associated Lagrange multiplier is exactly 0. In these cases, small perturbations in such constraint in some directions hardly affects the optimal objective value and solution.

In this way, for practical purposes, weakly active constraints can be treated as inactive. Hence, support vectors are identifiable by strictly positive Lagrange multipliers. It is important to emphasize that solving the SVM training problem is equivalent to solve the system defined by the KKT conditions, given they are necessary and sufficient for a convex optimization problem. By substituting the equalities (2.10) and (2.11) into $\mathcal{L}$, it is obtained an expression involving only the dual variables which has to be maximized:

$$\max_{\boldsymbol{\alpha}} \quad \mathcal{L}_d(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \tag{2.15}$$

$$\text{subject to} \quad \alpha_i \geq 0, \quad i = 1, 2, \ldots, \ell \tag{2.16}$$

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0 \tag{2.17}$$

The dual problem resolution yields ℓ non-negative values for the Lagrange multipliers. By replacing these values in equation (2.10), the optimal normal vector $\mathbf{w}_0$ is directly defined. Still, notice that the summation over all ℓ is exactly the same as over only the support vectors, since $\alpha_i = 0$ if i is not a support vector. Differently from $\mathbf{w}_0$, b_0 is implicitly determined by KKT complementarity conditions for any chosen support vector s ($s = 1, 2, \ldots, nSV$, where nSV is the number of support vectors). However, due to numerical instabilities, it is better to set b_0 as the mean over all values resulted from nSV calculations of (2.14), that is (BURGES, 1998;

KECMAN, 2005):

$$b_0 = \frac{1}{nSV}\sum_{s=1}^{nSV}\left(\frac{1}{y_s} - \mathbf{w}_0^T\mathbf{x}_s\right) \tag{2.18}$$

Once the optimal solution has been found, the decision function is obtained from the separating hyperplane H:

$$d(\mathbf{x}) = \mathbf{w}_0^T\mathbf{x} + b_0 = \sum_{i=1}^{\ell}\alpha_{0i}y_i\mathbf{x}_i^T\mathbf{x} + b_0 \tag{2.19}$$

If $d(\mathbf{x}) < 0$, then $\mathbf{x}$ is categorized into the negative class $(y = -1)$. Otherwise, if $d(\mathbf{x}) > 0$, $\mathbf{x}$ is classified as being in the positive class $(y = +1)$.

2.1.2 Linear Soft Margin Classifier for Non-Linearly Separable Data

In some situations, the linear classifier still is a good option to separate overlapping data. The training optimization problem can then be adapted to support the training step for linearly non-separable data. Differently from the previous classifier, it is now necessary to allow for training errors, since constraints (2.2) and (2.3) can be violated. Because of this, the margin becomes soft and examples within or beyond it (either in the correct side of the separating hyperplane or in the wrong one) are permitted, see Figure 2.4.

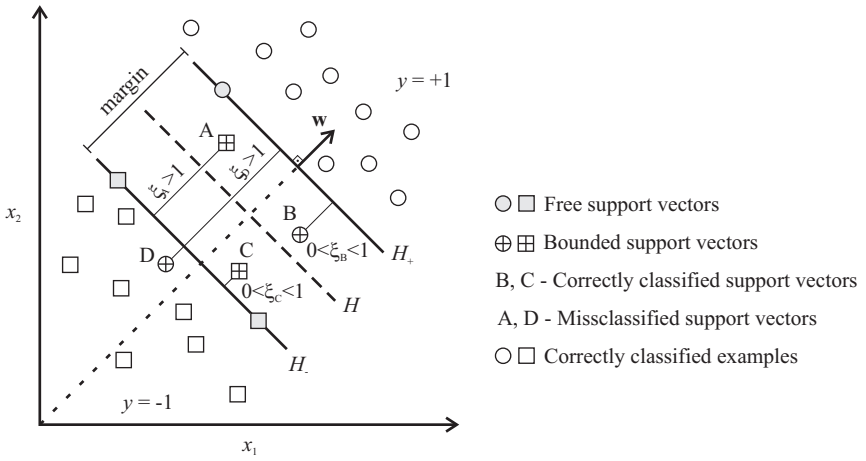

Figure 2.4: Binary classification for non-linearly separable data. Adapted from Kecman (2005), p. 20

In order to tackle this situation, new non-negative slack variables $(\xi_i, i = 1,2,\ldots,\ell)$ along with a penalization factor for wrongly classified examples (C) are introduced to the training problem, which becomes:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\cdot\sum_{i=1}^{\ell}\xi_i \tag{2.20}$$

$$\text{subject to} \quad y_i\cdot(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1,2,\ldots,\ell \tag{2.21}$$

$$\xi_i \geq 0, \quad \forall i \tag{2.22}$$

The i^{th} slack variable concerns the distance from point i to its corresponding margin bound. If it crosses such bound and is at its "wrong" side, $\xi_i > 0$. Otherwise, $\xi_i = 0$. In order to clarify the role of these slack variables, one may see Figure 2.4. The factor C, in turn, is related to the trade-off between training error and machine capacity. Actually, the problem (2.20)-(2.22) is a generalization for the training problem for linearly separable data. In the same way, the Lagrangian function is formulated and the KKT conditions are used to solve it. The Lagrangian is:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \cdot \sum_{i=1}^{\ell} \xi_i - \sum_{i=1}^{\ell} \alpha_i \cdot [y_i \cdot (\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{\ell} \beta_i \xi_i \tag{2.23}$$

in which $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are ℓ-dimensional vectors of Lagrange multipliers. Once more, it is necessary to find the saddle point $(\mathbf{w}_0, b_0, \boldsymbol{\xi}_0, \boldsymbol{\alpha}_0, \boldsymbol{\beta}_0)$ of (2.23). The KKT conditions are:

$$\frac{\partial \mathcal{L}(\mathbf{w}_0, b_0, \boldsymbol{\xi}_0, \boldsymbol{\alpha}_0, \boldsymbol{\beta}_0)}{\partial \mathbf{w}} = 0, \quad \mathbf{w}_0 = \sum_{i=1}^{\ell} \alpha_{0i} y_i \mathbf{x}_i \tag{2.24}$$

$$\frac{\partial \mathcal{L}(\mathbf{w}_0, b_0, \boldsymbol{\xi}_0, \boldsymbol{\alpha}_0, \boldsymbol{\beta}_0)}{\partial b} = 0, \quad \sum_{i=1}^{\ell} \alpha_{0i} y_i = 0 \tag{2.25}$$

$$\frac{\partial \mathcal{L}(\mathbf{w}_0, b_0, \boldsymbol{\xi}_0, \boldsymbol{\alpha}_0, \boldsymbol{\beta}_0)}{\partial \xi_i} = 0, \quad \alpha_{0i} + \beta_{0i} = C, \quad i = 1, 2, \ldots, \ell \tag{2.26}$$

$$y_i \cdot (\mathbf{w}_0^T\mathbf{x}_i + b_0) - 1 + \xi_{0i} \geq 0, \quad \forall i \tag{2.27}$$

$$\xi_{0i} \geq 0, \quad \forall i \tag{2.28}$$

$$\alpha_{0i} \geq 0, \quad \forall i \tag{2.29}$$

$$\beta_{0i} \geq 0, \quad \forall i \tag{2.30}$$

$$\alpha_{0i} \cdot [y_i \cdot (\mathbf{w}_0^T\mathbf{x}_i + b_0) - 1 + \xi_{0i}] = 0, \quad \forall i \tag{2.31}$$

$$\beta_{0i}\xi_{0i} = 0, \quad (C - \alpha_{0i}) \cdot \xi_{0i} = 0, \quad \forall i \tag{2.32}$$

If equations (2.24), (2.25) and (2.32) are replaced in (2.23), it is obtained a Lagrangian in function only of the dual variables $\alpha_i, i = 1, 2, \ldots, \ell$. The dual problem is exactly the same as the one presented for the classifier of linearly separable data, except the fact that the Lagrange multipliers α_i have now parameter C as upper bound so as to respect the non-negativity of the Lagrange multipliers β_i (see (2.32)). The problem is as follows:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \mathcal{L}_d(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, \ell \\ & \sum_{i=1}^{\ell} \alpha_i y_i = 0 \end{aligned} \tag{2.33}$$

The resulting decision function is also given by (2.19) and its signal defines the classi-

fication of an input $\mathbf{x}$ either in the positive or negative class as is presented in the previous subsection. The Lagrange multipliers, in turn, due to KKT complementarity conditions (2.31) and (2.32) have the following possible solutions:

- $\alpha_{0i} = 0, \xi_{0i} = 0$ and the input $\mathbf{x}_i$ is correctly classified.
- $0 < \alpha_{0i} < C$, then $y_i \cdot (\mathbf{w}_0^T \mathbf{x}_i + b_0) - 1 + \xi_{0i} = 0$ and $\xi_{0i} = 0$. Therefore, $y_i \cdot (\mathbf{w}_0^T \mathbf{x}_i + b_0) = 1$ and the example $(\mathbf{x}_i, y_i)$ is a support vector. The support vectors with associated Lagrange multipliers satisfying the condition $0 < \alpha_{0i} < C$ are named *free support vectors* and permit the calculation of b_0 as follows:

$$b_0 = \frac{1}{nFSV} \sum_{s=1}^{nFSV} \left(\frac{1}{y_s} - \mathbf{w}_0^T \mathbf{x}_s \right) \tag{2.34}$$

  where $nFSV$ is the number of free support vectors. Again, it is recommended to set b_0 as an average value.

- $\alpha_{0i} = C$, then $y_i \cdot (\mathbf{w}_0^T \mathbf{x}_i + b_0) - 1 + \xi_{0i} = 0$ and $\xi_{0i} \geq 0$. Hence, $(\mathbf{x}_i, y_i)$ is a *bounded support vector*, given that its related Lagrange multiplier achieves the upper bound C. Note that the slack variable is not precisely defined and because of this, bounded support vectors do not participate in the calculation of b_0. A bounded support vector is placed at the wrong side of either H_+ or H_- depending on the label y_i. For $0 < \xi_{0i} < 1$, $\mathbf{x}_i$ is correctly classified. Otherwise, if $\xi_{0i} \geq 1$, $\mathbf{x}_i$ is wrongly classified. For the sake of illustration, in Figure 2.4, A's actual label is $y = -1$, but it is at the right side of H and is then misclassified. On the other hand, B has label $y = 1$, but it is at the left side of H_+. However, because B does not pass the separating hyperplane H, it is still correctly classified. The same analysis can be done for the bounded support vectors C and D to conclude they are correctly and wrongly classified, in this order.

2.1.3 Non-Linear Classifier of Maximal Margin

When the decision function is not linear in input space, the linear classifiers previously described can not be used to separate data. For example, in Figure 2.5-a, the hyperplane has a poor performance in classifying the examples into their actual categories in the input space. On the other hand, the non-linear function perfectly accomplishes such task.

In these situations, SVM's main idea is to map the input vectors $\mathbf{x}_i \in \mathbb{R}^n$ into vectors $\Phi(\mathbf{x}_i)$ of a space of higher dimension named *feature space* ($\mathcal{F}$). The notation Φ represents the mapping $\mathbb{R}^n \rightarrow \mathcal{F}$. In $\mathcal{F}$, SVM obtains a linear separation hyperplane of maximal margin so as a linear classification is still executed but in a different space, see Figure 2.5-b.

Notice that the dual formulation (2.15) as well as the decision function (2.19), which hold for both the linearly separable and overlapping data, present input data only in form of dot

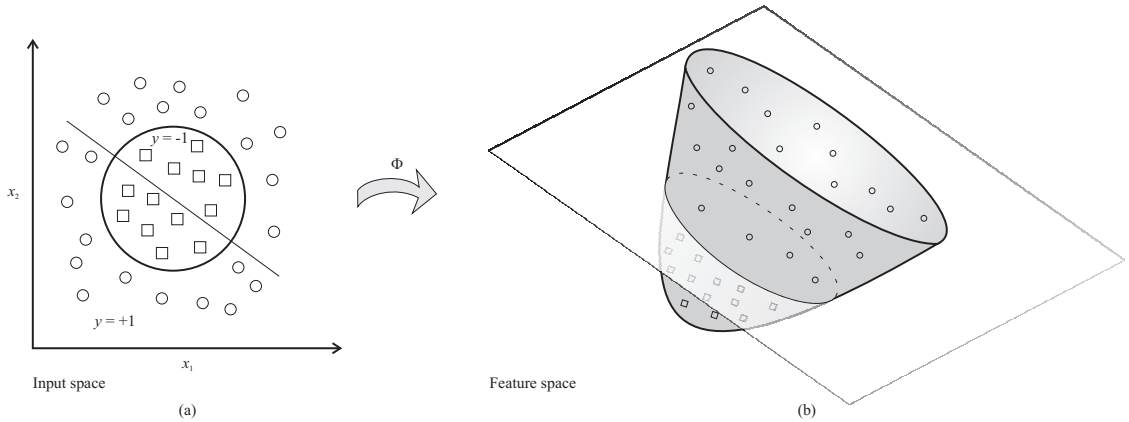

Figure 2.5: Non-linear binary classification

products $\mathbf{x}_i^T\mathbf{x}_j, i,j = 1,2,\ldots,\ell$. Then, when the mapping Φ is applied, the training algorithm becomes dependent on products of the form $\Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$. The choice of an appropriate mapping may be difficult, and besides that, the explicit calculation of the dot products $\Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$ may be computationally burdensome if the dimension of $\mathcal{F}$ is very large. The latter problem is related to the phenomenon known as *curse of dimensionality* and can be avoided by introducing kernel functions $K(\mathbf{x}_i,\mathbf{x}_j) = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$. In this way, dot products in feature space are directly calculated by computing $K(\mathbf{x}_i,\mathbf{x}_j)$, which in turn is defined on input space. Therefore, only the kernel function may be used in the training algorithm, a possibly high dimension of $\mathcal{F}$ is bypassed and explicit knowledge of the mapping Φ is not even required (BURGES, 1998; KECMAN, 2005). Some kernel functions are presented in Table 2.1.

Table 2.1: Common kernel functions. Adapted from Kecman (2001), p. 171

Kernel Function	**Type**
$K(\mathbf{x}_i,\mathbf{x}_j) = [(\mathbf{x}_i^T\mathbf{x}_j)+1]^d$	Complete polynomial of degree d
$K(\mathbf{x}_i,\mathbf{x}_j) = \exp\left[-\frac{(\mathbf{x}_i-\mathbf{x}_j)^2}{2\sigma^2}\right]$	Gaussian Radial Basis Function (RBF)
$K(\mathbf{x}_i,\mathbf{x}_j) = \tanh[(\mathbf{x}_i^T\mathbf{x}_j)+b]^*$	Multilayer perceptron

*Only for particular values of b

Kernel functions should satisfy the Mercer's conditions, which state that to describe a dot product in some $\mathcal{F}$, $K(\mathbf{x}_i,\mathbf{x}_j)$ must be symmetric and fulfill the inequality

$$\int\int K(\mathbf{x}_i,\mathbf{x}_j)g(\mathbf{x}_i)g(\mathbf{x}_j)d\mathbf{x}_i d\mathbf{x}_j > 0 \tag{2.35}$$

for all function $g(\cdot) \neq 0$ satisfying

$$\int g^2(\mathbf{x})d\mathbf{x} < \infty \tag{2.36}$$

For further details on kernel functions and Mercer's conditions, the interested reader may consult Cristiniani & Shawe-Taylor (2000), Vapnik (2000), Kecman (2001) and Schölkopf & Smola

(2002).

The generalized dual Lagrangian adapted to non-linear classifiers then becomes:

$$\mathcal{L}_d(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.37}$$

If the data is separable, constraints (2.16) and (2.17) should be taken into account, whereas if the data is non-separable, the Lagrange multipliers must be upper bounded by parameter C. Therefore, in the latter case, constraint (2.33) must be considered along with equality (2.17). The decision function (2.19) is slightly modified so as to incorporate the kernel function and the decision once again relies on its signal:

$$d(\mathbf{x}) = \mathbf{w}_0^T \Phi(\mathbf{x}) + b_0 = \sum_{i=1}^{\ell} \alpha_{0i} y_i K(\mathbf{x}_i, \mathbf{x}) + b_0 \tag{2.38}$$

To compute b_0 it is necessary to replace the dot products by the kernel function where they naturally emerge in (2.18) or (2.34), depending on the nature of training data (separable or not). Nevertheless, given that the training algorithm for overlapping data is in fact a generalization of the separable case, SVM non-linear classifiers embody the more general approach.

2.1.4 Multi-classification

The multi-classification problem is a generalization of the binary classification which entails three or more categories ($m \geq 3$). A usual technique to tackle multi-classification is to combine several binary classifiers (HSU & LIN, 2002).

One of the implementations of multi-class classifiers is the *one-against-all* approach. It constructs m decision functions by means of the resolution of m training problems of the form:

$$\min_{\mathbf{w}_k, b_k, \boldsymbol{\xi}_k} \quad \frac{1}{2} \mathbf{w}_k^T \mathbf{w}_k + C \cdot \sum_{i=1}^{\ell} \xi_{ki} \tag{2.39}$$

$$\text{subject to} \quad \mathbf{w}_k^T \Phi(\mathbf{x}_i) + b_k \geq 1 - \xi_{ki}, \quad \text{if } y_i = k, \quad i = 1, 2, \ldots, \ell \tag{2.40}$$

$$\mathbf{w}_k^T \Phi(\mathbf{x}_i) + b_k \leq -1 + \xi_{ki}, \quad \text{if } y_i \neq k, \quad \forall i \tag{2.41}$$

$$\xi_{ki} \geq 0, \quad \forall i \tag{2.42}$$

where $k = 1, 2, \ldots, m$ is the category index. Note that (2.39)-(2.42) is a general primal formulation of the training problem, since it permits errors during training step and also input vectors $\mathbf{x}_i$ are mapped into a feature space through Φ. However, transforming such problem into its dual counterpart, once more the dot products $\Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$ naturally appear and instead of them, a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ may be used. In this way, the explicit knowledge of the mapping Φ is not necessary. This same argument is valid for the decision functions resulted from training

problems.

Given an unseen input $\mathbf{x}$ (*i.e.*, not from the training set D), one has the task of selecting its category. For that, the m decision functions are calculated for the considered $\mathbf{x}$. The chosen category is the one associated with the decision function that has returned the greatest value. In summary:

$$\text{Class}(\mathbf{x}) \equiv \arg\max_k [\mathbf{w}_{0k}^T \Phi(\mathbf{x}) + b_{0k}] \tag{2.43}$$

Other approach for multi-classification using binary classifiers is the *one-against-one* method. Its main idea consists in the construction of $\frac{m(m-1)}{2}$ classifiers involving only two categories each. For instance, to train the data from classes j and k, the following optimization problem has to be solved:

$$\min_{\mathbf{w}_{jk}, b_{jk}, \boldsymbol{\xi}_{jk}} \quad \frac{1}{2}\mathbf{w}_{jk}^T \mathbf{w}_{jk} + C \cdot \sum_{i=1}^{\ell} \xi_{(jk)i} \tag{2.44}$$

$$\text{subject to} \quad \mathbf{w}_{jk}^T \Phi(\mathbf{x}_i) + b_{jk} \geq 1 - \xi_{(jk)i}, \quad \text{if } y_i = j, \quad i = 1, 2, \ldots, \ell \tag{2.45}$$

$$\mathbf{w}_{jk}^T \Phi(\mathbf{x}_i) + b_{jk} \leq -1 + \xi_{(jk)i}, \quad \text{if } y_i = k, \quad \forall i \tag{2.46}$$

$$\xi_{(jk)i} \geq 0, \quad \forall i \tag{2.47}$$

The dual formulation of such problem may be constructed and then solved as in the binary classifiers training step described in previous subsections. If in average each class has $\frac{\ell}{m}$ training examples, $\frac{m(m-1)}{2}$ problems with $\frac{2\ell}{m}$ decision variables are resolved.

In order to classify an unseen input $\mathbf{x}$, each one of the resulting $\frac{m(m-1)}{2}$ decision functions are calculated for $\mathbf{x}$. If the sign of $[\mathbf{w}_{0jk}^T \Phi(\mathbf{x}) + b_{0jk}]$ is positive, than one vote is computed for class j. Otherwise, if the sign is negative, then class k gains an additional vote. Following this reasoning, the class that has obtained the greatest number of votes, is the one to be associated to $\mathbf{x}$. This strategy of class choice is named *max-wins* strategy.

Platt *et al.* (2000) consider a similar training algorithm as the one-against-one method. The selection of class for unseen inputs ($\mathbf{x}$), however, is made by a different strategy. It is based on a Directed Acyclic Graph (DAG) whose nodes represent the binary decision functions. Then, before predicting the class of $\mathbf{x}$ it is necessary to go through a path along the DAG.

Besides the combination of binary classifiers approach to solve multi-classification problems, there are methods which consider all classes simultaneously. According to Schölkopf & Smola (2002), in terms of accuracy, the results obtained from these methods are comparable to those obtained from the methods previously described. However, the optimization problem has to deal with all support vectors at the same time and the binary classifiers, in turn, usually have smaller support vector sets, which has positive effects on training time. In addition, the authors contend that probably there is no multi-class approach that generally outperforms the others. In this way, the multi-classification method selection depends on the nature of the problem at hand, on the required accuracy and also on the available time for training. For further details in

multi-classification strategies, see Hsu & Lin (2002) and Schölkopf & Smola (2002).

Notice that both classification and multi-classification cases have the parameter C and the kernel parameter (*e.g.* σ, d) in their training problems. The values for these parameters must be defined beforehand by the user or by a systematic procedure. Indeed, this issue is detailed in Section 2.2, where the influence of these parameters in SVM performance is further discussed.

2.1.5 Support Vector Regression

In a regression, it is necessary to estimate the functional dependence between an output variable $y \in \mathbb{R}$ and an input variable $\mathbf{x} \in \mathbb{R}^n$. The main difference between classification and the regression problems is that, in the former, only discrete numbers associated with categories may be attributed to y, whilst in the latter, y assume real values, since $y = f(\mathbf{x})$ and $f : \mathbb{R}^n \to \mathbb{R}$. Similar to classification, the function estimate is based on the training of a SVM model using examples of the form (input, output). The training phase of an SVM for regression resembles the training phase of an SVM for classification purposes, given that both involve the resolution of a convex quadratic optimization problem. Nevertheless, Support Vector Regression (SVR) dual training problem entails 2ℓ decision variables, instead of only ℓ as in classification training step (SCHÖLKOPF & SMOLA, 2002; KECMAN, 2005).

Additionally, for the regression case, an analog of the soft margin is constructed in the space of the target values y by using Vapnik's linear ε-insensitive loss function (see Figure 2.6), which is defined as:

$$L(\mathbf{x}, y, f) = \max(0, |y - f(\mathbf{x})| - \varepsilon) \tag{2.48}$$

that is, the loss (cost) is zero if the difference between the predicted $f(\mathbf{x})$ and the observed y values is less than ε. Otherwise, the loss is given by the absolute difference between these two values. The Vapnik linear ε-insensitive loss function defines a "tube" of "radius" ε (Figure 2.6-a). If the observed value is within the tube, there is no computed loss, whilst for values outside the tube the cost is positive. It follows that:

$$|y - f(\mathbf{x})| - \varepsilon = \xi, \qquad \text{for data "above" the } \varepsilon\text{-tube} \tag{2.49}$$

$$|y - f(\mathbf{x})| - \varepsilon = \xi^*, \qquad \text{for data "below" the } \varepsilon\text{-tube} \tag{2.50}$$

where ξ and ξ^* are the slack variables for the mutually exclusive situations presented.

Besides the ε-insensitive loss function, there are other loss functions which can be incorporated by SVR as long as they are convex in order to ensure the convexity of the training optimization problem and then the existence (and uniqueness for strict convexity) of a minimum. For example, the quadratic ε-insensitive (2.51) loss function leads to a training problem with the same conveniences provided by the linear ε-insensitive loss function. In this work, however, only the latter loss function is considered. For more information about loss functions

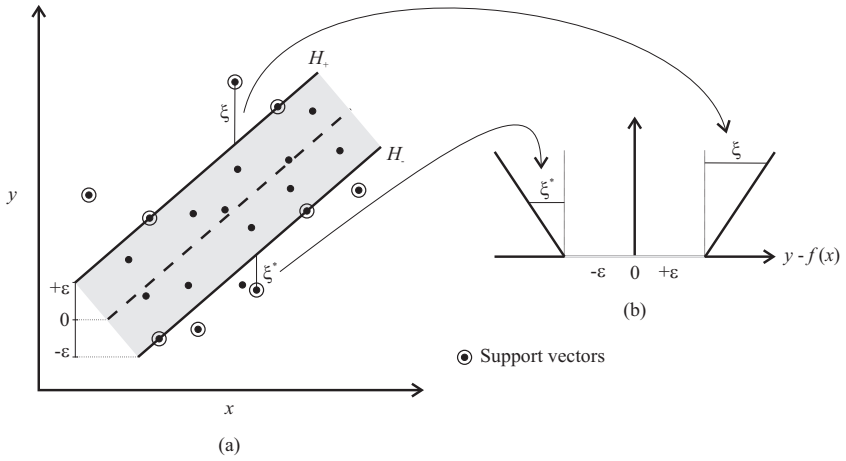

Figure 2.6: Vapnik's ε-insensitive loss function

in SVR context, consult Vapnik (2000) and Schölkopf & Smola (2002).

$$L(\mathbf{x}, y, f) = \max(0, |y - f(\mathbf{x})|^2 - \varepsilon) \tag{2.51}$$

This subsection presents the basic concepts of SVR for the case of linear and non-linear approximation functions as well as some insights of SVR applied to time series prediction.

2.1.5.1 Linear Regression Function

As in classification, the only information available for a SVR is the training data set $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)\}, \mathbf{x}_i \in \mathbb{R}^n, y \in \mathbb{R}$. It is supposed that a linear function is a good regression alternative. Then, it is necessary to find the regression hyperplane which describes best the training data, so as to allow for the use of such hyperplane to effectively regress unseen input vectors. The equation of the regression hyperplane is:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b \tag{2.52}$$

In order to find the optimal regression hyperplane, besides the training error measured by the ε−insensitive loss function, as well as in classification, it is necessary to minimize the term $\mathbf{w}^T\mathbf{w}$ related to machine capacity. A small $\mathbf{w}^T\mathbf{w}$ corresponds to a linear function that is flat (SCHÖLKOPF & SMOLA, 2002; SMOLA & SCHÖLKOPF, 2004). The primal optimization problem is then defined:

$$\min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \cdot \sum_{i=1}^{\ell} \xi_i + \xi_i^* \tag{2.53}$$

$$\text{subject to} \quad y_i - \mathbf{w}^T\mathbf{x}_i - b \leq \varepsilon + \xi_i, \quad i = 1, 2, \ldots, \ell \tag{2.54}$$

$$\mathbf{w}^T\mathbf{x}_i + b - y_i \leq \varepsilon + \xi_i^*, \quad \forall i \tag{2.55}$$

$$\xi_i \geq 0, \quad \forall i \tag{2.56}$$

$$\xi_i^* \geq 0, \quad \forall i \tag{2.57}$$

The corresponding Lagrangian is:

$$\begin{aligned}\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \cdot \sum_{i=1}^{\ell}(\xi_i + \xi_i^*) - \sum_{i=1}^{\ell}\alpha_i \cdot [\mathbf{w}^T\mathbf{x}_i + b - y_i + \varepsilon + \xi_i] \\ - \sum_{i=1}^{\ell}\alpha_i^* \cdot [y_i - \mathbf{w}^T\mathbf{x}_i - b + \varepsilon + \xi_i^*] - \sum_{i=1}^{\ell}(\beta_i\xi_i + \beta_i^*\xi_i^*)\end{aligned} \tag{2.58}$$

in which $\boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*$ are the ℓ-dimensional vectors of Lagrange multipliers associated to constraints (2.54)-(2.57) respectively. Notice that that α_i and α_i^* can not be strictly positive simultaneously, given that there is no point satisfying both (2.54) and (2.55) at the same time. Hence, $\alpha_i\alpha_i^* = 0$. The Lagrangian in (2.58) must be minimized with respect to primal variables $\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*$ and maximized with respect to dual variables $\boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*$. Then the saddle point $(\mathbf{w}_0, b_0, \boldsymbol{\xi}_0, \boldsymbol{\xi}_0^*, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_0^*, \boldsymbol{\beta}_0, \boldsymbol{\beta}_0^*)$ has to be found. The KKT conditions are:

$$\frac{\partial\mathcal{L}(\mathbf{w}_0, b_0, \boldsymbol{\xi}_0, \boldsymbol{\xi}_0^*, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_0^*, \boldsymbol{\beta}_0, \boldsymbol{\beta}_0^*)}{\partial\mathbf{w}} = 0, \quad \mathbf{w}_0 = \sum_{i=1}^{\ell}(\alpha_{0i} - \alpha_{0i}^*)\mathbf{x}_i \tag{2.59}$$

$$\frac{\partial\mathcal{L}(\mathbf{w}_0, b_0, \boldsymbol{\xi}_0, \boldsymbol{\xi}_0^*, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_0^*, \boldsymbol{\beta}_0, \boldsymbol{\beta}_0^*)}{\partial b} = 0, \quad \sum_{i=1}^{\ell}(\alpha_{0i} - \alpha_{0i}^*) = 0 \tag{2.60}$$

$$\frac{\partial\mathcal{L}(\mathbf{w}_0, b_0, \boldsymbol{\xi}_0, \boldsymbol{\xi}_0^*, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_0^*, \boldsymbol{\beta}_0, \boldsymbol{\beta}_0^*)}{\partial\xi_i} = 0, \quad C - \alpha_{0i} = \beta_{0i}, \quad i = 1, 2, \ldots, \ell \tag{2.61}$$

$$\frac{\partial\mathcal{L}(\mathbf{w}_0, b_0, \boldsymbol{\xi}_0, \boldsymbol{\xi}_0^*, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_0^*, \boldsymbol{\beta}_0, \boldsymbol{\beta}_0^*)}{\partial\xi_i^*} = 0, \quad C - \alpha_{0i}^* = \beta_{0i}^*, \quad \forall i \tag{2.62}$$

$$\mathbf{w}_0^T\mathbf{x}_i + b_0 - y_i + \varepsilon + \xi_{0i} \geq 0, \quad \forall i \tag{2.63}$$

$$y_i - \mathbf{w}_0^T\mathbf{x}_i - b_0 + \varepsilon + \xi_{0i}^* \geq 0, \quad \forall i \tag{2.64}$$

$$\xi_{0i} \geq 0, \quad \forall i \tag{2.65}$$

$$\xi_{0i}^* \geq 0, \quad \forall i \tag{2.66}$$

$$\alpha_{0i} \geq 0, \quad \forall i \tag{2.67}$$

$$\alpha_{0i}^* \geq 0, \quad \forall i \tag{2.68}$$

$$\beta_{0i} \geq 0, \quad \forall i \tag{2.69}$$

$$\beta_{0i}^* \geq 0, \quad \forall i \tag{2.70}$$

$$\alpha_{0i} \cdot [\mathbf{w}_0^T\mathbf{x}_i + b_0 - y_i + \varepsilon + \xi_{0i}] = 0, \quad \forall i \tag{2.71}$$

$$\alpha_{0i}^* \cdot [y_i - \mathbf{w}_0^T\mathbf{x}_i - b_0 + \varepsilon + \xi_{0i}^*] = 0, \quad \forall i \tag{2.72}$$

$$\beta_{0i}\xi_{0i} = 0, \quad (C - \alpha_{0i}) \cdot \xi_{0i} = 0, \quad \forall i \tag{2.73}$$

$$\beta_{0i}^*\xi_{0i}^* = 0, \quad (C - \alpha_{0i}^*) \cdot \xi_{0i}^* = 0, \quad \forall i \tag{2.74}$$

By replacing equalities (2.59)-(2.60) in (2.58), a Lagrangian in function only of the dual vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$ is obtained. As in classification, the dual optimization problem may be solved:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad \mathcal{L}_d(\boldsymbol{\alpha},\boldsymbol{\alpha}^*) = -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\mathbf{x}_i^T\mathbf{x}_j - \sum_{i=1}^{\ell}[\varepsilon(\alpha_i+\alpha_i^*)+y_i(\alpha_i-\alpha_i^*)] \tag{2.75}$$

$$\text{subject to} \quad \sum_{i=1}^{\ell}(\alpha_i-\alpha_i^*)=0 \tag{2.76}$$

$$0\le\alpha_i\le C, \quad i=1,2,\ldots,\ell \tag{2.77}$$

$$0\le\alpha_i^*\le C, \quad \forall i \tag{2.78}$$

As a result from the dual problem, non-negative values for the Lagrange multipliers α_i and α_i^* for all i are obtained. From them, the optimal normal vector $\mathbf{w}_0$ is directly defined by (2.59). By the possible values for α_{0i} and α_{0i}^* as well as considering the KKT complementarity conditions (2.71)-(2.74), one may conclude:

- If $0<\alpha_{0i}<C$, then $\xi_{0i}=0$ is true. Besides that, the equality $\mathbf{w}_0^T\mathbf{x}_i+b-y_i=-\varepsilon$ is obtained and then the example $(\mathbf{x}_i,y_i)$ intercepts the parallel hyperplane that is ε above the regression hyperplane (H_+). Similarly, if $0<\alpha_{0i}^*<C$, from $\xi_{0i}^*=0$ holds and consequently the equality $y_i-\mathbf{w}_0^T\mathbf{x}_i-b=-\varepsilon$ is valid. Hence, the point $(\mathbf{x}_i,y_i)$ intercepts the hyperplane that is ε below the regression hyperplane (H_-). When either $0<\alpha_{0i}<C$ or $0<\alpha_{0i}^*<C$ is true, the related example $(\mathbf{x}_i,y_i)$ is named *free support vector*, which allow for the calculation of the linear coefficient b_0:

$$b_0=\frac{1}{nFSV}\left[\sum_{s=1}^{nUFSV}(y_s-\mathbf{w}_0^T\mathbf{x}_s-\varepsilon)+\sum_{s=1}^{nLFSV}(y_s-\mathbf{w}_0^T\mathbf{x}_s+\varepsilon)\right] \tag{2.79}$$

  where $nUFSV$ and $nLFSV$ are respectively the number of free support vectors lying in H_+ and H_-.

- For data "above" the ε-tube, *i.e.* $\xi_{0i}>0$, the equality $\alpha_{0i}=C$ holds. On the other hand, if data is "below" the ε-tube, then $\xi_{0i}^*>0$, which implies $\alpha_{0i}^*=C$. Training data that satisfy these conditions are called *bounded support vectors*. As in the case of soft-margin classifier, these support vectors do not construct the value of b_0, since it can not be uniquely determined due to positive but not exactly known slack variables values.

- For training examples lying "within" the ε-tube, $|y_i-\mathbf{w}_0^T\mathbf{x}_i-b|<\varepsilon$ is valid and as a consequence $\alpha_{0i}=\alpha_{0i}^*=0$. Such points are not support vectors and also do not construct the regression hyperplane, which is defined as follows:

$$f(\mathbf{x})=\mathbf{w}_0^T\mathbf{x}+b_0=\sum_{i=1}^{\ell}(\alpha_i-\alpha_i^*)\mathbf{x}_i^T\mathbf{x}+b_0 \tag{2.80}$$

2.1.5.2 Non-Linear Regression Function

The generalization from linear to non-linear regression functions is possible by the use of kernel functions in the same way linear classifiers evolve to non-linear ones. In this way, even if the regression function is non-linear in input space, a regression hyperplane can be found in a feature space.

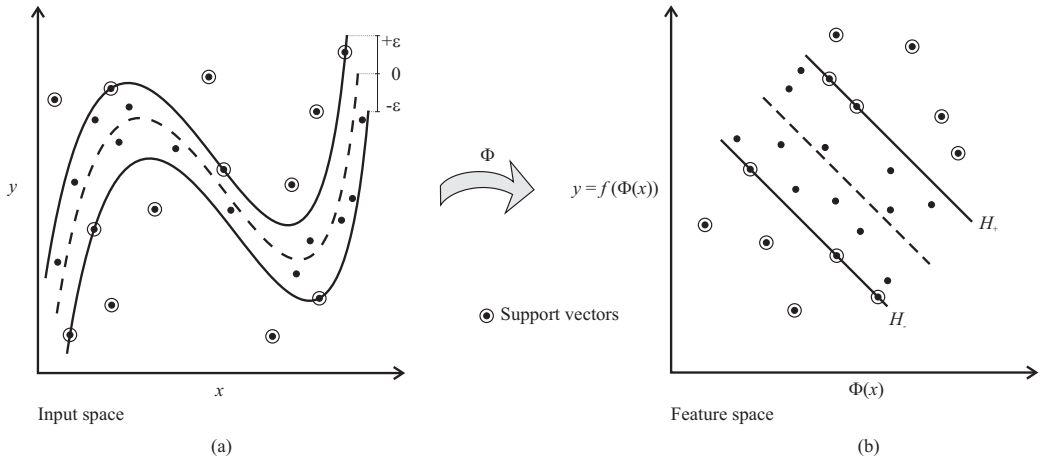

Figure 2.7: Non-linear regression

Once more notice that the dual training problem in subsection 2.1.5.1 presents input data in the form of dot products $\mathbf{x}_i^T\mathbf{x}_j$. Analogous to non-linear classifiers, such dot products are replaced by a kernel function $K(\mathbf{x}_i,\mathbf{x}_j) = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$ set *a priori*. The normal vector $\mathbf{w}_0$ may not be directly obtained, since its expression (2.81) becomes function of the mapping Φ, which often involves a high dimension or it is even not known. Both cases render the explicit computation of $\mathbf{w}_0$ impractical.

$$\mathbf{w}_0 = \sum_{i=1}^{\ell}(\alpha_{0i} - \alpha_{0i}^*)\Phi(\mathbf{x}_i) \tag{2.81}$$

The linear coefficient b_0, in turn, can be calculated by either the KKT complementarity conditions (2.71) or (2.72). After replacing $\mathbf{w}_0$ in these equations, the dot product $\Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$ naturally appears and may be substituted by $K(\mathbf{x}_i,\mathbf{x}_j)$. Again, it is better to set b_0 as the average over all calculated expressions (KECMAN, 2005).

Therefore, one may have the non-linear regression function:

$$f(\mathbf{x}) = \mathbf{w}_0^T\Phi(\mathbf{x}) + b_0 = \sum_{i=1}^{\ell}(\alpha_{0i} - \alpha_{0i}^*)\Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}) + b_0 = \sum_{i=1}^{\ell}(\alpha_{0i} - \alpha_{0i}^*)K(\mathbf{x}_i,\mathbf{x}) + b_0 \tag{2.82}$$

SVR and traditional statistical (non)-linear regression models are different approaches to solve regression problems. Firstly, traditional models are classified as (non)-linear with respect to the parameters of the regression function. SVR, however, is classified as (non)-linear de-

pending on the (non)-linearity on the input variables ($\mathbf{x}$). One advantage of SVR over statistical linear regression models is the fact that the errors do not need to be drawn from a Normal distribution with zero mean and constant variance. In fact, SVR does not require assumptions over the errors. Additionally, differently from SVR, to apply a statistical non-linear regression model, a previous knowledge about the relationship between the response and input variables is often required. Also, due to the use of kernels, SVR always involve a linear regression function, either in the input or in the feature space. For more in traditional statistical (non)-linear regression methods, see Montgomery *et al.* (2001).

2.1.5.3 Time Series Prediction

According to Fuller (1996), a real valued time series can be considered as a collection of random variables indexed in time. For a specific event under analysis (*e.g.* system failure), the realizations of such random variables generate a set of observations ordered in time. Some of the most common purposes of studying time series are to learn about the underlying mechanism generating the data, to predict future values of the phenomenon under analysis and/or to optimally control a system. For instance, financial series of stock prices may provide investors with information whether or not they may invest in the next future. Also, a series of reliability values of a critical machine gives insights of its "health", which is very useful for maintenance planning.

Generally, a time series is not drawn independently. Conversely, the statistical learning model underlying SVR assumes independent and identically distributed samples. Despite this fundamental difference mentioned by Schölkopf & Smola (2002), SVR has been widely applied to the problem of time series prediction and has provided excellent results for them, comparable to or even better than the ones originated from other approaches such as NN. This empirical evidence can be verified in the work by Müller *et al.* (1999) that shows the superior performance of SVR in a benchmark set from the Santa Fe Time Series Competition (WEIGEND & GERSHENFELD, 1994) as well as in the reliability related papers from Hong & Pai (2006) and Chen (2007). Moreover, a survey of works using SVM for time series prediction applied to diverse fields is presented by Sapankevych & Sankar (2009).

The general time series prediction model can be represented as follows:

$$y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-p}) \tag{2.83}$$

where y_t is the observation at time t which is function of the p past observations. In other words, the input vector $\mathbf{x}_t = (y_{t-1}, y_{t-2}, \ldots, y_{t-p})$, where p denotes its dimension, is directly related to the future value y_t. Additionally, if t data observations compose the time series, one may have to construct the examples in the form $(\mathbf{x}_i, y_i), i = 1, 2, \ldots t-p$ as shown in Table 2.2. In this way, the training set is comprised by $t-p$ training examples.

Table 2.2: Construction of data pairs for time series prediction

i	**Input** $\mathbf{x}_i$				**Output** y_i
1	y_1	y_2	$\cdots$	y_p	y_{p+1+k}
2	y_2	y_3	$\cdots$	y_{p+1}	y_{p+2+k}
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
$t-p$	y_{t-p}	y_{t-p+1}	$\cdots$	y_{t-1}	y_{t+k}

Besides that, the future outcomes may be predicted for different steps in time. In Table 2.2, k is the number of steps ahead minus one. For example, if the $(i-1)$th values are inputs to forecast the i^{th} one, it is then performed a single-step-ahead prediction ($k=0$). If the same inputs are used to predict the $(i+1)$th outcomes, a two-step-ahead forecast takes place ($k=1$). Generally, predicting two or more steps ahead with $(i-1)$th input values is said to be a multi-step-ahead forecast.

Once the training examples are formed, the application of SVR follows the same reasoning shown in subsections 2.1.5.1 and 2.1.5.2. An advantage of using SVR approach in dealing with time series prediction is the fact that it is model-independent and can tackle non-linear and non-stationary series, which may not be handled by traditional methods if simplifying assumptions or alternative techniques to render them stationary are not considered. Basically, a non-stationary time series randomly varies along time around a constant mean, reflecting a form of equilibrium (MORETTIN & TOLOI, 2004).

As in the (multi-)classification problems, SVR also depends on a set of parameters that has to be defined *a priori*. Besides the penalization for errors C and the kernel parameter (*e.g.* σ in RBF case), SVR also demands the definition of ε from the Vapnik's ε-insensitive loss function. In both classification and regression tasks, the choice of these parameters is often very difficult. This issue gives rise to the model selection problem, which is discussed in Section 2.2. The following Subsection introduces the most used optimization techniques to solve the SVM training problem.

2.1.6 Optimization Techniques and Available Support Vector Machines Libraries

Different optimization techniques can be applied to the SVM learning problem. One of them is the Interior Point Method (NOCEDAL & WRIGHT, 2006), that is indicated for small to moderately sized data sets, up to 10^4 (SCHÖLKOPF & SMOLA, 2002). Alternatively, the faster Sequential Minimal Optimization (SMO) approach can be adopted. Loosely speaking, SMO decomposes the quadratic training problem into the smallest possible quadratic subproblem, which involves only two examples (two Lagrange multipliers) so as to allow for the analytic treatment of them instead of numerical. At every step, SMO chooses two Lagrange multipliers

to jointly optimize, finds the optimal values for these multipliers, and updates the SVM to reflect the new optimal values. Since the storage of large matrices is not required, large data sets can be handled and problems with numerical precision are avoided (PLATT, 1998).

There are several free SVM solvers available in the Internet, which are periodically improved by the developers. Two of them are the SVM^{light} and the LIBSVM. SVM^{light} is an implementation of SVM learner that solves the SVM dual problem by means of a decomposition strategy that generates a series of smaller optimization problems to be resolved. The dual decision variables ($\boldsymbol{\alpha}$) are divided in two groups: the first one comprises the variables that can vary their values during the optimization process and forms the so-called working set; the second one is formed by variables with fixed values. The selection of which variables will be in the working set is made through the choice of the steepest descent feasible direction, which in turn will make much progress towards the minimum of the objective function. For further details, see Joachims (1999).

LIBSVM is an integrated software for support vector classification, multi-classification and regression tasks. It can be easily linked with other programs through several programming languages such as MATLAB. Additionally, it implements a SMO algorithm for solving the training problem (CHANG & LIN, 2001; FAN *et al.*, 2005). In this dissertation, LIBSVM is the used library for training of SVM, as well as for prediction via a particular SVM already trained.

SVM performance is influenced by the values of the parameter C, ε (regression case) and kernel parameter (*e.g.* σ, d). The problem of selecting the most suitable set of parameters is detailed in next section.

2.2 Model Selection Problem

The performance of SVM strongly depends on the chosen set of parameters. The task of tuning the SVM parameters so as to obtain the most suitable set of values for them is known as the *model selection problem*. In classification problems, the trade-off between model capacity and training error represented by C and the kernel parameter (*e.g.* the RBF width σ or the polynomial degree d) are the user defined parameters. For instance, a large C forces the SVM classification algorithm to reduce the training errors, which in turn can be accomplished by increasing the machine capacity (by means of $\mathbf{w}^T\mathbf{w}$) and as a consequence may reduce the margin. This is contrary to the main objective of margin maximization and also does not guarantee a good generalization performance of the classifier.

Regression problems, along with C and the kernel parameter, present the "radius" ε of the tube. For a small C, penalty on errors gets negligible and regression function becomes flat, while for a large C a penalty gets more important and the resulting regression function tries to fit the data. A small ε inclines the SVR function to fit the data, since a very thin ε-tube may be not sufficiently wide to include even few data points. A large ε-tube, however, may be wide enough, which renders the SVR function flat and, as a consequence, it may not describe well

the training set. If a RBF kernel is considered, a small σ means the kernel is more localized. Thus, the SVR function has a tendency to overfit, while a large σ makes the ε-tube less flexible (ITO & NAKANO, 2003). In Figure 2.8, it is illustrated the effects of small and large values of the parameters for the regression case.

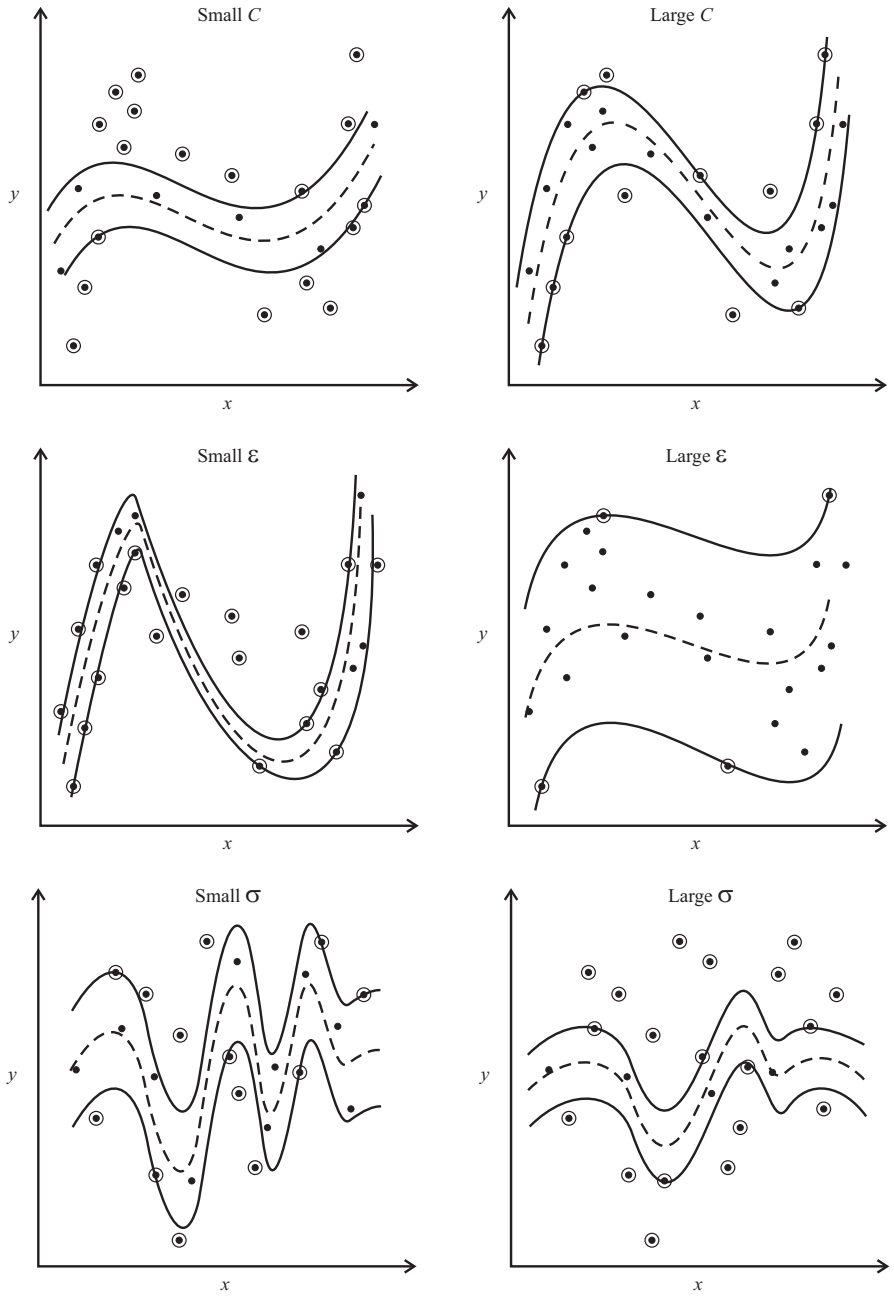

Figure 2.8: Influence of parameters in SVR. Adapted from Ito & Nakano (2003), p. 2078

Given the influence of parameters on SVM performance, it is necessary to select them as accurately as possible. The most naive attempt to set these parameters is the trial and error method, which is time consuming and does not ensure a useful set will be found. More educated ways to tune these parameters are then listed:

- Grid search: ranges of parameters are defined and then made discrete so as to allow for the

test of all possible combinations. This procedure may be suitable if only few parameters are to be tuned, but otherwise may be time consuming and may become impractical if there are several parameters as well as many possible values for each one of them. Hsu & Lin (2002) use this method for adjusting the SVM parameters in the context of multi-classification problems.

- Pattern search: is a direct search method, suitable to optimize a wide range of objective functions, given that it does not require derivatives (LEWIS *et al.*, 2000). The number of evaluated combinations is smaller than the quantity assessed in the grid-search method. It was applied to SVR model selection in the context of drug designs by Momma & Bennett (2002). The authors also make a comparison between the pattern search and the grid search.

- Gradient-based search: requires continuous and differentiable error functions. Vapnik & Chapelle (2000) derive some generalization error bounds for classification and in the later work by Chapelle *et al.* (2002) a gradient descent method is applied to obtain the set of parameters that minimize those bounds. Chang & Lin (2005) present some error bounds for regression and use a quasi-Newton method to get the parameters minimizing them. As stated by Ito & Nakano (2003) the linear ε loss function does not produce a smooth error surface, which burden gradient methods. Alternatively, the authors incorporate the quadratic ε loss function to SVR and then present the derivatives of the error function with respect to the parameters. For details in gradient-based methods see Nocedal & Wright (2006).

- Bayesian evidence framework: its main idea is to maximize the posterior probability of the parameter distribution to get the optimal parameter. The evidence framework was adapted to SVM model selection by Kwok (2000). The author describes the methodology for classification problems. Later, Gestel *et al.* (2001) and Yan *et al.* (2004) apply evidence framework to regression situations. The first work is related to financial series forecasting. The second one is associated to the refinery oil industry and aims at estimating the freezing point of light diesel oil in a fractionator by means of process data records.

- Probabilistic search heuristics: are flexible optimization techniques that do not require derivatives and are often based on nature aspects. For example, GA and PSO are widely applied to optimization problems from different contexts. Specifically in SVM field, Pai (2006) and Chen (2007) apply GA to select the SVR parameters to forecast series related to reliability of engineered components. Additionally, in the electricity management field, PSO is used by Fei *et al.* (2009) in a SVR to predict the quantity of gases dissolved in power transformer oil based on previous observed values and by Hong (2009) in selecting the parameter of a SVR to forecast electric load. In the domain of fault detection

Samanta & Nataraj (2009) apply PSO to choose the kernel parameter of the SVM classifier. Simulated Annealing (SA) may also be part of this group and is applied to SVR forecast electricity load and software reliability by respectively Pai & Hong (2005) and Pai & Hong (2006).

Other systematic manners to find the parameter values so as to allow for generalization error minimization are available in literature. The interested reader may consult for instance Fröhlich & Zell (2005) for SVM classification and regression parameters selection and Wu & Wang (2009) for the classification case only. In this work, the PSO approach is adopted to select the parameter values of SVM, given it is well-suited to optimize real-valued functions as well as does not require derivatives, which avoids problems with non-smooth error surfaces. In addition, if compared to GA, for example, PSO requires less computational effort.

In practice, the data set is divided in two parts, one for actual training (training set formed by ℓ elements) and the other for posterior test (test set comprised by λ examples). The test set plays the role of unseen data and are not used during SVM training. Instead, it is used to evaluate an error function usually involving the real output values (y_h) and the predicted ones ($\hat{y}_h$) resulted from the trained SVM model, where $h = 1, 2, \ldots, \lambda$ is the index of an example from the test set. Two error functions commonly applied for SVM testing are the Normalized Root Mean Square Error (NRMSE) and the Mean Absolute Percentage Error (MAPE). Eventually the Mean Square Error (MSE) is also used. These error functions are defined as follows:

$$\text{NRMSE} = \sqrt{\frac{\sum_{h=1}^{\lambda}(y_h - \hat{y}_h)^2}{\sum_{h=1}^{\lambda} y_h^2}} \tag{2.84}$$

$$\text{MAPE} = \frac{1}{\lambda}\sum_{h=1}^{\lambda}\left|\frac{y_h - \hat{y}_h}{y_h}\right| \cdot 100\% \tag{2.85}$$

$$\text{MSE} = \frac{1}{\lambda}\sum_{h=1}^{\lambda}(y_h - \hat{y}_h)^2 \tag{2.86}$$

In order to select the most suitable parameters, the mentioned error functions may be iteratively evaluated on the test set so as to guide the quest for improved parameter values. Nevertheless, a unique test set may mislead to an error estimate far from the real generalization error. In this way, specially the grid, pattern and probabilistic search methodologies incorporates during the training phase a sort of model validation, in which the training set is split into even smaller subsets that participate in either actual training or model validation in an alternated manner. One of the following training strategies may be adopted:

- Validation sets: the training set is divided into two subsets, one for actual training and a second one to guide the search for optimal parameter values (validation set). Once the parameters are found, the machine's generalization performance can be evaluated on a

test set. It is expected a test error greater than the validation error, since the machine would be specialized in the validation set. However, with this procedure, one may have insights about the machine ability to generalize. Hong & Pai (2006) and Pai (2006) apply this strategy. Also, instead of one single validation set, Chen *et al.* (2004), in their work related to electricity load prediction, make use of two validation sets and the quest for the SVR parameters are based on the mean validation error resulted from both of them.

- Cross-validation: the training set is split randomly into several subsets, say k. A k-fold cross-validation consists in training the SVM model with $k-1$ subsets and validate it on the remaining subset by means of an error function. This is made k times so as each subset participate in the validation phase once. The error is averaged over the k validations (equations (2.84) and (2.85) may be multiplied by $\frac{1}{k}$). This procedure is more time consuming than the single test set approach but it gives a better estimate of the generalization error.

- Leave-one-out: this is the extreme of the cross-validation strategy. If the training set is formed by ℓ examples, ℓ trainings are performed. In each one of them $\ell-1$ examples actually participate in the training phase and the remaining one is used for model validation. Hence, every example participate in validation once. This approach is the most time-consuming but, according to Schölkopf & Smola (2002) it provides an almost unbiased estimate of the error. Additionally, Lee *et al.* (2004) present a manner to enhance the efficiency of this procedure for Gaussian kernel-based SVM.

It is important to emphasize that the described strategies occur during the training phase, which can be didactically divided in actual training and validation. After that phase, the set of parameters which return the minor error value may be adopted and are then often used to train the entire training set (*i.e.* all ℓ examples at once). This procedure has some theoretical issues specially associated with cross-validation that are described by Schölkopf & Smola (2002). For example, given that the retraining with all examples makes use of the same data which guided the search for parameters, it can lead to overfitting. Also the optimal parameter settings for data sets of size $\frac{(k-1)\ell}{k}$ and ℓ do not usually coincide. Nevertheless, applications not considering these potential problems are frequently performed with promising results. Finally, after setting the parameters, the SVM model is tested on test set.

Additionally, it is noteworthy that when handling time series data it is important to realize that the entries are inherently indexed on time. This is crucial specially when dealing with non-stationary time series, where the mixing of data may lead to completely different realizations of the underlying phenomenon at hand. In this way, choosing random subsets to apply cross-validation for parameter selection is not an indicated technique for these cases, even if those subsets could be internally sorted or divided in approximate equal time intervals. Given that a subset is extracted from the training in order to validate the model, the machine would not be

able to catch the existent correlation from the original data. As an alternative, the validation sets approach respecting the time series order can be adopted by dividing data set into training, validation and test sets considering the original order, so as to allow for performance evaluations of the machine ability to predict future outcomes. The leave-one-out procedure could also be applied since only one data entry at a time is excluded from training with possibly no serious influence on the natural relationship among data. However, this latter approach has the cost of high computational effort, which depending on the parameter search tool used may be prohibitive.

The majority of the application examples considered in this work are related to reliability time series (Chapter 4). Thus, along with PSO, this work makes use of the validation sets approach. Even for the example concerning real data from oil production wells the validation sets procedure is adopted, since the cross-validation and leave-one-out techniques are time consuming. In next section, the main ideas and characteristics of PSO are described.

2.3 Particle Swarm Optimization

The PSO algorithm was introduced by Kennedy & Eberhart (1995) and it is based on the social behavior of biological organisms that move in groups such as birds and fishes. It was originally developed to solve non-linear optimization problems. PSO also has some ties to evolutionary algorithms such as GA, since it is population-based (swarm). However, a fundamental difference between these paradigms is the fact that evolutionary algorithms are based on natural evolution concepts, which embody a competitive philosophy denoted by the idea that only the fittest individuals tends to survive. Conversely, PSO incorporates a cooperative approach to solve a problem, given that all individuals (particles) – which are allowed to survive – change themselves over time and one particle's successful adaptation is shared and reflected in the performance of its neighbors (KENNEDY *et al.*, 2001).

The basic element of PSO is a particle, which can fly throughout search space towards an optimum by using its own information as well as the information provided by other particles comprising its neighborhood. The performance of a particle is determined by its *fitness*, which is assessed by calculating a predefined objective function related to the problem to be solved at its current position.

In PSO, a particle's neighborhood is the subset of particles with which it is able to communicate (BRATTON & KENNEDY, 2007). Depending on how the neighborhood is determined, the PSO algorithm may embody the *gbest* model, where each particle is connected to every other particle in the swarm so as it can obtain information from them. In other words, the neighborhood of a particle is the entire swarm. Alternatively, in the *lbest* model a particle is not able to communicate with all other particles but only with some of them. The simplest *lbest* model, also known as *ring* model, connects each particle to only two other particles in the swarm. It is important to notice that the neighborhood concept gives rise to a communication

network among particles which does not necessarily depends on distances. Indeed, it is better to think about the swarm communication network as a graph where vertices represent particles and edges indicate the connections among them without any associated weight. Then, the *gbest* model is related to a complete graph in which all vertices (*i.e.* particles) are connected to each other, whereas the *ring* model forms a cycle with length equal to the number of particles. For example, the left side of Figure 2.9 depicts a cycle of length 12. Other types of swarm communication networks are also shown in Figure (2.9). For more on graphs, see Bondy & Murty (2008).

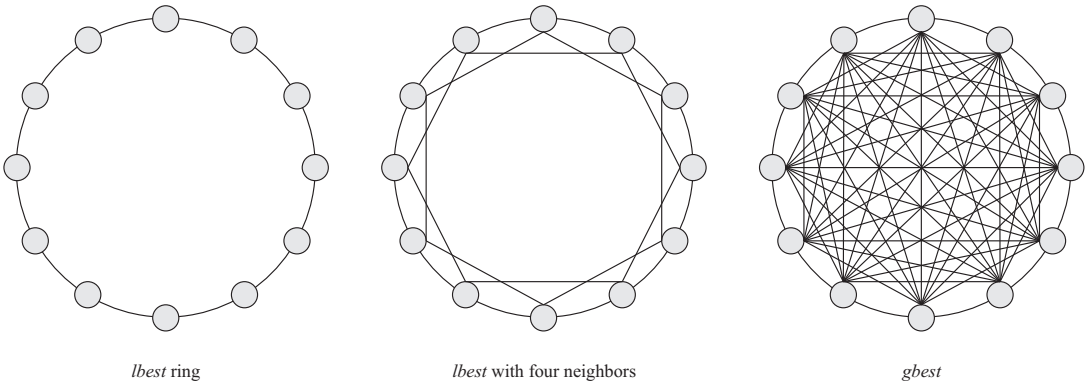

Figure 2.9: Different swarm communication networks. Adapted from Bratton & Kennedy (2007)

If Euclidean distances are used to define particles' neighbors, the communication networks' inherent cycles as depicted on Figure 2.9 are not guaranteed. This may eventually lead to a lack of exploration ability, which is not interesting.

According to Bratton & Kennedy (2007), the *gbest* model usually converges more rapidly than the *lbest* approach. Sometimes this characteristic may be actually a drawback of the former model, since it can eventually result in premature convergence of the algorithm to a inferior local in the case of multi-modal functions. However, in some cases the *gbest* model can deliver competitive performance even on complex multi-modal problems. Additionally, the *gbest* model has the advantage that it often requires less function evaluations, which is very useful when such assessments are computationally costly. In fact, as the number of particles' neighbors increases, one may get a mix of both advantages and shortcomings of *lbest* and *gbest* approaches (EBERHART & KENNEDY, 1995).

A particle i is formed by three vectors:

- Its current position in search space $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$.
- The best individual position it has found so far $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{in})$.
- Its velocity $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{in})$.

Traditionally the current positions $\mathbf{x}_i$ and velocities $\mathbf{v}_i$ are initialized respectively by means of a uniform distribution parametrized by the search space and by the maximum velocity v_{max}.

The particles then move throughout the search space by the following set of recursive update equations:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 u_1 \cdot [p_{ij}(t) - x_{ij}(t)] + c_2 u_2 \cdot [p_{gj}(t) - x_{ij}(t)], \quad j = 1,2,\ldots,n \quad (2.87)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \quad \forall j \quad (2.88)$$

where c_1 and c_2 are constants, u_1 and u_2 are independent uniform random numbers from the interval [0,1] generated at every update for each individual dimension $j = 1,2,\ldots,n$ and $\mathbf{p}_g(t)$ is the n-dimensional vector formed by the best position encountered so far by any neighbor of particle i. Note that velocities and positions at time $t+1$ are influenced by the distances of the particle's current position from its own best experience $\mathbf{p}_i(t)$ and the neighborhood's best experience $\mathbf{p}_g(t)$. The second part of (2.87) represents the "cognition" of the particle, which is its private "thinking". The last part of (2.87), in turn, is associated with the particle's "social" ability, which represents the collaboration among particles. Notice also that velocities and positions are part of the same equation, even though units of both being different (length per time unit and length, respectively). One may interpret the future position as the previous one plus the velocity multiplied by a time unit. In this way, there is no problem involving units in equations (2.87) and (2.88), as it would be thought at first glance.

During the iterations, if velocities are not constrained to an upper bound (v_{max}), the PSO algorithm is prone to enter a state of explosion, since the random weighting of u_1 and u_2 causes velocities and thus particle's positions to increase rapidly. In this way, the v_{max} approach, illustrated in the following pseudocode, can be used for every dimension j and particle i. According to Bratton & Kennedy (2007), however, a single value v_{max} is not necessarily applicable to all sizes of problem spaces and finding its appropriate value is critical to the PSO performance and it may be a difficult task.

```
procedure CONSTRAINVELOCITY(v_max)
    if v_ij(t+1) > v_max then
        v_ij(t+1) = v_max
    else
        if v_ij(t+1) < -v_max then
            v_ij(t+1) = -v_max
end procedure
```

Alternatively, the *inertia weight* w may replace v_{max} to adjust the influence of previous velocity of the particle during the optimization process and then to balance the trade-off between global and local search. By adjusting w, the swarm has a tendency to eventually constrict itself to the region containing the best fitness and exploit this region (SHI & EBERHART, 1998; BRATTON & KENNEDY, 2007). The function for velocity update (2.87) becomes:

$$v_{ij}(t+1) = w\, v_{ij}(t) + c_1 u_1 \cdot [p_{ij}(t) - x_{ij}(t)] + c_2 u_2 \cdot [p_{gj}(t) - x_{ij}(t)], \quad j = 1,2,\ldots,n \quad (2.89)$$

It is also possible to vary the value of w during PSO iteration, which may be greater at the beginning so as to allow for exploration (coverage of the entire search space) and gradually smaller while the algorithm evolves to encourage exploitation (fine adjustments in a specific area) on the most promising regions found during exploration. If w is a constant, Shi & Eberhart (1998) recommend to pick a value for it from the interval $[0.9, 1.2]$. On the other hand, if it is changed dynamically, typically it varies from 0.9 to 0.4 throughout PSO iterations (KENNEDY *et al.*, 2001).

Another manner to avoid the velocity explosion during PSO iterations is to use a *constriction factor* χ multiplying all parts of the velocity update equation:

$$v_{ij}(t+1) = \chi \left\{ v_{ij}(t) + c_1 u_1 \cdot [p_{ij}(t) - x_{ij}(t)] + c_2 u_2 \cdot [p_{gj}(t) - x_{ij}(t)] \right\}, \quad j = 1, 2, \ldots, n \tag{2.90}$$

where

$$\chi = \frac{2}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|}, \quad \varphi = c_1 + c_2 \tag{2.91}$$

Notice that χ has no real values when $\varphi < 4$. The idea of the constriction factor is that the amplitude of particles' oscillations decreases as they focus on a previous best point from their respective neighborhoods. In this way, as φ increases, χ decreases and such amplitudes become even smaller. However, if a member of a neighborhood finds a better point, the other particles can perfectly explore the new region. Hence, the constriction factor does not forbid particles switching from exploratory to exploitative mode and vice versa (KENNEDY *et al.*, 2001). Bratton & Kennedy (2007) affirms that for simplicity most implementations of constricted PSO set $\varphi = 4.1$, which assures convergence and yields $\chi \approx 7.2984 \cdot 10^{-1}$, $c_1 = c_2 = 2.05$.

Although the constriction factor does not require a limit on particles' velocities, empirical experiments have shown that taking the variables' ranges as bounds for velocities can provide better results. Limiting velocities this way does not confine particles to feasible search space, but in general does not allow them to go far beyond the region of interest (KENNEDY *et al.*, 2001). Actually, artificially constraining particles' positions when they reach the boundary of search space is not recommended since it can affect the performance of PSO. In this way, infeasible particles may emerge and then the most straightforward method for handling them is to leave their velocities and infeasible positions unaltered. Besides that, the fitness evaluation step may be skipped in order to avoid infeasible positions becoming particles' best and/or neighborhood best positions. With this procedure, called *let particles fly*, infeasible particles may be drawn back to feasible search space by the influence of their personal and neighborhood bests.

3 PROPOSED PARTICLE SWARM OPTIMIZATION AND SUPPORT VECTOR MACHINE METHODOLOGY FOR RELIABILITY PREDICTION

In this work, PSO is considered to tackle the model selection problem for SVR tasks. As the Gaussian RBF kernel is broadly used in reliability related SVM works, it is the one taken into account. In this way, the parameters C, ε and the Gaussian RBF width σ must be defined. These three SVR parameters become decision variables to the PSO algorithm and they form a 3-dimensional search space. In fact, instead of using σ, it is considered $\gamma = \frac{0.5}{\sigma^2}$, which can be noticed on the expression of the RBF kernel in Table 2.1. This is necessary due to LIBSVM requirements, since it works with gamma values and not directly with the width σ. Thus, the i^{th} particle is described by the vectors $\mathbf{x}_i = (x_{i1}, x_{i2}, x_{i3})$, $\mathbf{p}_i = (p_{i1}, p_{i2}, p_{i3})$ and $\mathbf{v}_i = (v_{i1}, v_{i2}, v_{i3})$, where the first, second and third dimensions are respectively related to C, ε and γ.

Bratton & Kennedy (2007) contends that a standard PSO algorithm includes a *lbest* model, the usage of the constriction factor for velocities' and thus positions' updates, the number of particles set as 50, a non-uniform swarm initialization and the procedure of skipping fitness evaluation when particles exit the feasible search space.

In this work, similarly to the suggested PSO, it is used the constriction factor, infeasible particles are allowed but their fitness values are not assessed and a *lbest* model is implemented. Differently from the standard algorithm, the traditional random uniform swarm initialization is performed and the number of particles is set to 30. Indeed, Bratton & Kennedy (2007) states that 20-100 particles had produced quite similar empirical results.

Additionally, a *gbest* PSO is also implemented in order to have its performance compared to the *lbest* one in the specific context of reliability prediction. Also, in the PSO fitness evaluation phase, instead of only evaluating an ordinary objective function, the coupling with LIBSVM takes place. Next Section details the steps involved in the implemented PSO combined to LIBSVM.

3.1 Steps

3.1.1 Read of Data and Definition of Variables' Bounds

Before the initialization of PSO swarm, it is necessary to read the available data of inputs and outputs from a file in text format. Such file is organized as follows: the first column is comprised of the output values $(y_1, y_2, \ldots, y_{\ell+\vartheta+\lambda})$ and the following ones are each filled with a dimension of the input vector $\mathbf{x}$. This data set, for example, may have been originated from a condition monitoring procedure.

After reading all data, the entire set is subdivided in training, validation and test sets, whose

sizes (ℓ, ϑ, λ, respectively) are defined by the user. Usually, the majority of the entries are reserved for the training step, and the remaining ones are approximately equally divided to form the validation and test sets. The implemented PSO+SVM entails the validation sets approach in which the order of observations are respected, since cross-validation and leave-one-out are computational costly approaches and problems concerning reliability prediction based on time series are also considered.

The maximum and minimum values of the variables ($x_j^{min}, x_j^{max}, j = 1,2,3$) define intervals with different magnitudes. According to Kecman (2005), the "radius" of the SVR tube ε can be defined as a percentage of the mean of the training outputs ($y_i, i = 1,2,\ldots,\ell$). Following this idea, in this work, ε has its lower and upper bounds defined respectively as $\frac{0.001}{\ell}\sum_{i=1}^{\ell} y_i$ and $\frac{0.15}{\ell}\sum_{i=1}^{\ell} y_i$. This way of defining the boundary values of ε is adapted to the data under analysis. However, the minimum and maximum values of C and γ are not defined using the available observations and are determined rather in an arbitrary way. As a consequence, their ranges are greater than the one defined for ε.

3.1.2 Particle Swarm Initialization

In this work, it is implemented the traditional random uniform swarm initialization and the particles' initial positions are randomly selected from their respective intervals of definition. The positions $\mathbf{p}_i$ are initially set equal to $\mathbf{x}_i$ for each particle i.

Given that the variables' ranges are very different, the velocities are initialized in a special manner. The maximum velocity of each dimension v_j^{max} is set to 10% of the range where the specific variable is defined:

$$v_j^{max} = \frac{1}{10}(x_j^{max} - x_j^{min}), \quad j = 1,2,3 \tag{3.1}$$

in which x_j^{max} and x_j^{min} are the maximum and minimum values the related variable can assume. After that, velocities are randomly chosen from the interval $[-v_j^{max}, v_j^{max}]$, $j = 1,2,3$, for all particles. Only 10% of the range was chosen to initially set small velocities in an attempt to avoid the exit of particles to infeasible areas in early stages of the PSO algorithm.

The initialization procedure for an arbitrary particle i is summarized in the following pseudocode. The notation n indicates the number of dimensions or variables taken into account. For the SVR model selection problem considering RBF kernels, $n = 3$. The notation $\textsc{rand}(\cdot)$, in turn, is the function that returns a real number randomly selected in the interval passed as argument, according to a uniform distribution.

procedure INITIALIZEPARTICLE $(x_1^{min}, x_1^{max}, x_2^{min}, x_2^{max}, \ldots, x_n^{min}, x_n^{max})$
  for $j = 1, 2, \ldots, n$ **do**
    $x_{ij} \leftarrow$ RAND (x_j^{min}, x_j^{max})
    $p_{ij} \leftarrow x_{ij}$
    $v_{ij} \leftarrow$ RAND $(-v_j^{max}, v_j^{max})$ ▷ v_j^{max} defined by Equation (3.1)
  end for
  return particle i
end procedure

3.1.3 Definition of Particles' Neighborhoods

This step is required only when the *lbest* model is adopted. If the *gbest* one is considered, the neighborhood is equal to the entire swarm and there is no necessity to explicitly define particles' neighbors.

In the *lbest* approach, particles' neighbors are arbitrarily defined considering the particles' generation order and not taking into account any sort of distance metrics. For example, in the case of *lbest* ring model, particle i has $i-1$ and $i+1$ as neighbors. If $i = 1$, then the "left" neighbor is the last particle and, conversely, if the last particle is considered, its "right" neighbor is the first particle. For the sake of illustration, consider a swarm with 10 particles. Table 3.1 presents the neighborhood of each one of them when it is formed by 2 or 4 other particles. The list of particles concerns the order of generation in the initialization step.

Table 3.1: Examples of particles' neighborhoods

Particle	**2 neighbors**	**4 neighbors**
1	10, 2	9, 10, 2, 3
2	1, 3	10, 1, 3, 4
3	2, 4	1, 2, 4, 5
4	3, 5	2, 3, 5, 6
5	4, 6	3, 4, 6, 7
6	5, 7	4, 5, 7, 8
7	6, 8	5, 6, 8, 9
8	7, 9	6, 7, 9, 10
9	8, 10	7, 8, 10, 1
10	9, 1	8, 9, 1, 2

3.1.4 Fitness Evaluation: Coupling of Particle Swarm Optimization and Support Vector Machine

The objective function denoting the fitness of particles, in this work, is the NRMSE (2.84). At the fitness evaluation step, the coupling between PSO and SVM takes place. The validation

sets approach is adopted so as to guide the search of optimal parameter values by PSO. In this way, given a specific particle, whose current position ($\mathbf{x}$) defines a set of parameters, C, ε, γ, along with the training and validation sets at hand, LIBSVM is able to perform the SVR. Firstly, it solves the training problem taking into account the training set. It then provides the support vectors (both bounded and free), their respective Lagrange multipliers values as well as the value of the linear coefficient b_0. With these results it is possible to calculate the regression function. Secondly, these outcomes are used to feed the prediction portion of LIBSVM. Thus, the trained "machine" is used to predict the outputs from the input values of the validation set. With the predicted and the available real values it is possible to calculate the validation NRMSE.

Also, the fitness evaluation phase entails the update of particles' best positions ($\mathbf{p}$). If a particle's current position $\mathbf{x}$ results in a smaller validation NRMSE, then its best position becomes $\mathbf{x}$ and the calculated fitness value is stored. That is, $\mathbf{p}$ is made equal to $\mathbf{x}$ and a particle fitness is always related to its best position $\mathbf{p}$. Otherwise, nothing changes.

The fitness evaluation of all particles takes place immediately after the initialization and definition of neighborhoods (if *lbest* is adopted) steps so as to start the search procedure by PSO. It is also performed after the update of particles' velocities and positions and is followed by the update of the global best particle and the particles' best neighbors (when *lbest* model is considered).

3.1.5 Update of Particles' Velocities and Positions

Similarly to the standard algorithm, the developed PSO use the constriction factor in its updating rules as well as the permission of infeasible particles existence without fitness evaluation in such circumstance. In fact, the impacts of the *let particles fly* strategy could be empirically observed by some experiments performed in early stages of this work. When the particles exited the feasible search space in terms of one or more variables, their positions were set as either the lower or upper bound of the related dimensions, depending on the limit they have surpassed. This procedure negatively influenced the PSO performance, given that the particles were inclined to occupy the boundary regions of the feasible search space, which prejudiced its exploration. The implementation of *let particles fly* gave noticeable improvements to the PSO performance.

The update of particles' velocities and positions is accomplished by computing Equations (2.90) and (2.88), respectively. Additionally, along with the constriction factor, velocities are bounded to the definition ranges of variables so as to avoid particles going to far from the feasible search space. Hence, after initialization step, $v_j^{max} = x_j^{max} - x_j^{min}, j = 1,2,3$.

3.1.6 Update of Global Best and Particles' Best Neighbors

The update of global best and particles' best neighbors always takes place after the fitness evaluation step, given that it is based on particles' fitness values. The global best updating consists in identifying and storing the particle that have led to the smallest validation NRMSE overall particles of the swarm up to the present iteration. Not only the global best particle with its associated positions and fitness is stored, but also the number of the iteration in which it was found ($bIter$). This number may be used in the assessment of the stop criteria.

If the *lbest* model is adopted, then it is necessary to verify the best particles' neighbors at each PSO iteration, since they play the role of guiding search along with particles' best positions. In a particle's neighborhood, including itself, the one with the smallest validation NRMSE becomes the best neighbor in the current iteration.

3.1.7 Stop Criteria

These steps are repeated until a stop criterion is reached. The proposed PSO involves three of them:

1. Maximum number of iterations ($nIter$).
2. The global best particle (and thus the fitness value) is the same in 10% of the maximum number of iterations.
3. The global best fitness value in consecutive iterations are different but such difference is less than a tolerance δ.

The update of particles' velocities and positions, the fitness evaluation and the update of global best and best particles' neighbors are repeated until one of the considered stop criteria is met. As a result, the PSO+SVM procedure provides the "machine" with the most suitable parameter values C, ε, γ. This PSO-optimized SVM is then used to predict the outputs from the input values of the test set so as to have at least an idea of its generalization ability, which is given by the test NRMSE. Such evaluation, as in the fitness assessment step, is made by means of the prediction portion of LIBSVM.

In order to summarize and to provide the reader with the essence of what has been just explained the PSO+SVM algorithm is given in the forms of pseudocode and flow chart (Figure 3.1).

3.2 Proposed Methodology Pseudocode and Flow Chart

In the pseudocode, *nPart* and *nNeigh* are respectively the number of particles and of particles' neighbors. Besides that, svm is a "machine" returned from LIBSVM, f_i is the the fitness

value (validation NRMSE) of particle i and f_{test} is the test NRMSE returned by the best particle found in all PSO iterations. The symbol * means optimal in relation to the validation NRMSE.

procedure PARTICLESWARMOPTIMIZATION $(n, \mathbf{x}^{min}, \mathbf{x}^{max}, nPart, nNeigh, c_1, c_2, \chi, nIter, \delta, D, \ell, \vartheta, \lambda)$
- ▷ read data from a text file
- ▷ define variables' bounds
- ▷ particle swarm initialization and first fitness evaluation
- **for** $i = 1, \ldots, nPart$ **do**
  - INITIALIZEPARTICLE$(\mathbf{x}^{min}, \mathbf{x}^{max})$
  - svm $\leftarrow$ trainLIBSVM $(\mathbf{p}_i, D_1, D_2, \ldots, D_\ell)$ ▷ train
  - $(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_\vartheta) \leftarrow$ predLIBSVM(svm, $D_{\ell+1}, D_{\ell+2}, \ldots, D_{\ell+\vartheta}$) ▷ predict validation outputs
  - $f_i \leftarrow$ NRMSE $(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_\vartheta)$ ▷ particle fitness, Equation (2.84)
- **end for**
- ▷ find best global particle
- $f^* \leftarrow \min_i(f_i), i = 1, 2, \ldots, nPart$; $b \leftarrow i$ ▷ update best global fitness
- $\mathbf{p}^* \leftarrow \mathbf{p}_b$ ▷ update best global positon
- $bIter \leftarrow 0$ ▷ update best iteration
- ▷ define particles' neighborhood and best neihgbor
- ▷ perform PSO
- **for** $k = 1, 2, \ldots, nIter$ **do**
  - **for** $i = 1, 2, \ldots, nPart$ **do**
    - $\mathbf{v}_i \leftarrow \min($Equation (2.90)$, \mathbf{x}^{max} - \mathbf{x}^{min})$ ▷ update particle velocity
    - $\mathbf{x}_i \leftarrow$ Equation (2.88) ▷ update particle current position
    - **if** $\mathbf{x}_i$ is feasible **then**
      - ▷ evaluate fitness
      - svm $\leftarrow$ trainLIBSVM $(\mathbf{x}_i, D_1, D_2, \ldots, D_\ell)$
      - $(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_\vartheta) \leftarrow$ predLIBSVM(svm, $D_{\ell+1}, D_{\ell+2}, \ldots, D_{\ell+\vartheta}$)
      - $f \leftarrow$ NRMSE $(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_\vartheta)$
      - **if** $f < f_i$ **then**
        - $f_i \leftarrow f$ ▷ update particle fitness
        - $\mathbf{p}_i \leftarrow \mathbf{x}_i$ ▷ update particle best position
  - **end for**
  - ▷ update best global particle $(f^*, \mathbf{p}^*, bIter)$
  - ▷ update best particles' neighbors
  - ▷ verify whether stop criterion 2 or 3 is met
  - **if** $0 < |f_k^* - f_{k-1}^*| < \delta$ **or** $(k - bIter = 10\% \cdot nIter$ **and** $f_k^* = f_{bIter}^*)$ **then**
    - **break**
- **end for**
- $(\hat{y}_{\ell+\vartheta+1}, \hat{y}_{\ell+\vartheta+2}, \ldots, \hat{y}_{\ell+\vartheta+\lambda}) \leftarrow$ predLIBSVM (svm*, $D_{\ell+\vartheta+1}, D_{\ell+\vartheta+2}, \ldots, D_{\ell+\vartheta+\lambda}$) ▷ predict test outputs
- $f_{test} \leftarrow$ NRMSE $(\hat{y}_{\ell+\vartheta+1}, \hat{y}_{\ell+\vartheta+2}, \ldots, \hat{y}_{\ell+\vartheta+\lambda})$ ▷ Equation (2.84)
- **return** $f^*, \mathbf{p}^*, f_{test}$

end procedure ▷ end of procedure

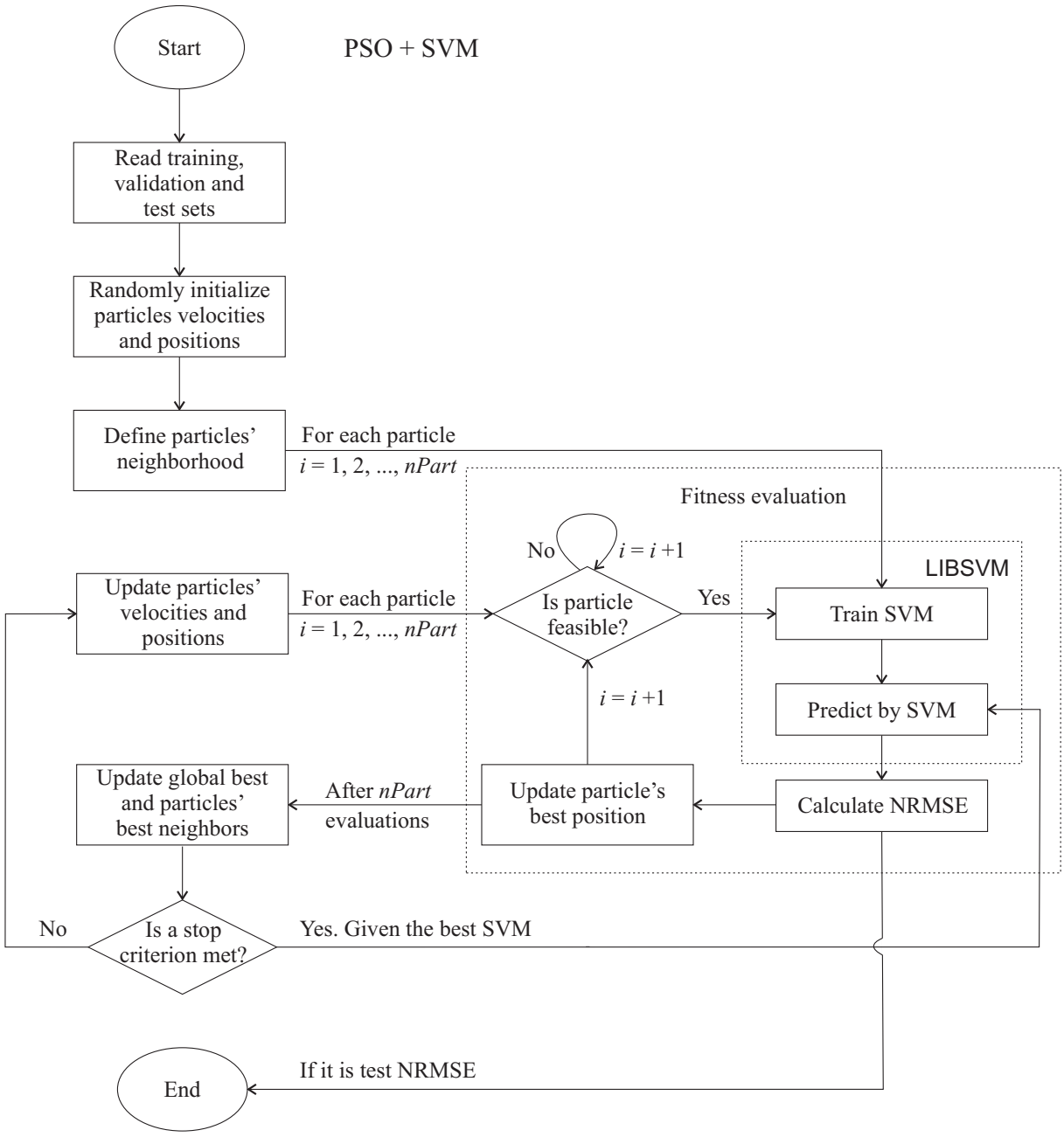

Figure 3.1: Flow chart of the proposed PSO+SVM methodology

4 RELIABILITY PREDICTION BY PARTICLE SWARM OPTIMIZATION AND SUPPORT VECTOR MACHINES

This chapter presents three problems from literature related to reliability prediction based on time series data and one application example with data collected from oil production wells. They are solved by means of the PSO+SVM algorithm described in Chapter 3. As recommended by Bratton & Kennedy (2007), the *lbest* model is adopted. The first two literature examples are related to the forecast of failure times of engineered components and the third one concerns the prediction of miles to failure of a car engine. It is worth to forecast these aspects associated with component failures so as to capture non-linear trends they may present and thus provide support to reliability-based maintenance decisions.

Xu *et al.* (2003) contends that in practice the short-term (single-step) forecasts are more useful since they provide timely information for preventive and corrective maintenance plans, even though the multi-step predictions may catch some of system dynamics. In this way, single-step-ahead forecasts are taken into account. In addition, one-dimensional input vectors are considered, that is, $p = 1$ and $\mathbf{x}_i = x_i = y_{i-1}$ and consequently the data sets are reduced by one entry.

The last example is related to the prediction of TBF of oil wells by means of different characteristics of the system, such as the number of installed rods. This example illustrates the application of the proposed methodology in a real situation. Also, it entails numerical as well as categorical variables, which may be handled in different manners before they are used by the PSO+SVM. In addition, differently from the time series based examples, the last one involves a multi-dimensional input vector.

Given that PSO is a stochastic tool, 30 runs of the algorithm are performed in order to analyze its behavior. Although the NRMSE was the only error function which guided the search for parameters by PSO, the MAPE and the MSE related to particles were also computed. Additionally, in the forthcoming Sections 4.1, 4.2, 4.3 and 4.4, apart from the Tables regarding descriptive statistics, all other Tables along with the Figures are associated with the *lbest* PSO+SVM run that resulted in the smallest test NRMSE value. Even though such "machines" may not give the most suitable validation NRMSE, they show the best generalization performance. For comparison purposes, all examples were also solved by means of a *gbest* PSO model combined with SVM (Section 4.5.1).

The PSO algorithm was implemented in MATLAB 7.8 and linked with LIBSVM. All experiments were run in a computer with 2GHz, 2.9Gb of RAM and Linux Ubuntu 9.04 operational system.

4.1 Example 1: Failure Times of a Submarine Diesel Engine

In this first example, the time series regards the times of unscheduled maintenance actions for a submarine diesel engine under deterioration process and is extracted from Ascher & Feingold (1984). The data is presented in Table 4.1. Since a single-step ahead forecast with one-dimensional input vectors is performed, the data set is reduced from 71 to 70 in order to elaborate a time series in the same reasoning described in Section 2.1.5.3. Then the first 44 data points are used for training, the following 12 for validation and the last 14 for test purposes.

Table 4.1: Engine age ($\times$ 1000 hours) at time of unscheduled maintenance actions. Adapted from Ascher & Feingold (1984), p. 75

Action	Age	Action	Age	Action	Age	Action	Age	Action	Age
1	1.382	16	17.632	30	21.061	44	21.888	58	23.491
2	2.990	17	18.122	31	21.309	45	21.930	59	23.526
3	4.124	18	19.067	32	21.310	46	21.943	60	23.774
4	6.827	19	19.172	33	21.378	47	21.946	61	23.791
5	7.472	20	19.299	34	21.391	48	22.181	62	23.822
6	7.567	21	19.360	35	21.456	49	22.311	63	24.006
7	8.845	22	19.686	36	21.461	50	22.634	64	24.286
8	9.450	23	19.940	37	21.603	51	22.635	65	25.000
9	9.794	24	19.944	38	21.658	52	22.669	66	25.010
10	10.848	25	20.121	39	21.688	53	22.691	67	25.048
11	11.993	26	20.132	40	21.750	54	22.846	68	25.268
12	12.300	27	20.431	41	21.815	55	22.947	69	25.400
13	15.413	28	20.525	42	21.820	56	23.149	70	25.500
14	16.497	29	21.057	43	21.822	57	23.305	71	25.518
15	17.352								

The *lbest* model involves 4 neighbors and the underlying swarm communication network can be visualized in the middle graph of Figure 2.9. The PSO required parameters are listed in Table 4.2 and are also valid for the subsequent examples.

Table 4.2: PSO required parameters

Parameter	Value
Number of particles	30
Number of neighbors	4
$c_1 = c_2$	2.05
Constriction factor (χ)	$7.2984 \cdot 10^{-1}$
Maximum number of iterations	6000
Maximum number of iterations with equal best fitness value	600
Tolerance (δ)	$1 \cdot 10^{-12}$

In addition, the definition intervals of the PSO variables (C, ε, γ) as well as the initial values of $v_j^{max}, j = 1,2,3$ (*i.e.* 10% of variables ranges) are shown in Table 4.3. Notice that after the

swarm initialization, these maximum velocities become equal to the definition range of each variable. Additionally, ε interval is determined as described in Chapter 3, that is, from 0.1% to 15% of the mean of training outputs.

Table 4.3: PSO variables' intervals and initial maximum velocities, Example 1

Variable	**Interval**	**Initial v_j^{max}**
C	$[1 \cdot 10^{-1}, 2000]$	199.9900
ε	$[1.7302 \cdot 10^{-2}, 2.5953]$	$2.5780 \cdot 10^{-1}$
γ	$[1 \cdot 10^{-6}, 50]$	4.9999

The descriptive statistics of the obtained results from the 30 PSO+SVM runs are in Table 4.4. The parameters related to the machine which provided the smallest test NRMSE are $C = 263.6966$, $\varepsilon = 1.3701 \cdot 10^{-1}$ and $\gamma = 6.7315 \cdot 10^{-3}$.

Table 4.4: Descriptive statistics of parameters and error functions, stop criteria frequency and performance for 30 PSO+SVM runs, *lbest*, Example 1

		Minimum	**Maximum**	**Median**	**Mean**	**Std. dev.***
Parameters	C	190.4848	1930.2743	732.5368	822.3439	560.3221
	ε	$3.4194 \cdot 10^{-2}$	$2.3899 \cdot 10^{-1}$	$1.9672 \cdot 10^{-1}$	$1.5857 \cdot 10^{-1}$	$6.0396 \cdot 10^{-2}$
	γ	$4.4980 \cdot 10^{-3}$	49.9974	$6.6326 \cdot 10^{-3}$	1.6729	9.1270
Validation error	NRMSE	$4.4060 \cdot 10^{-3}$	$2.9305 \cdot 10^{-1}$	$4.4123 \cdot 10^{-3}$	$1.4035 \cdot 10^{-2}$	$5.2700 \cdot 10^{-2}$
	MAPE (%)	$3.7297 \cdot 10^{-1}$	26.2757	$3.7501 \cdot 10^{-1}$	1.2384	4.7288
	MSE	$9.9229 \cdot 10^{-3}$	43.8973	$9.9515 \cdot 10^{-3}$	1.6979	8.6070
Test error	NRMSE	$7.1126 \cdot 10^{-3}$	$3.8265 \cdot 10^{-1}$	$7.4624 \cdot 10^{-3}$	$2.0307 \cdot 10^{-2}$	$6.8440 \cdot 10^{-2}$
	MAPE (%)	$4.7571 \cdot 10^{-1}$	38.1006	$5.2994 \cdot 10^{-1}$	1.8181	6.8533
	MSE	$3.0472 \cdot 10^{-2}$	88.1988	$3.6380 \cdot 10^{-2}$	3.4281	17.2899
				Absolute (relative, %) frequency		
Stop criteria	Maximum number of iterations (6000)			13 (43.3333)		
	Equal best fitness for 600 iterations			13 (43.3333)		
	Tolerance $\delta = 1 \cdot 10^{-12}$			4 (13.3334)		
				Metric value		
Performance	Mean time per run (minutes)			10.8342		
	Mean number of trainings			116016.1667		
	Mean number of predictions			116016.1667		

*Standard deviation

Figure 4.1 provides a snapshot of the swarm evolution during the optimization process. Notice that at the 50$^{\text{th}}$ iteration there are particles outside the feasible search space, but the final swarm is comprised only by feasible particles. Observe also that given the influence of the *lbest* model and of the validation NRMSE, particles were inclined to form clusters in the search space. Figure 4.2, in turn, shows the NRMSE convergence along PSO. This specific PSO run stopped at iteration 703, in which the tolerance $\delta = 1 \cdot 10^{-12}$ was attained.

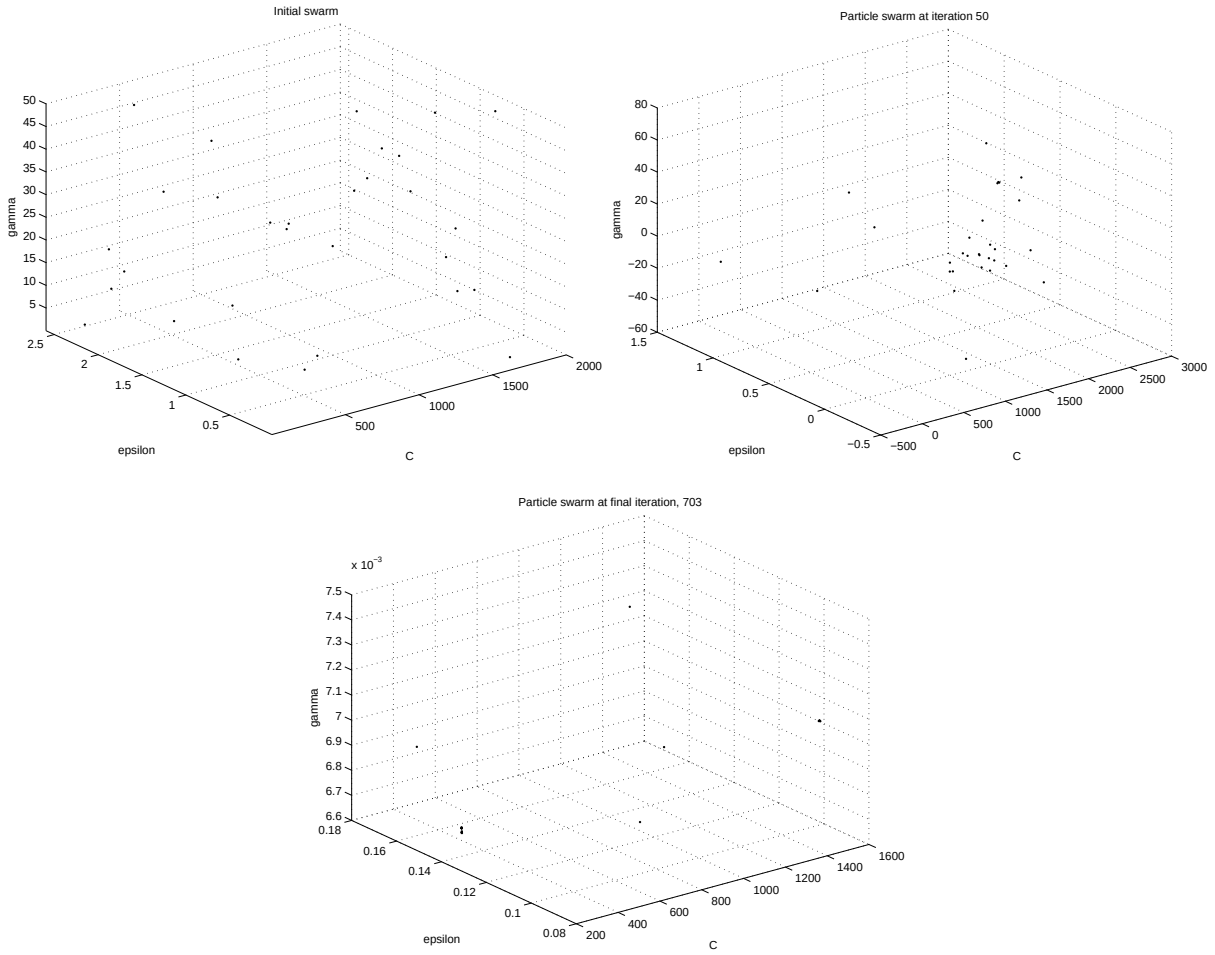

Figure 4.1: Swarm evolution during PSO, Example 1

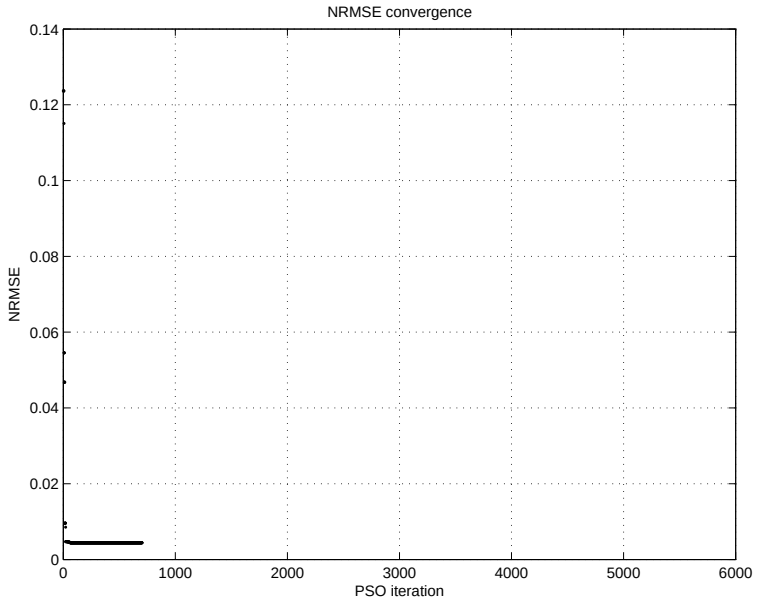

Figure 4.2: Validation NRMSE convergence, Example 1

Table 4.5 presents the real output values as well as validation and prediction results from the selected SVR. Table 4.6, in turn, shows the support vectors, their respective Lagrange multipliers values and the classification as free or bounded support vector. Notice that only 22 from the 44 training points were chosen to be support vectors and from these, only 5 are free. Substituting the values from Table 4.6 in equations (2.79) and (2.82), one may obtain the linear coefficient b_0 and the regression function $f(\mathbf{x})$, in this order. The real outputs, the validation and test prediction values as well as the support vectors are depicted in Figure 4.3.

Table 4.5: Real failure time (engine age) and predictions by SVR, Example 1

Validation			**Test**		
Action	**Real age**	**Predicted age**	**Action**	**Real age**	**Predicted age**
46	21.9430	22.0328	58	23.4910	23.4414
47	21.9460	22.0459	59	23.5260	23.6327
48	22.1810	22.0489	60	23.7740	23.6686
49	22.3110	22.2870	61	23.7910	23.9224
50	22.6340	22.4195	62	23.8220	23.9398
51	22.6350	22.7503	63	24.0060	23.9714
52	22.6690	22.7513	64	24.2860	24.1582
53	22.6910	22.7863	65	25.0000	24.4398
54	22.8460	22.8089	66	25.0100	25.1356
55	22.9470	22.9684	67	25.0480	25.1450
56	23.1490	23.0725	68	25.2680	25.1809
57	23.3050	23.2807	69	25.4000	25.3857
			70	25.5000	25.5062
			71	25.5180	25.5962
NRMSE		$4.4142 \cdot 10^{-3}$	NRMSE		$7.1126 \cdot 10^{-3}$
MAPE(%)		$3.7486 \cdot 10^{-1}$	MAPE (%)		$4.7767 \cdot 10^{-1}$
MSE		$9.9597 \cdot 10^{-3}$	MSE		$3.0472 \cdot 10^{-2}$

Table 4.6: Support vectors' details, Example 1

(x,y)	α	α^*	**Type**	(x,y)	α	α^*	**Type**
(1.382, 2.990)	0	135.2668	free	(16.497, 17.352)	122.0212	0	free
(4.124, 6.827)	263.6966	0	bounded	(17.352, 17.632)	0	263.6966	bounded
(7.472, 7.567)	0	263.6966	bounded	(18.122, 19.067)	263.6966	0	bounded
(7.567 , 8.845)	263.6966	0	bounded	(19.067, 19.172)	0	263.6966	bounded
(8.845 , 9.450)	0	263.6966	bounded	(19.172, 19.299)	0	44.2857	free
(9.450 , 9.794)	0	263.6966	bounded	(19.299, 19.360)	0	263.6966	bounded
(9.794 , 10.848)	263.6966	0	bounded	(19.940, 19.944)	0	263.6966	bounded
(10.848, 11.993)	57.7336	0	free	(20.121, 20.132)	0	263.6966	bounded
(11.993, 12.300)	0	263.6966	bounded	(20.132, 20.431)	263.4942	0	free
(12.300, 15.413)	263.6966	0	bounded	(20.525, 21.057)	263.6966	0	bounded
(15.413, 16.497)	263.6966	0	bounded	(21.061, 21.309)	263.6966	0	bounded

Indeed, Hong & Pai (2006) solve this same example application by means of SVM coupled with a method to find the parameters C, ε, σ as well as with other forecast tools, namely the

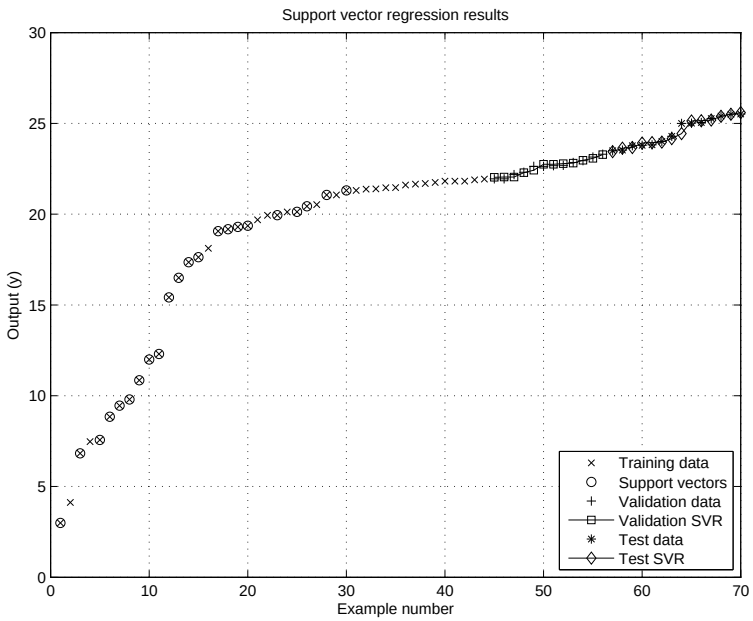

Figure 4.3: SVR results, Example 1

Duane model, the ARIMA and the General Regression Neural Network (GRNN). The search method for SVR parameters entails the following idea: (i) fix the values of two parameters (C, ε) and find the optimal value of the remaining one (σ_0); (ii) given σ_0 and ε, obtain the optimal value C_0; (iii) find ε_0 based on σ_0 and C_0. This procedure is guided by the evaluation of NRMSE and the authors consider 45, 12 and 14 points for training, validation and test, respectively.

Nevertheless, the optimal parameters' values presented along with the information provided are not sufficient to reproduce the NRMSE value of $6.4500 \cdot 10^{-3}$ reported. This data set with the mentioned division as well as the presented optimal parameters were used to train and predict the validation and test outputs by means of LIBSVM, but the NRMSE found was much greater than $6.4500 \cdot 10^{-3}$ and also the tendency of both validation and test predictions was the opposite of the real one. Despite that, the test NRMSE results from Hong & Pai (2006) are presented in Table 4.7 with the additional entry corresponding to the best test NRMSE obtained by the PSO+SVM approach from this work. It can be observed that it is competitive with all other values provided by the different tools.

Table 4.7: Test NRMSE from different forecast models, Example 1. Adapted from Hong & Pai (2006), p. 160

Method	**Test NRMSE**
PSO+SVM	$7.1126 \cdot 10^{-3}$
SVM	$6.4500 \cdot 10^{-3}$
Duane	$1.0590 \cdot 10^{-2}$
GRNN	$9.7300 \cdot 10^{-3}$
ARIMA	$3.3660 \cdot 10^{-2}$

4.2 Example 2: Turbochargers in Diesel Engines

The second application example is extracted from Xu *et al.* (2003) and is related to turbochargers in diesel engines. As stated by the authors, the reliability is the one of the most important considerations for diesel engine systems. In this way, an accurate prediction of its reliability provides a good assessment of its performance. Table 4.8 presents the failure times of 40 turbochargers of the same type as well as the non-parametric estimation of their reliabilities calculated by:

$$R(T_i) = 1 - \frac{i - 0.3}{n + 0.4} \tag{4.1}$$

where i is the failure index and n is the data sample size.

Table 4.8: Turbochargers failure times ($\times$ 1000 hours) and reliability data. Adapted from Xu *et al.* (2003), p. 259

i	T_i	$R(T_i)$	i	T_i	$R(T_i)$	i	T_i	$R(T_i)$	i	T_i	$R(T_i)$
1	1.6	0.9930	11	5.1	0.8934	21	6.5	0.7938	31	7.9	0.6942
2	2.0	0.9831	12	5.3	0.8835	22	6.7	0.7839	32	8.0	0.6843
3	2.6	0.9731	13	5.4	0.8735	23	7.0	0.7739	33	8.1	0.6743
4	3.0	0.9631	14	5.6	0.8635	24	7.1	0.7639	34	8.3	0.6643
5	3.5	0.9532	15	5.8	0.8536	25	7.3	0.7540	35	8.4	0.6544
6	3.9	0.9432	16	6.0	0.8436	26	7.3	0.7440	36	8.4	0.6444
7	4.5	0.9333	17	6.0	0.8337	27	7.3	0.7341	37	8.5	0.6345
8	4.6	0.9233	18	6.1	0.8237	28	7.7	0.7241	38	8.7	0.6245
9	4.8	0.9133	19	6.3	0.8137	29	7.7	0.7141	39	8.8	0.6145
10	5.0	0.9034	20	6.5	0.8038	30	7.8	0.7042	40	9.0	0.6046

Xu *et al.* (2003) made two experiments with these data, both regarding reliability forecasts. In a first situation they considered previous reliability values and also the failure times as input data. The second experiment, in turn, had their inputs comprised only by past reliability values. The authors used several sorts of NN as forecast tool and observed that better results were obtained in the latter experiment. Additionally, they compared various types of NN (Multilayer Perceptron Neural Network (MLP-NN) with logistic and Gaussian activations and Radial Basis Function Neural Network (RBF-NN) with Gaussian activation) and ARIMA performances.

Also, Chen (2007) makes use of the same data set to train a SVM and then predict reliability using it. The authors consider 4-dimensional input vectors $\mathbf{x}$ (*i.e.* $p = 4$) and the SVR parameters are obtained by GA. Also, despite the time series characteristics, they adopt a cross-validation technique embedded in GA so as to guide the search for optimal C, ε, σ. Once this set is found, they retrain all training data and then perform prediction tasks on the test set. The author compared GA+SVM results with the ones obtained by GRNN, MLP-NN, RBF-NN, Neural Fuzzy Network (NFN) and ARIMA.

In this work, besides the turbocharger reliability prediction it is also performed the forecast of its failure times. The input vectors for the reliability experiment are made only by a single

past reliability value and do not consider failure times at all. For failure times case, similarly, input vectors are one-dimensional and formed by the immediately previous failure time.

4.2.1 Example 2.1: Reliability Forecast

Actually, this example is the simplest among the ones presented in this work, given that the reliability values does not depend on previous values of reliability or on failure times, but only on the failure index (see Equation (4.1)). In this way, the time series pairs $(y_{i-1}, y_i), i = 1, 2, \cdots n$ form a straight line. One may question why such an "easy" problem may need sophisticated tools like NN, ARIMA and SVM to be solved. However, this particular example is widely used in literature works, as can be inferred from the comments in the beginning of this Section, maybe to show the effectiveness of these methods in also tackling simple problems.

For this case the first 30 points are used for SVR training, the following 4 entries guide the search for optimal parameters C, ε and γ by PSO and the last 5 examples form the test set. The only difference in parameters' intervals shown in Table 4.3 regards the lower and upper bounds of ε, whose definition depends on the data at hand. For the reliability prediction case, they are respectively $8.5358 \cdot 10^{-4}$ and $1.2804 \cdot 10^{-1}$.

The descriptive statistics of the model selection results for the 30 PSO+SVM runs are presented in Table 4.9. Observe that due to problem simplicity, standard deviations of all parameters and errors were small if compared to the ones from the other examples. Additionally, none of the runs made use of the maximum number of iterations as stop criterion and the mean elapsed time was very low, only about one minute per run.

The parameter values associated with the PSO+SVM run that resulted in the smallest test NRMSE are $C = 1923.6203, \varepsilon = 8.3873 \cdot 10^{-4}, \gamma = 8.9796 \cdot 10^{-4}$. Figure 4.4 depicts the particle swarm in the initial, 50^{th} and 1822^{th} (final) iterations. At this particular run, the best global validation NRMSE was found in iteration 1222 and remained the same for the next 600 iterations. The validation NRMSE evolution can be seen in Figure 4.5.

The real and predicted values for validation and test outputs were very near from each other and are listed in Table 4.10. The support vectors features are shown in Table 4.11, from which can be noted that only two bounded support vectors were required by the SVR model. Figure 4.6 presents the two support vectors in addition to the validation and reliability forecasts. Interestingly, the chosen support vectors were the first and last training points.

Xu *et al.* (2003) and Chen (2007) present the test NRMSE values for this example from different time series models. In order to update the list of such values with the results of the proposed methodology in this work, Table 4.12 is provided. Notice that among all methods, PSO+SVM was able to give the smallest test NRMSE. This fact indicates the ability of PSO in handling the model selection problem related to SVR as well as the great capacity of SVR itself to tackle reliability forecast problems. It is important to emphasize that this better result was attained even with a smaller training set size. Xu *et al.* (2003) mention only the training and

Table 4.9: Descriptive statistics of parameters and error functions, stop criteria frequency and performance for 30 PSO+SVM runs, *lbest*, Example 2.1

		Minimum	**Maximum**	**Median**	**Mean**	**Std. dev.***
Parameters	C	1900.9063	1946.8917	1917.5483	1919.3760	10.8901
	ε	$8.3864 \cdot 10^{-4}$	$8.4078 \cdot 10^{-4}$	$8.3869 \cdot 10^{-4}$	$8.3906 \cdot 10^{-4}$	$6.7860 \cdot 10^{-7}$
	γ	$8.8721 \cdot 10^{-4}$	$9.0869 \cdot 10^{-4}$	$9.0078 \cdot 10^{-4}$	$9.0041 \cdot 10^{-4}$	$4.6488 \cdot 10^{-6}$
Validation error	NRMSE	$8.4451 \cdot 10^{-5}$	$4.1839 \cdot 10^{-4}$	$8.4487 \cdot 10^{-5}$	$1.0678 \cdot 10^{-4}$	$8.4700 \cdot 10^{-5}$
	MAPE (%)	$7.4023 \cdot 10^{-3}$	$4.1137 \cdot 10^{-2}$	$7.4037 \cdot 10^{-3}$	$9.6523 \cdot 10^{-3}$	$8.5579 \cdot 10^{-3}$
	MSE	$3.1960 \cdot 10^{-9}$	$7.8443 \cdot 10^{-8}$	$3.1987 \cdot 10^{-9}$	$8.2172 \cdot 10^{-9}$	$1.9088 \cdot 10^{-8}$
Test error	NRMSE	$2.0304 \cdot 10^{-4}$	$5.8311 \cdot 10^{-4}$	$2.0444 \cdot 10^{-4}$	$2.2999 \cdot 10^{-4}$	$9.5987 \cdot 10^{-5}$
	MAPE (%)	$1.8678 \cdot 10^{-2}$	$5.7884 \cdot 10^{-2}$	$1.8831 \cdot 10^{-2}$	$2.1465 \cdot 10^{-2}$	$9.8995 \cdot 10^{-3}$
	MSE	$1.6087 \cdot 10^{-8}$	$1.3267 \cdot 10^{-7}$	$1.6308 \cdot 10^{-8}$	$2.4114 \cdot 10^{-8}$	$2.9507 \cdot 10^{-8}$
				Absolute (relative, %) frequency		
Stop criteria	Maximum number of iterations (6000)			0 (0)		
	Equal best fitness for 600 iterations			25 (83.3333)		
	Tolerance $\delta = 1 \cdot 10^{-12}$			5 (16.6667)		
				Metric value		
Performance	Mean time per run (minutes)			1.0066		
	Mean number of trainings			67989.7000		
	Mean number of predictions			67989.7000		

*Standard deviation

Table 4.10: Real failure time and predictions by SVR, Example 2.1

Validation			**Test**		
i	$R(T_i)$	**Predicted** $R(T_i)$	i	$R(T_i)$	**Predicted** $R(T_i)$
32	0.6843	0.6842	36	0.6444	0.6445
33	0.6743	0.6743	37	0.6345	0.6345
34	0.6643	0.6644	38	0.6245	0.6247
35	0.6544	0.6544	39	0.6145	0.6147
			40	0.6046	0.6047
NRMSE		$8.4525 \cdot 10^{-5}$	NRMSE		$2.0305 \cdot 10^{-4}$
MAPE (%)		$7.4037 \cdot 10^{-3}$	MAPE (%)		$1.8678 \cdot 10^{-2}$
MSE		$3.2016 \cdot 10^{-9}$	MSE		$1.6087 \cdot 10^{-8}$

Table 4.11: Support vectors' details, Example 2.1

(x, y)	α	α^*	**Type**
(0.9930, 0.9831)	1923.6203	0	bounded
(0.7042, 0.6942)	0	1923.6203	bounded

validation sets. Thus, one may infer that the validation set actually played the role of the test set. Chen (2007), in turn, makes use of the cross-validation approach and after parameters have been found, all training set was retrained by SVR. Hence, in both cases, the training sets were

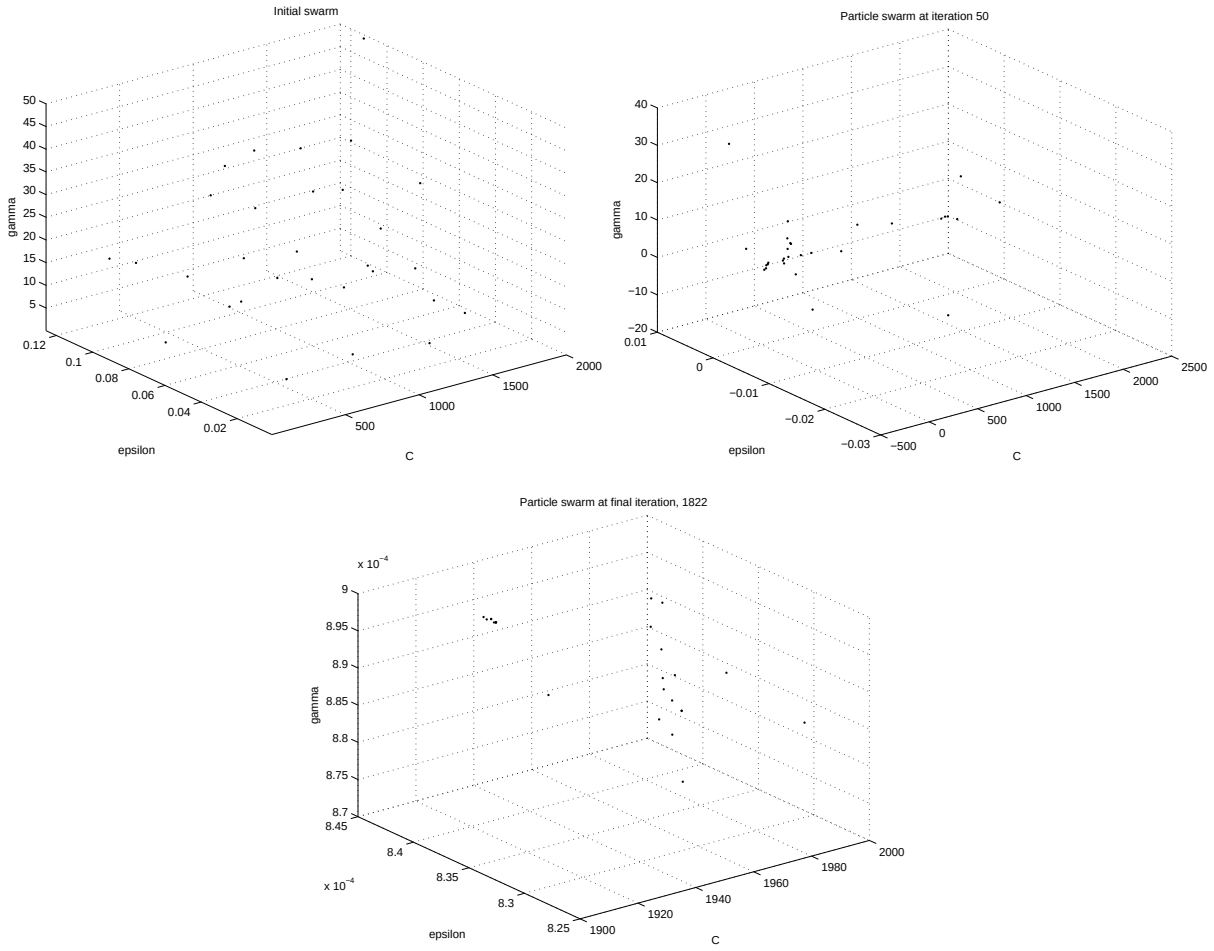

Figure 4.4: Swarm evolution during PSO, Example 2.1

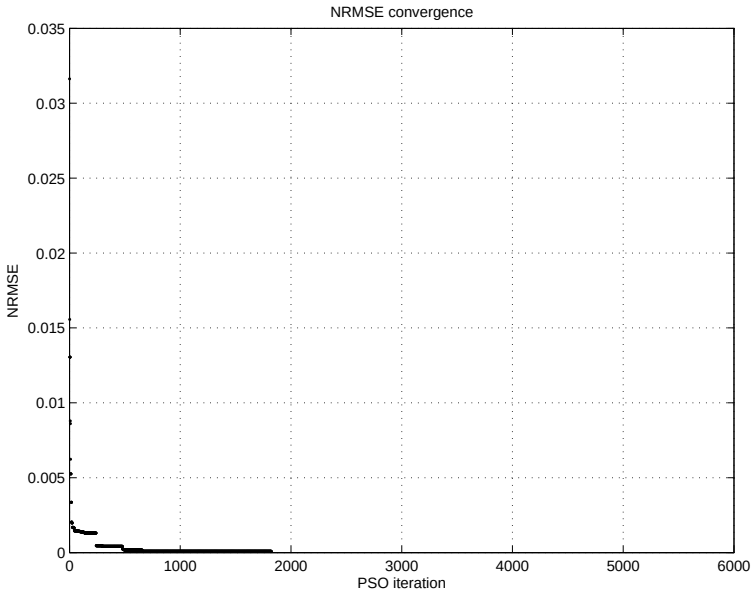

Figure 4.5: Validation NRMSE convergence, Example 2.1

indeed greater than the one used in this work.

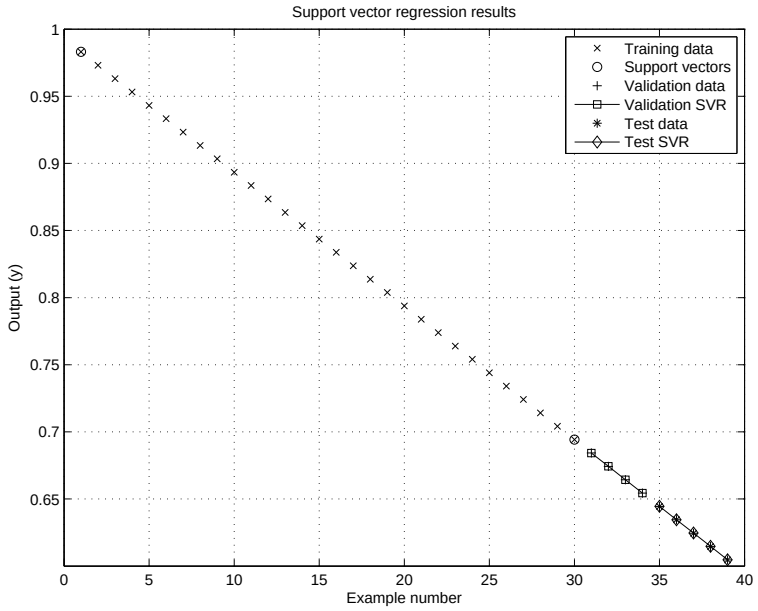

Figure 4.6: SVR results, Example 2.1

Table 4.12: Test NRMSE from different forecast models, Example 2.1. Updated from Xu *et al.* (2003), p. 260, and Chen (2007), p.430

Method	**Test NRMSE**
PSO+SVM	$2.0305 \cdot 10^{-4}$
GA+SVM	$4.0000 \cdot 10^{-4}$
NFN	$3.6900 \cdot 10^{-3}$
RBF-NN (Gaussian activation)	$3.9100 \cdot 10^{-3}$
MLP-NN (Gaussian activation)	$2.4970 \cdot 10^{-2}$
MLP-NN (logistic activation)	$3.9700 \cdot 10^{-2}$
GRNN	$1.0850 \cdot 10^{-2}$
ARIMA	$1.9900 \cdot 10^{-2}$

4.2.2 Example 2.2: Failure Times Forecast

Differently from the reliability prediction problem, in the failure times forecast the data set is divided approximately with the same proportions of the first example, that is, 63%, 17% and 20% from the data dedicated to training, validation and test purposes, in this order. This yields respectively 24, 7 and 8 points. Also, $\varepsilon \in [5.2750 \cdot 10^{-3}, 7.9125 \cdot 10^{-1}]$.

Descriptive statistics of parameters and error functions as well as some performance metrics for *lbest* model are listed on Table 4.13. Observe that in the majority of the runs, the best global fitness value remained the same in 600 consecutive iterations and the maximum number of iterations was not attained in none of them. This indicates that the PSO is able to find good solutions without requiring the maximum number of iterations, which positively influence the algorithm elapsed time.

The parameter values associated with the "machine" which provided the smallest *lbest* test

Table 4.13: Descriptive statistics of parameters and error functions, stop criteria frequency and performance for 30 PSO+SVM runs, *lbest*, Example 2.2

		Minimum	**Maximum**	**Median**	**Mean**	**Std. dev.***
Parameters	C	34.3824	1997.3652	1650.7974	1276.5337	778.9501
	ε	$1.1999 \cdot 10^{-2}$	$1.5487 \cdot 10^{-1}$	$5.2240 \cdot 10^{-2}$	$8.8812 \cdot 10^{-2}$	$6.3591 \cdot 10^{-2}$
	γ	$4.6374 \cdot 10^{-3}$	$5.2242 \cdot 10^{-1}$	$1.9170 \cdot 10^{-2}$	$1.5319 \cdot 10^{-1}$	$2.2412 \cdot 10^{-2}$
Validation error	NRMSE	$1.6620 \cdot 10^{-2}$	$1.6862 \cdot 10^{-2}$	$1.6823 \cdot 10^{-2}$	$1.6777 \cdot 10^{-2}$	$9.3827 \cdot 10^{-5}$
	MAPE (%)	1.2262	1.3132	1.2422	1.2539	$2.8847 \cdot 10^{-2}$
	MSE	$1.6274 \cdot 10^{-2}$	$1.6751 \cdot 10^{-2}$	$1.6674 \cdot 10^{-2}$	$1.6583 \cdot 10^{-2}$	$1.8497 \cdot 10^{-4}$
Test error	NRMSE	$1.3412 \cdot 10^{-2}$	$7.1785 \cdot 10^{-2}$	$1.7837 \cdot 10^{-2}$	$3.1674 \cdot 10^{-2}$	$2.4001 \cdot 10^{-2}$
	MAPE (%)	1.1299	5.0759	1.4722	2.4071	1.6025
	MSE	$1.3087 \cdot 10^{-2}$	$3.7489 \cdot 10^{-1}$	$2.3149 \cdot 10^{-2}$	$1.1353 \cdot 10^{-1}$	$1.5178 \cdot 10^{-1}$
				Absolute (relative, %) frequency		
Stop criteria	Maximum number of iterations (6000)			0 (0)		
	Equal best fitness for 600 iterations			29 (96.6667)		
	Tolerance $\delta = 1 \cdot 10^{-12}$			1 (3.3333)		
				Metric value		
Performance	Mean time per run (minutes)			10.5648		
	Mean number of trainings			68977.6667		
	Mean number of predictions			68977.6667		

*Standard deviation

NRMSE are $C = 1936.7744, \varepsilon = 1.5474 \cdot 10^{-1}, \gamma = 4.6374 \cdot 10^{-3}$ and that specific run stopped at iteration 2836 since the global best particle reached the associated best position 600 iterations earlier. Figure 4.7 show the evolution of the particle swarm in three different phases of the algorithm and Figure 4.8 depicts the NRMSE values *versus* the PSO iterations. Tables 4.14 and 4.15 present, respectively, the real and predicted failure times for validation and test sets and the support vectors' details. Notice that for this example the test errors are smaller than the validation ones and that only 6 from the 24 training examples are support vectors (50% of them are free). Lastly, the SVR results are summarized in Figure 4.9.

4.3 Example 3: Miles to Failure of a Car Engine

This example is associated with the prediction of Miles To Failure (MTF) of a car engine and it also comes from Xu *et al.* (2003), who collected data from 100 units of a specific car engine (Table 4.16).

The objective is to predict future MTF of car engines based on past failure evidence. Once more, it is performed a single-step-ahead forecast with one-dimensional input vectors, which results in a data set with 99 entries. From these 80, 9 and 10 points are used for training, validation and test purposes, respecting their natural order. In this example $\varepsilon \in [3.7590 \cdot 10^{-2}, 5.6385]$.

Table 4.17 presents the descriptive statistics related to parameters and error functions in 30 PSO+SVM runs. Only in this example the mean number of predictions were different from

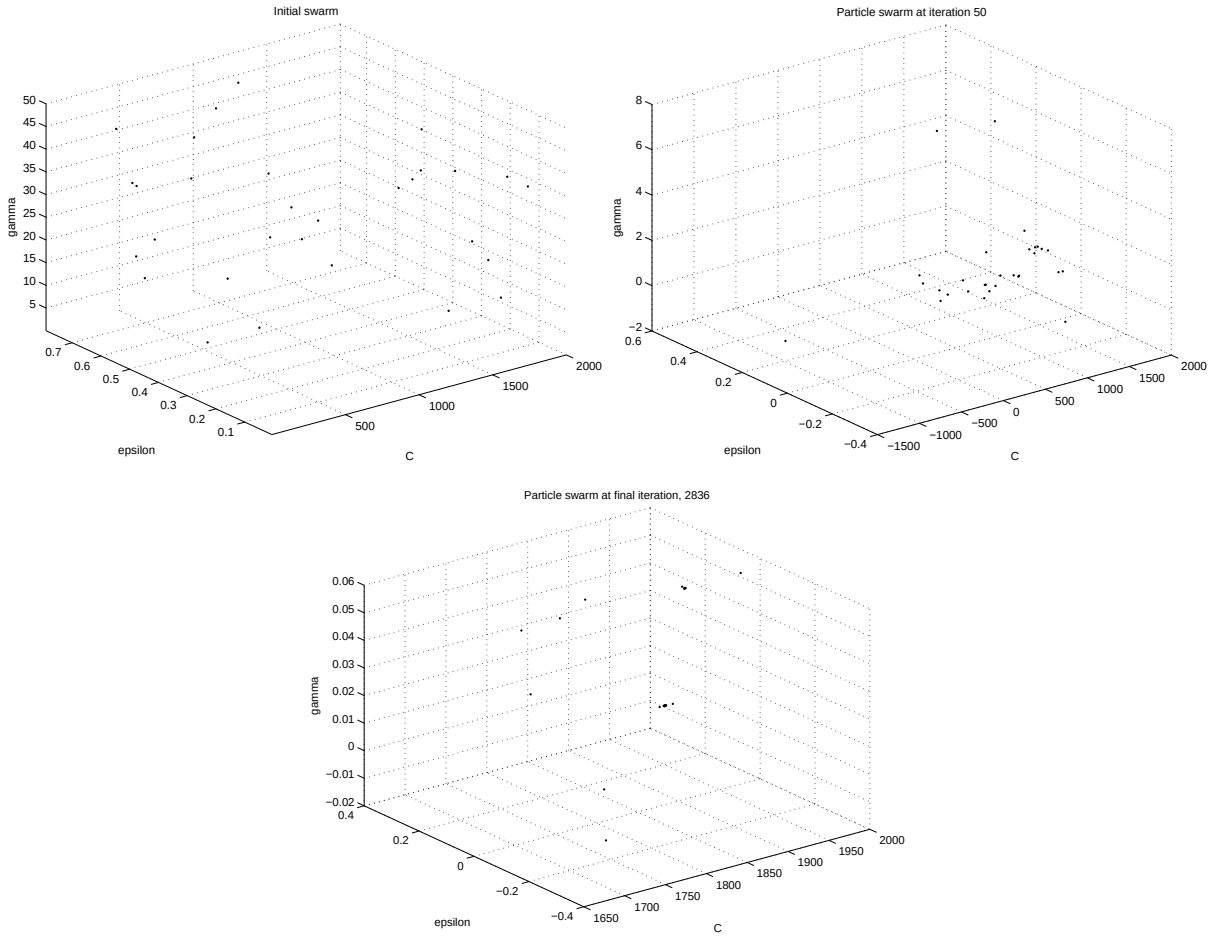

Figure 4.7: Swarm evolution during PSO, Example 2.2

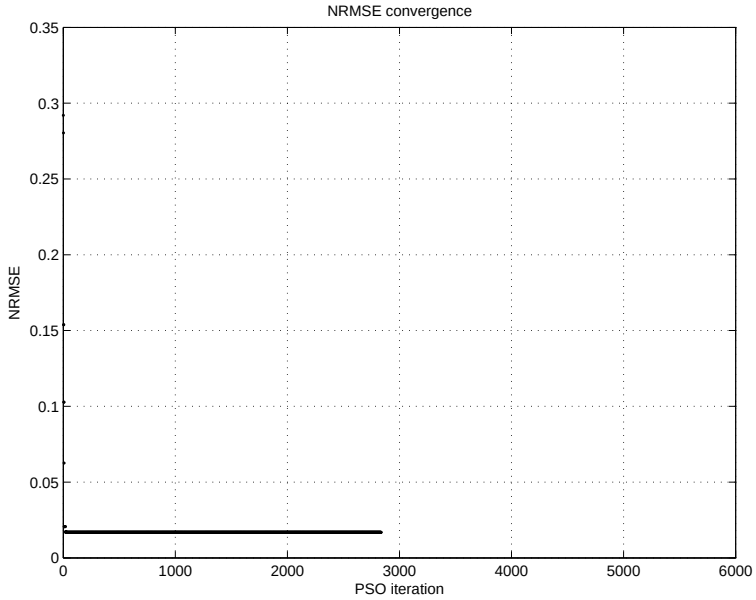

Figure 4.8: Validation NRMSE convergence, Example 2.2

the number of trainings. This is possible because, for some values of ε, all points may lay

Table 4.14: Real failure time and predictions by SVR, Example 2.2

Validation			**Test**		
i	T_i	**Predicted** T_i	i	T_i	**Predicted** T_i
26	7.3	7.4145	33	8.1	8.0696
27	7.3	7.4145	34	8.3	8.1622
28	7.7	7.4145	35	8.4	8.3464
29	7.7	7.7902	36	8.4	8.4381
30	7.8	7.7902	37	8.5	8.4381
31	7.9	7.8836	38	8.7	8.5294
32	8.0	7.9767	39	8.8	8.7109
			40	9.0	8.8011
NRMSE		$1.6827 \cdot 10^{-2}$	NRMSE		$1.3412 \cdot 10^{-2}$
MAPE (%)		1.2344	MAPE (%)		1.1299
MSE		$1.6682 \cdot 10^{-2}$	MSE		$1.3087 \cdot 10^{-2}$

Table 4.15: Support vectors' details, Example 2.2

(x,y)	α	α^*	**Type**
(1.6, 2.0)	0	390.8909	free
(3.9, 4.5)	1936.7744	0	bounded
(4.5, 4.6)	0	975.3477	free
(6.0, 6.0)	0	1936.7744	bounded
(6.5, 6.5)	0	570.5358	free
(6.7, 7.0)	1936.7744	0	bounded

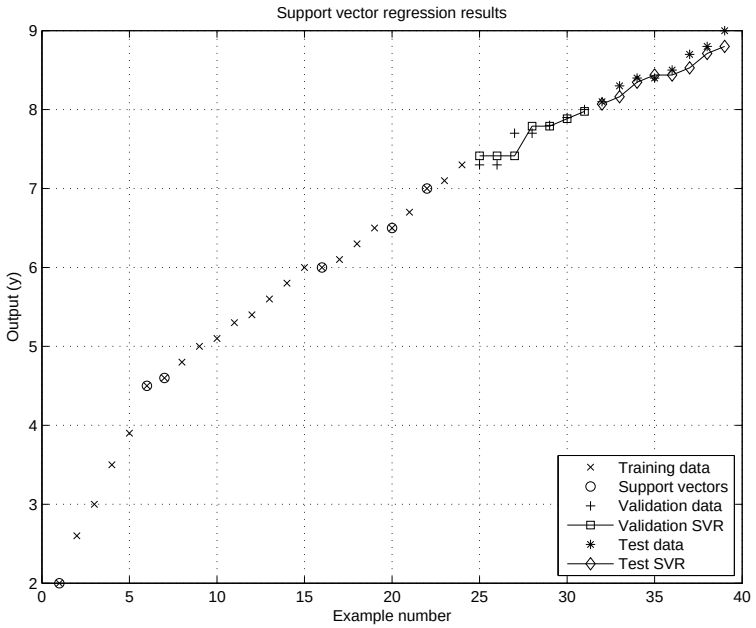

Figure 4.9: SVR results, Example 2.2

Table 4.16: Miles to failure ($\times$ 1000 hours) of a car engine. Adapted from Xu *et al.* (2003), p. 262-263

Number	MTF	Number	MTF	Number	MTF	Number	MTF
1	37.1429	26	35.8095	51	37.1429	76	36.3810
2	37.4286	27	36.9524	52	37.8095	77	38.0000
3	37.6190	28	37.6190	53	38.0952	78	38.1905
4	38.5714	29	37.8095	54	38.6667	79	38.6667
5	40.0000	30	38.0952	55	40.0619	80	38.6667
6	35.8095	31	36.8571	56	36.1905	81	37.1429
7	36.2857	32	38.0952	57	36.3810	82	37.6190
8	36.2857	33	38.0952	58	37.0476	83	37.6190
9	36.4762	34	38.3810	59	37.2381	84	38.0952
10	38.1905	35	39.0476	60	38.0000	85	39.0476
11	36.1905	36	37.2381	61	35.7143	86	36.2857
12	36.8571	37	37.3333	62	36.4762	87	37.1429
13	37.6190	38	37.5238	63	37.3333	88	37.5238
14	37.8095	39	37.8095	64	37.6190	89	37.8095
15	38.7619	40	38.5714	65	38.4762	90	38.0000
16	35.9048	41	37.1429	66	36.8571	91	36.8571
17	36.4762	42	37.2381	67	37.1429	92	37.0476
18	36.8571	43	37.6190	68	37.9048	93	37.9048
19	37.1429	44	38.1905	69	38.0952	94	38.1905
20	37.4286	45	38.5714	70	38.8571	95	39.5238
21	37.4286	46	36.0952	71	37.1429	96	35.4286
22	37.6190	47	37.2381	72	37.6190	97	36.0000
23	38.3810	48	37.4286	73	37.6190	98	37.7143
24	38.5714	49	37.5238	74	37.8095	99	38.0952
25	39.4286	50	39.0476	75	38.3810	100	38.5714

within the ε-tube, resulting in a model without support vectors. In this way, the first part of the regression function vanish. Then, when LIBSVM predicts based on a "machine" with these features, it returns only a constant value equal to b_0. This situation is not desirable, thus when the outcome of a training is an SVM without support vectors, LIBSVM is not allowed to predict and the fitness values associated with the particle under consideration remains unaltered.

The run which resulted in the smallest test NRMSE is related to the following parameter values: $C = 18.2629$, $\varepsilon = 7.9411 \cdot 10^{-2}$ and $\gamma = 6.8979 \cdot 10^{-1}$. In this particular run, PSO+SVM steps continued up to the 4225^{th} iteration and, like all other runs, stopped after 600 iterations with the same best global validation NRMSE.

Figure 4.10 depicts the particle swarm in three different moments. The final swarm, as in Example 1, forms some clusters. However it can be noticed from the axes ranges that the parameter values are rather concentrated in those ranges if compared with their respective original intervals. The validation NRMSE convergence can be visualized in Figure 4.11.

Table 4.18 shows the real and predicted values by the SVM which provided the smallest test NRMSE. Table 4.19, in turn, lists the support vectors, the associated Lagrange multipliers and also their classification as free or bounded. From the 80 training points, 68 were selected as support vectors and only 5 of them were free. This high number of support vectors can be

Table 4.17: Descriptive statistics of parameters and error functions, stop criteria frequency and performance for 30 PSO+SVM runs, *lbest*, Example 3

		Minimum	Maximum	Median	Mean	Std. dev.*
Parameters	C	16.3248	30.1075	21.7998	22.5316	5.4798
	ε	$7.9411 \cdot 10^{-2}$	$2.3345 \cdot 10^{-1}$	$1.6092 \cdot 10^{-1}$	$1.5861 \cdot 10^{-1}$	$7.6099 \cdot 10^{-2}$
	γ	$4.9354 \cdot 10^{-1}$	$7.2213 \cdot 10^{-1}$	$5.9863 \cdot 10^{-1}$	$6.0241 \cdot 10^{-1}$	$1.0515 \cdot 10^{-1}$
Validation error	NRMSE	$8.5041 \cdot 10^{-3}$	$9.1959 \cdot 10^{-3}$	$8.8512 \cdot 10^{-3}$	$8.8506 \cdot 10^{-3}$	$3.5098 \cdot 10^{-4}$
	MAPE (%)	$6.2633 \cdot 10^{-1}$	$7.0907 \cdot 10^{-1}$	$6.7005 \cdot 10^{-1}$	$6.6897 \cdot 10^{-1}$	$4.0743 \cdot 10^{-2}$
	MSE	$1.0273 \cdot 10^{-1}$	$1.2012 \cdot 10^{-1}$	$1.1145 \cdot 10^{-1}$	$1.1144 \cdot 10^{-1}$	$8.8249 \cdot 10^{-3}$
Test error	NRMSE	$1.8969 \cdot 10^{-2}$	$1.9468 \cdot 10^{-2}$	$1.9265 \cdot 10^{-2}$	$1.9242 \cdot 10^{-2}$	$2.2394 \cdot 10^{-4}$
	MAPE (%)	1.4338	1.4978	1.4697	1.4679	$2.9455 \cdot 10^{-2}$
	MSE	$5.0739 \cdot 10^{-1}$	$5.3441 \cdot 10^{-1}$	$5.2338 \cdot 10^{-1}$	$5.2217 \cdot 10^{-1}$	$1.2150 \cdot 10^{-2}$
				Absolute (relative, %) frequency		
Stop criteria	Maximum number of iterations (6000)			0 (0)		
	Equal best fitness for 600 iterations			30 (100)		
	Tolerance $\delta = 1 \cdot 10^{-12}$			0 (0)		
				Metric value		
Performance	Mean time per run (minutes)			4.9337		
	Mean number of trainings			66653.6333		
	Mean number of predictions			66563.8000		

*Standard deviation

justified by the small ε (near 0.2% of the mean of output training values), which results in a thin ε-tube and also by the complex behavior of the time series, as can be visualized in Figure 4.12.

Table 4.18: Real MTF and predictions by SVR ($\times$ 1000 hours), Example 3

Validation			Test		
Number	**MTF**	**Predicted MTF**	**Number**	**MTF**	**Predicted MTF**
82	37.6190	37.5078	91	36.8570	38.3673
83	37.6190	37.8891	92	37.0480	37.4383
84	38.0960	37.8891	93	37.9050	37.4813
85	39.0470	38.4474	94	38.1900	38.2621
86	36.2860	36.9183	95	39.5240	38.4914
87	37.1430	37.0125	96	35.4290	35.7357
88	37.5240	37.5082	97	36.0000	36.3744
89	37.8090	37.7766	98	37.7140	36.6496
90	38.0000	38.1398	99	38.0950	38.0127
			100	38.5714	38.4468
NRMSE		$8.5074 \cdot 10^{-3}$	NRMSE		$1.8969 \cdot 10^{-2}$
MAPE (%)		$6.3127 \cdot 10^{-1}$	MAPE (%)		1.4338
MSE		$1.0281 \cdot 10^{-1}$	MSE		$5.0739 \cdot 10^{-1}$

Zio *et al.* (2008) applied an Infinite Impulse Response Locally Recurrent Neural Network (IIR-LRNN) to solve this example and updated the results presented in Xu *et al.* (2003). These results are repeated in Table 4.20 along with the best test NRMSE provided by the PSO+SVM

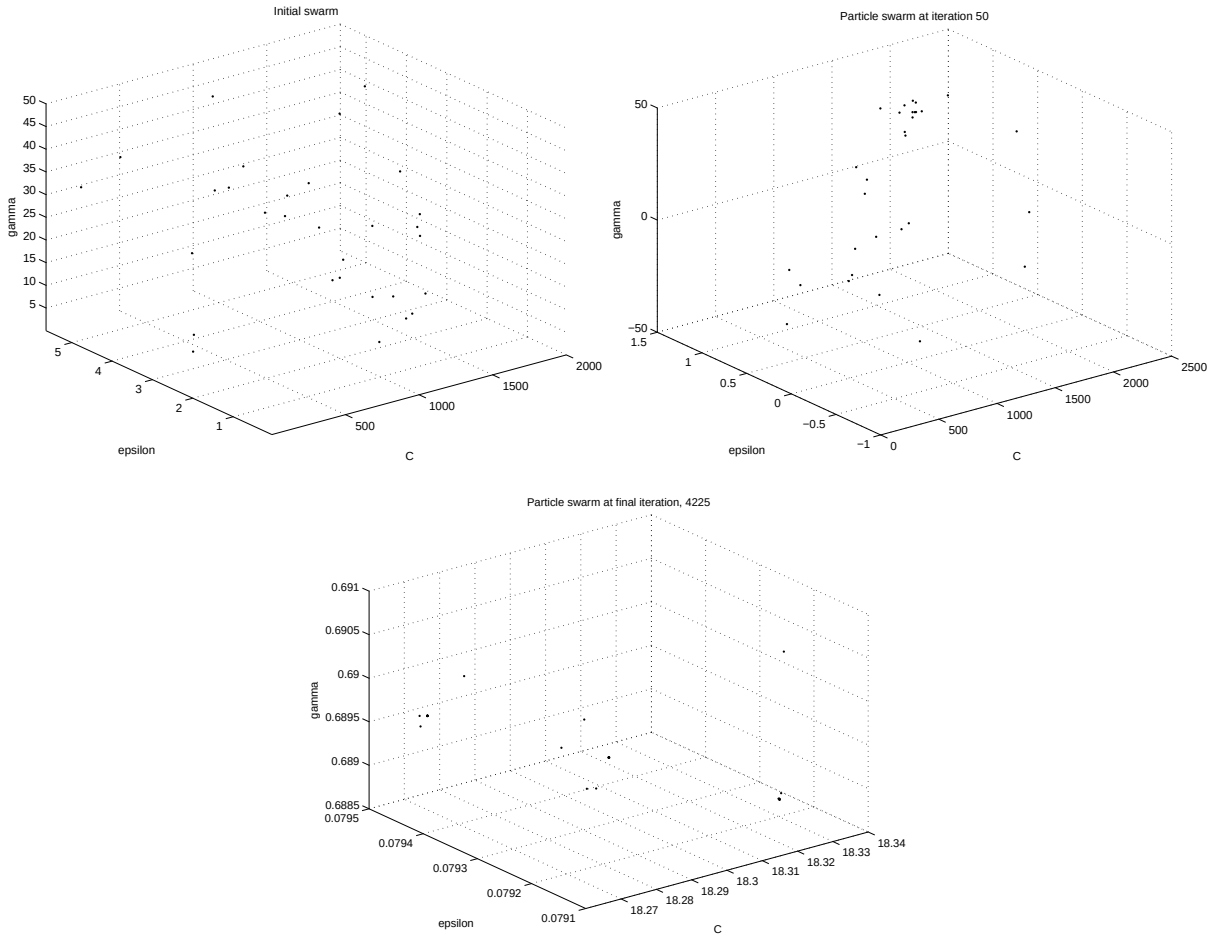

Figure 4.10: Swarm evolution during PSO, Example 3

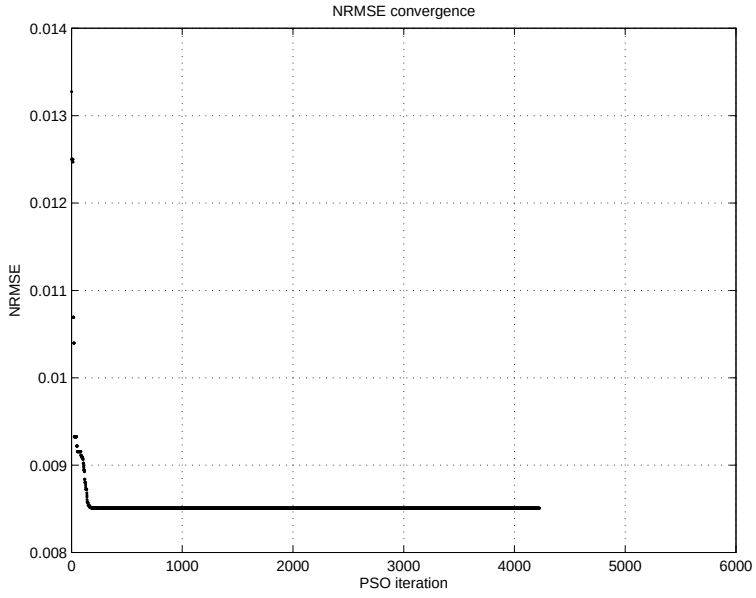

Figure 4.11: Validation NRMSE convergence, Example 3

methodology. The latter value occupies the third position in the rank. One possible reason for that is the smaller number of training points, due to the validation phase adopted. In order to

Table 4.19: Support vectors' details, Example 3

(x,y)	α	α^*	**Type**	(x,y)	α	α^*	**Type**
(37.1429, 37.4286)	0	7.3029	free	(37.1420, 37.2390)	8.9927	0	free
(37.4286, 37.6190)	18.2629	0	bounded	(37.6190, 38.1900)	0	18.2629	bounded
(37.6190, 38.5714)	18.2629	0	bounded	(38.1900, 38.5710)	18.2629	0	bounded
(38.5714, 40.0000)	0	15.1115	free	(38.5710, 36.0960)	0	18.2629	bounded
(40.0000, 35.8095)	0	18.2629	bounded	(36.0960, 37.2380)	0	18.2629	bounded
(35.8095, 36.2857)	0	18.2629	bounded	(37.2380, 37.4280)	18.2629	0	bounded
(36.2857, 36.2857)	0	18.2629	bounded	(37.4280, 37.5240)	18.2629	0	bounded
(36.2857, 36.4762)	18.2629	0	bounded	(39.0480, 37.1430)	18.2629	0	bounded
(36.4762, 38.1905)	0	18.2629	bounded	(37.1430, 37.8090)	18.2629	0	bounded
(38.1905, 36.1905)	18.2629	0	bounded	(38.0950, 38.6670)	18.2629	0	bounded
(36.8571, 37.6190)	0	18.2629	bounded	(38.6670, 40.0620)	18.2629	0	bounded
(37.6190, 37.8095)	18.2629	0	bounded	(40.0620, 36.1900)	0	18.2629	bounded
(37.8095, 38.7619)	0	18.2629	bounded	(36.1900, 36.3810)	0	18.2629	bounded
(38.7619, 35.9048)	0	18.2629	bounded	(36.3810, 37.0480)	18.2629	0	bounded
(36.4762, 36.8571)	0	18.2629	bounded	(37.0480, 37.2380)	0	18.2629	bounded
(36.8571, 37.1429)	0	18.2629	bounded	(37.2380, 38.0000)	1.0729	0	free
(37.6190, 38.3810)	18.2629	0	bounded	(38.0000, 35.7140)	18.2629	0	bounded
(38.3810, 38.5714)	18.2629	0	bounded	(35.7140, 36.4770)	18.2629	0	bounded
(38.5714, 39.4286)	18.2629	0	bounded	(36.4770, 37.3330)	0	18.2629	bounded
(39.4286, 35.8095)	0	18.2629	bounded	(37.6190, 38.4760)	0	18.2629	bounded
(35.8095, 36.9524)	18.2629	0	bounded	(38.4760, 36.8570)	18.2629	0	bounded
(36.9524, 37.6190)	18.2629	0	bounded	(36.8570, 37.1430)	0	18.2629	bounded
(37.6190, 37.8095)	0	18.2629	bounded	(37.1430, 37.9050)	18.2629	0	bounded
(38.0952, 36.8571)	0	18.2629	bounded	(37.9050, 38.0950)	0	18.2629	bounded
(36.8571, 38.0960)	18.2629	0	bounded	(38.8570, 37.1430)	18.2629	0	bounded
(38.0960, 38.0950)	0	18.2629	bounded	(37.1430, 37.6190)	0	18.2629	bounded
(38.0950, 38.3810)	18.2629	0	bounded	(37.6190, 37.6190)	0	5.9140	free
(38.3810, 39.0470)	18.2629	0	bounded	(37.6190, 37.8100)	18.2629	0	bounded
(39.0470, 37.2390)	0	18.2629	bounded	(37.8100, 38.3810)	0	18.2629	bounded
(37.2390, 37.3330)	0	18.2629	bounded	(38.3810, 36.3810)	18.2629	0	bounded
(37.3330, 37.5240)	18.2629	0	bounded	(38.0000, 38.1900)	0	18.2629	bounded
(37.5240, 37.8090)	0	18.2629	bounded	(38.1900, 38.6670)	18.2629	0	bounded
(37.8090, 38.5720)	0	18.2629	bounded	(38.6670, 38.6670)	18.2629	0	bounded
(38.5720, 37.1420)	18.2629	0	bounded	(38.6670, 37.1420)	0	18.2629	bounded

investigate this issue, 30 runs of PSO+SVM with the same parameters and without a validation set were executed. In this way, the number of training points increased to 89 and the number of test entries remained the same (10). The best test NRMSE result was $1.2536 \cdot 10^{-2}$ with associated parameters quite different from the ones obtained with the validation and test procedures: $C = 1144.3099, \varepsilon = 5.7140 \cdot 10^{-3}, \gamma = 7.8020$. In addition, 37 of the 89 training points became support vectors, and 20 from these were free. Such test NRMSE would shift the PSO+SVM approach to a second position, only loosing for MLP-NN (Gaussian activation), but for a small amount, in the order of 10^{-4}.

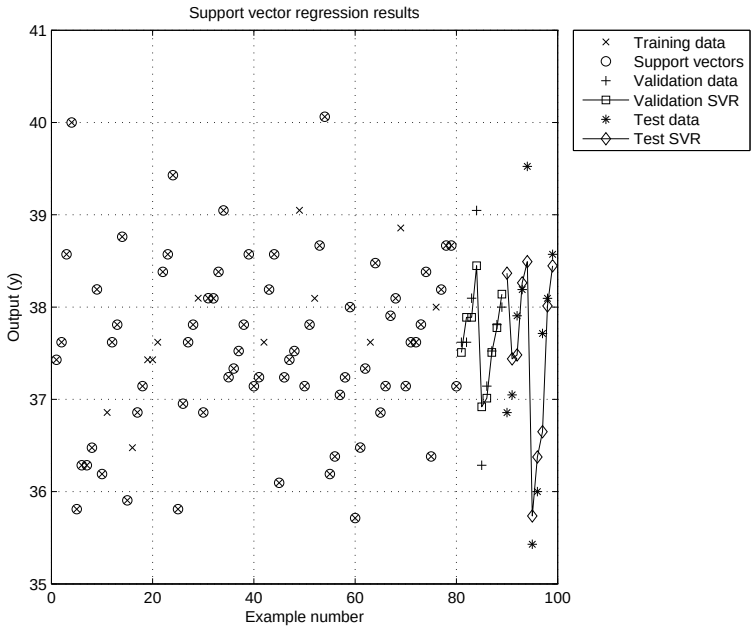

Figure 4.12: SVR results, Example 3

Table 4.20: Test NRMSE from different forecast models, Example 3. Updated from Xu *et al.* (2003), p. 264, Zio *et al.* (2008)

Method	**Test NRMSE**
PSO+SVM	$1.8969 \cdot 10^{-2}$*; $1.2536 \cdot 10^{-2}$**
RBF-NN (Gaussian activation)	$2.1100 \cdot 10^{-2}$
MLP-NN (Gaussian activation)	$1.2200 \cdot 10^{-2}$
MLP-NN (logistic activation)	$1.5600 \cdot 10^{-2}$
IIR-LRNN	$1.5800 \cdot 10^{-2}$
ARIMA	$4.2200 \cdot 10^{-2}$

* With validation; ** Without validation

4.4 Example 4: Time Between Failures of Oil Production Wells

In this example, differently from the previous ones, it is resolved a regression problem involving real data related to features (numerical and categorical) of the system under analysis.

The systems of interest are oil production wells. The reliability metric considered is the TBF, which is believed to be influenced by specific features of the wells. The failure of wells represent the interruption of oil production and, as a consequence, economical losses. In this way, the prediction of the TBF of these systems may permit preventive actions so as to reduce or even avoid the effects of the very next failure.

This example is based on a database that was presented by Barros Jr. (2006). It contains records of TBF, TTR and related factors of different onshore wells from 1983 to 2006. The author makes a comprehensive analysis of the variables of the database and proposes the use of BNs integrated with Markov chains to estimate the availability of oil wells. The database incor-

porates the concept of *socket* (ASCHER & FEINGOLD, 1984), which loosely speaking means that the records are associated with the equipments installed in the wells and are not related to the equipments themselves. For example, it is expected that the behaviors of pumps consecutively installed in a specific place of a well are approximately the same, since the environmental and operational conditions to which they are subjected have not changed.

According to Barros Jr. (2006), in the considered context, it has been observed that the most critical components of an oil production well are the pump, the rods and the columns. These equipments are related to the artificial elevation of oil to the surface. The two considered artificial elevation methods are the mechanical and the one via progressive cavities. For both types of elevation methods the columns have the same role of permitting the passage of the rods and also of isolating the well boundaries. In the mechanical elevation, the rotating energy of an engine is transformed in an alternating motion that is transmitted to the rods and the pump is responsible to transmit the energy to the fluid, which is brought to the surface. In the elevation by progressive cavities, in turn, the rotating energy of an engine on the surface is transmitted to the rods that also rotates. The rotating rods transmit energy to the pump, which is within the well and whose components' disposition permits the passage of the oil.

In this example, it is considered the wells' failures due to failures on their installed rods. The elevation method type, the kind of installed filter and the concentration of water and solids within the well are factors that influence the rods' performance. These factors, along with the number of installed rods, are the variables considered to predict the wells' TBF. Hence, only a subset of the entire database is used. Despite the great number of entries in this subset (more than 10.000), there are many empty cells or cases that present inconsistent information. Also, the database involves essentially non-homogeneous data, given that the records concern various wells located in different places and consequently subject to diverse environmental factors.

As an attempt to reduce the effects of the data non-homogeneity, it was selected a specific group of wells that are located essentially in the same geographical area with similar characteristics. The cases that presented any empty cell associated with a variable of interest were eliminated. After pre-processing the database, a data set with 214 examples was obtained and divided in a training set with the first 170 points, a validation set with the following 20 entries and a test set formed by the last 24 examples.

The description of the input variables selected from the database are presented in Table 4.21 along with the characteristics of the TBF itself. Each one of the input variables reflects a specific feature of the wells taken into account. Basically, it is considered the percentage of water and solids within the wells; the number of installed rods of different lengths ($3/4$, $5/8$, $7/8$, 1, in inches); the absence (N) or presence of a filter (if present, its type C, S or F is recorded and the related quality increases from C to F); the way the oil is pumped upwards (Progressive Cavities Pumping (PCP) or Mechanical Pumping (MP)).

Notice that x_6 and x_7 are categorical variables. The former has an ordinal scale, that is, the associated categories have an underlying order, but can not be quantified. The latter, in turn, has

Table 4.21: Selected variables that influence the TBF

Name	Variable	Type	Range/Categories
x_1	Percentage of water and solids in the well	Numerical	$[0, 98.3]$
x_2	Number of $3/4$" rods installed	Numerical	$[0, 104]$
x_3	Number of $5/8$" rods installed	Numerical	$[0, 101]$
x_4	Number of $7/8$" rods installed	Numerical	$[0, 96]$
x_5	Number of 1" rods installed	Numerical	$[0, 95]$
x_6	Type of installed filter	Categorical	N*, C, S, F
x_7	Type of elevation method	Categorical	PCP, MP
y	Time Between Failures (TBF, in days)	Numerical	$[2, 3469]$

*No filter installed

a nominal scale and then its categories only denotes the sort of elevation method, which forbids any sort of ordering or arithmetical operations. Nevertheless, the SVM training problem only accepts numerical values, thus the categorical variables have to be treated before being used by the PSO+SVM algorithm. Traditional statistical regression methods often handle categorical variables by transforming them into indicator or dummy variables (MONTGOMERY *et al.*, 2001). That is, if a variable has two associated categories, for example the type of pump used, the indicator variable x^{ind} is either 0 to denote that a PCP is used or 1 to indicate that a MP is installed. In general, if a categorical variable has r related categories, then $r-1$ indicator variables are necessary. For SVM, Hsu *et al.* (2009) also recommend the use of indicator variables to handle categorical variables. The transformation of the categorical variables x_6 and x_7 are shown in Table 4.22.

Table 4.22: Transformation of categorical variables x_6 and x_7 into indicator variables

x_6	$x_{6,1}^{ind}$	$x_{6,2}^{ind}$	$x_{6,3}^{ind}$	x_7	x_7^{ind}
N	0	0	0	PCP	0
C	0	0	1	MP	1
S	0	1	0		
F	1	0	0		

It can be observed from Table 4.21 that the TBF' interval is quite different from the ranges of the numerical input variables. In this way, in order to obtain better results, it is necessary to scale the data. The usual scaling range is [0,1], however, a 0 value for the output y gives rise to a division by 0 in the computation of MAPE. Thus, instead of using [0,1], the data is scaled within [1,2]. In addition, each variable is scaled by using its proper minimum and maximum training values (scaling factors), which results in 8 different scales, 7 for inputs and 1 for the TBF. In other words, each dimension of the input vector $\mathbf{x}$ as well as the output variable y are scaled on their own. Also, as the validation and test sets play the role of unseen data, they are scaled using the scaling factors from the training set. The formula is as follows:

$$\text{Scaled } x_{ij} = \frac{(x_{ij} - x_{min,j})}{(x_{max,j} - x_{min,j})} \cdot (up - low) + low \tag{4.2}$$

where i is the example index, j is the dimension of $\mathbf{x}_i$ under consideration, low and up are the boundaries of the scale interval, $x_{min,j}$ and $x_{max,j}$ are respectively the minimum and maximum training values of the j^{th} dimension of $\mathbf{x}_i$ for $i = 1, 2, \ldots, \ell$. When validation and test points are considered ($i > \ell$), the same scaling factors $x_{min,j}$ and $x_{max,j}$ are used. Moreover, for the present example, $low = 1$ and $up = 2$. Following the same reasoning from Equation (4.2), one obtains a scaling expression for the output variable y:

$$\text{Scaled } y_i = \frac{(y_i - y_{min})}{(y_{max} - y_{min})} \cdot (up - low) + low \tag{4.3}$$

in which y_{min} and y_{max} are the minimum and maximum training values of y, that is, $y_{min} = \min_i(y_1, y_2, \ldots, y_i, \ldots, y_\ell)$ and $y_{max} = \max_i(y_1, y_2, \ldots, y_i, \ldots, y_\ell)$. Again, when validation and test values of y are considered ($i > \ell$), y_{min} and y_{max} are also used.

Montgomery *et al.* (2001) assert that, although indicator variables with 0-1 values are often a best choice, any two distinct values (*e.g.* 1 and 2) for an indicator variable would be satisfactory. Hence, in order to follow the same scale of the numerical variables, the categorical ones are transformed in 1-2 indicator variables. In Table 4.22, 0 and 1 values are then substituted by 1 and 2, in this order.

After applying the necessary transformations and scales, the proposed PSO+SVM methodology can be used. The PSO parameters as well as the bounds for C and γ were the same as the ones adopted in the previous time series based examples. However, $\varepsilon \in [1.1106 \cdot 10^{-3}, 1.6659 \cdot 10^{-1}]$.

Descriptive statistics of the 30 PSO+SVM runs are presented in Table 4.23. All runs attained the stop criterion related to equal best fitness values for 600 consecutive iterations. The parameter values related to the "machine" that provided the smallest test NRMSE are $C = 4.4422$, $\varepsilon = 1.1774 \cdot 10^{-2}$ and $\gamma = 2.1226 \cdot 10^{-1}$. The related errors are listed in Table 4.24.

In Figure 4.13, the evolution of the particle swarm can be visualized in three different moments. Notice that in iteration 50 there were infeasible particles, but at the 5276^{th} and last iteration all particles were within the feasible search space. The validation NRMSE convergence is shown in Figure 4.14.

For this example, the SVR results are presented in separate pictures, given that a unique graphic would be very dense due to the number of data entries involved and would render its analysis difficult. In this way, Figure 4.15 presents the SVR training results, whilst Figure 4.16 depicts the SVR validation and test outcomes. From the former figure, it can be observed that the majority of the training examples became support vectors. Indeed, from the 170 training points, 141 were selected as support vectors (120 bounded and 21 free). From the latter figure,

Table 4.23: Descriptive statistics of parameters and error functions, stop criteria frequency and performance for 30 PSO+SVM runs, *lbest*, Example 4

		Minimum	Maximum	Median	Mean	Std. dev.*
Parameters	C	2.6099	11.5072	2.8707	3.7511	2.6397
	ε	$1.1774 \cdot 10^{-2}$	$2.1867 \cdot 10^{-2}$	$1.5028 \cdot 10^{-2}$	$1.5799 \cdot 10^{-2}$	$2.1879 \cdot 10^{-3}$
	γ	$1.1226 \cdot 10^{-1}$	$7.5687 \cdot 10^{-1}$	$6.7570 \cdot 10^{-1}$	$6.5571 \cdot 10^{-1}$	$1.0901 \cdot 10^{-1}$
Validation error	NRMSE	$3.1551 \cdot 10^{-2}$	$3.1852 \cdot 10^{-2}$	$3.1561 \cdot 10^{-2}$	$3.1595 \cdot 10^{-2}$	$9.3203 \cdot 10^{-5}$
	MAPE (%)	2.2365	2.4069	2.2412	2.2602	$5.0494 \cdot 10^{-2}$
	MSE	$1.0697 \cdot 10^{-3}$	$1.0903 \cdot 10^{-3}$	$1.0704 \cdot 10^{-3}$	$1.0728 \cdot 10^{-3}$	$6.3498 \cdot 10^{-6}$
Test error	NRMSE	$4.2354 \cdot 10^{-2}$	$4.8617 \cdot 10^{-2}$	$4.7869 \cdot 10^{-2}$	$4.7234 \cdot 10^{-2}$	$1.8450 \cdot 10^{-3}$
	MAPE (%)	3.4024	4.1029	3.9496	3.9067	$1.7208 \cdot 10^{-1}$
	MSE	$1.9059 \cdot 10^{-3}$	$2.5112 \cdot 10^{-3}$	$2.4346 \cdot 10^{-3}$	$2.3739 \cdot 10^{-3}$	$1.7768 \cdot 10^{-4}$
				Absolute (relative, %) frequency		
Stop criteria	Maximum number of iterations (6000)			0 (0)		
	Equal best fitness for 600 iterations			30 (100)		
	Tolerance $\delta = 1 \cdot 10^{-12}$			0 (0)		
				Metric value		
Performance	Mean time per run (minutes)			14.6685		
	Mean number of trainings			67929.3000		
	Mean number of predictions			67929.3000		

*Standard deviation

Table 4.24: Validation and test errors from the "machine" with the smallest test NRMSE, Example 4

Error function	Validation	Test
NRMSE	$3.1747 \cdot 10^{-2}$	$4.2354 \cdot 10^{-2}$
MAPE (%)	2.4069	3.4024
MSE	$1.0831 \cdot 10^{-3}$	$1.9059 \cdot 10^{-3}$

notice that despite the low quantity of precise predictions, the machine attempts to catch the trend of the validation and test data. Additionally, both figures have the scaled output as vertical axis and the example number as the horizontal one, provided that the input vectors are multi-dimensional and it is not possible to draw a graphic involving all input vectors with the output TBF values.

4.5 Discussion

In all presented examples, the validation NRMSE dropped to values very near the best one found in early stages of the PSO algorithm, which suggests the ability of PSO in finding good solutions even with a few number of iterations. Additionally, it can be inferred that the SVM, with an appropriate set of parameters, is able to provide excellent results to reliability

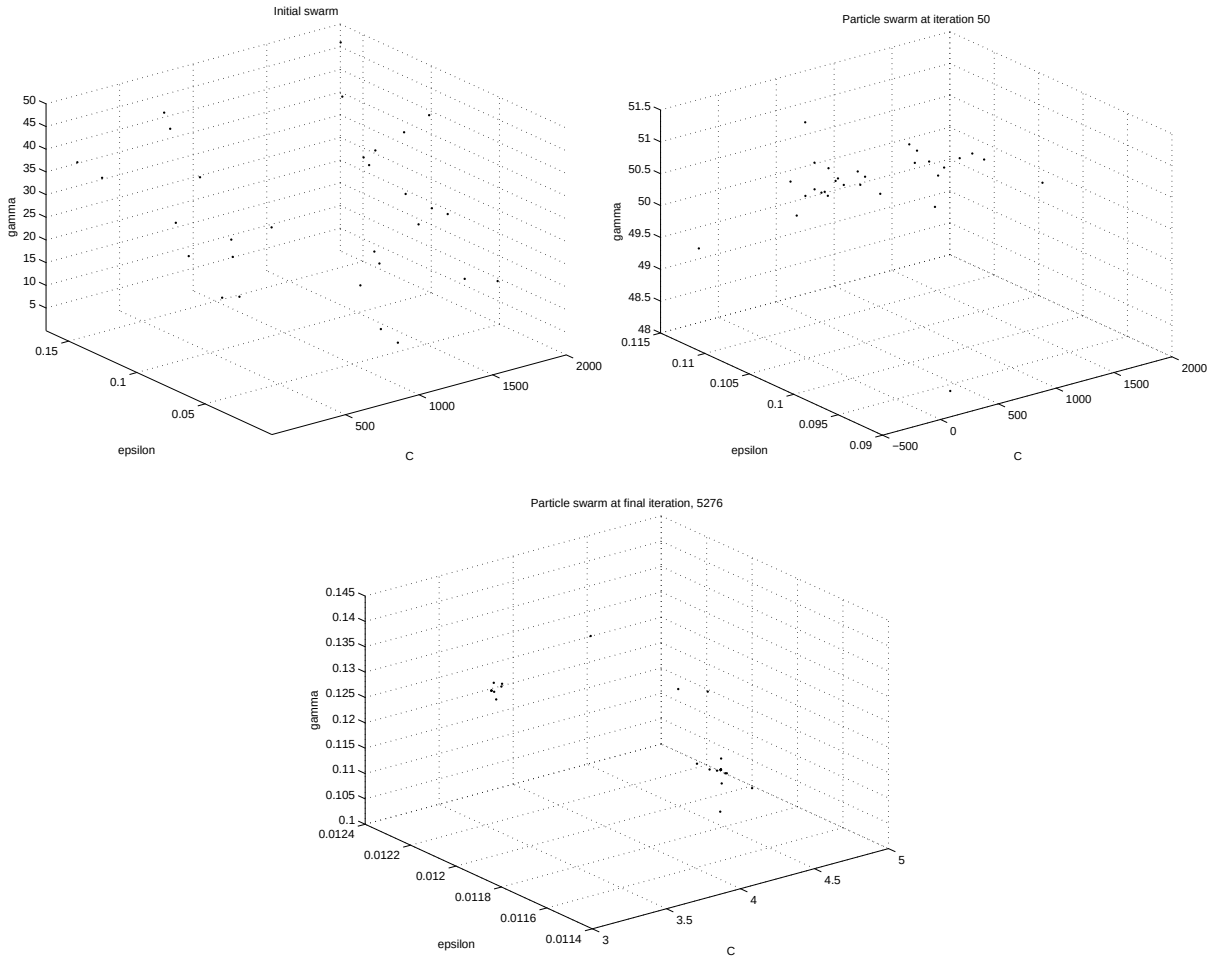

Figure 4.13: Swarm evolution during PSO, Example 4

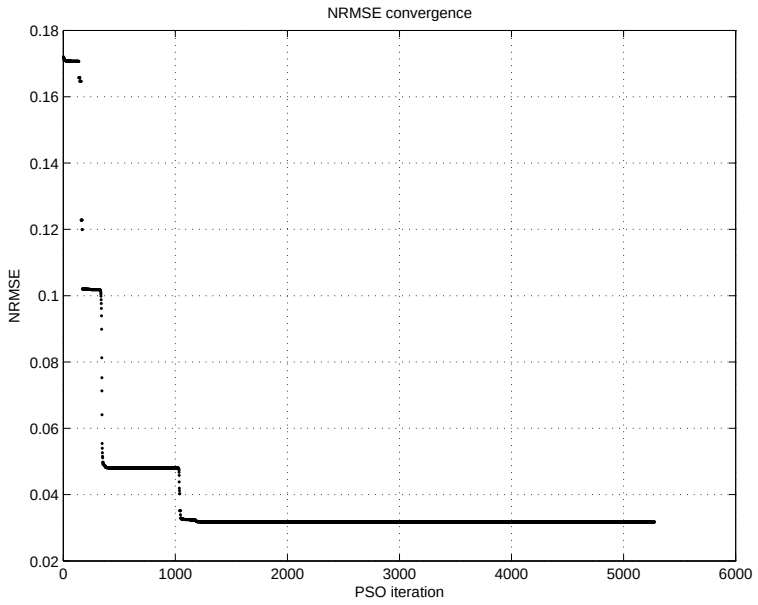

Figure 4.14: Validation NRMSE convergence, Example 4

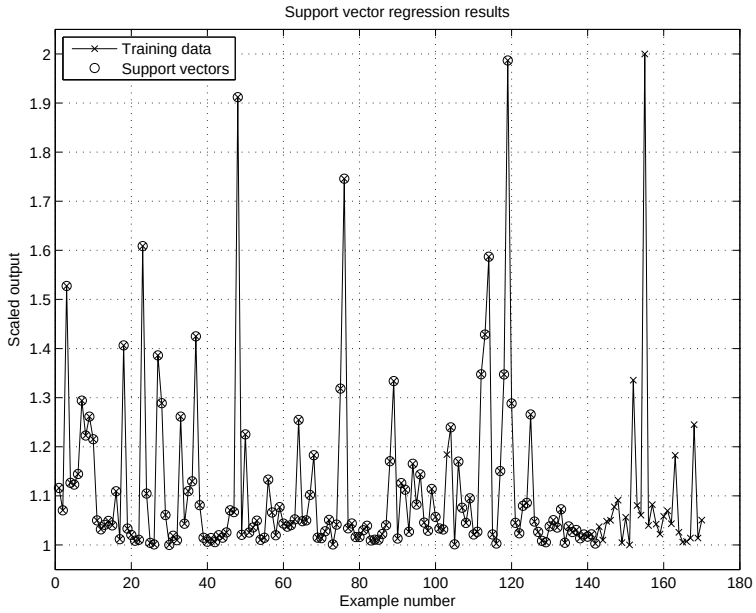

Figure 4.15: SVR training results, Example 4

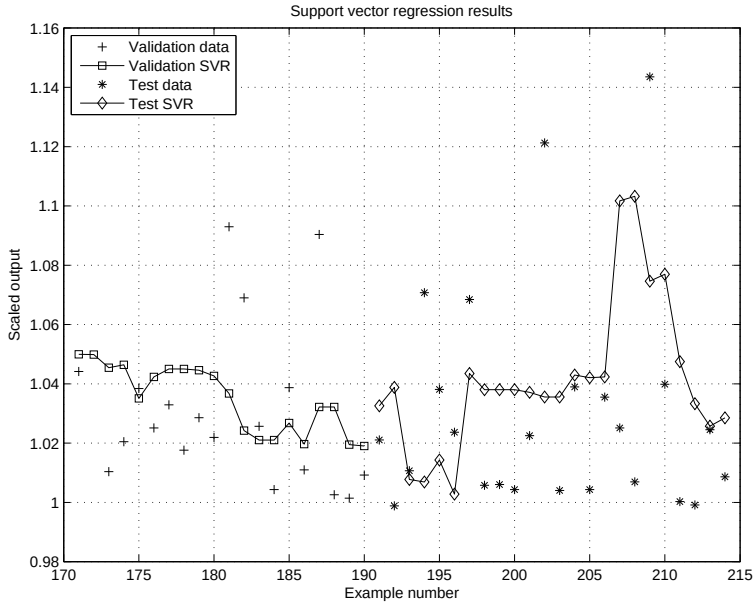

Figure 4.16: SVR validation and test results, Example 4

prediction based on time series, better than or comparable to its competitive models such as NN and ARIMA.

Considering Example 4, the quality of the obtained results certainly is related to the quality of the data set used. Firstly, even considering wells from the same geographical area, the original database subset presented, essentially, non-homogeneous records and many empty cells or cases with contradictory information. Moreover, the use of categorical variables influences the SVM performance, since it involves a quadratic programming problem in its training step, which treat all variables as if they were numerical. Hence, the use of categorical variables are

indicated for the cases when there is not a quantitative manner to measure the factor of interest. For example, transform a variable that is naturally numerical into a categorical one is usually not recommended.

Even with these shortcomings, the PSO+SVM was able to provide small NRMSE test values in Example 4. In this way, its performance would be certainly enhanced if the data set were originated from a database without errors. Additionally, other manners to handle categorical variables have to be analyzed.

In the majority of the examples, the descriptive statistics showed a high variance for the parameter representing the trade-off between training error and machine capacity (C). This fact indicates the difficulty in tuning this parameter and also in using techniques that assign only discrete values for it, such as the grid search model. If the inherent trade-off of SVM could be explicitly treated, C could be omitted, remaining only the other two parameters ε and γ to adjust. Indeed, Mierswa (2007) proposes a multi-objective approach of SVM, in which the minimization of training errors is one objective and the margin maximization is the other one. In this situation, C is no longer needed.

4.5.1 Performance Comparison Between *lbest* and *gbest* Models

All examples were solved also by the *gbest* model, in the same conditions of *lbest*, *i.e.*, using the same PSO parameters and SVM data set division among training, validation and test points. The test NRMSE is the metric of greatest interest because it provides an idea of the generalization ability of the "machine" under consideration. By taking the 30 test NRMSE values resulted from each PSO model as independent samples, a Wilcoxon-Mann-Whitney test (WACKERLY *et al.*, 2002) can be performed for each example so as to assess the chance of obtaining greater values for the test NRMSE values with the *gbest* model than with the *lbest* approach. In Table 4.25, the p-value of the one-sided test is presented for every case. Notice that the two examples concerning the reliability and failure times of turbochargers as well as Example 4 yielded a non-significant p-value for the level of significance of 5%.

Table 4.25: Mean test NRMSE values and Wilcoxon-Mann-Whitney test results *lbest* $\times$ *gbest*

Example	**Mean test NRMSE**		p**-value**
	lbest	*gbest*	
1	$2.0307 \cdot 10^{-2}$	$3.3670 \cdot 10^{-2}$	$9.3340 \cdot 10^{-5}$
2.1	$2.2999 \cdot 10^{-4}$	$5.5485 \cdot 10^{-4}$	$1.1490 \cdot 10^{-1}$
2.2	$3.1674 \cdot 10^{-2}$	$3.3395 \cdot 10^{-2}$	$9.1130 \cdot 10^{-2}$
3	$1.9242 \cdot 10^{-2}$	$2.0282 \cdot 10^{-2}$	$2.0160 \cdot 10^{-3}$
4	$4.7234 \cdot 10^{-2}$	$5.0871 \cdot 10^{-2}$	$6.1820 \cdot 10^{-1}$

An interesting point of the turbochargers failure times (Example 2.2) is that not all PSO runs could catch their increasing trend. Only 20% from the *gbest* runs were not able to predict output

values from the test set with the correct increasing trend against 26.6667% of the *lbest* runs. Nevertheless, the Wilcoxon-Mann-Whitney test was non-significant, *i.e.*, there is no evidence to affirm that the *gbest* provides better test NRMSE values than the *lbest* does, considering the level of significance of 5%.

Bratton & Kennedy (2007) asserts that usually the *gbest* approach presents a faster convergence if compared to *lbest*. However, for the presented examples, all *lbest* runs had a smaller mean time per run in absolute terms (Table 4.26). Also, apart from Example 1, the mean number of SVM predictions (fitness evaluations) was smaller for the *lbest* approach for all examples (Table 4.27). In order to statistically compare the required times by each model and also the number of predictions, Wilcoxon-Mann-Whitney tests may be performed for each case.

For the computational time case, Table 4.26 presents the p-values from the related statistical tests. Notice that, for a level of significance of 10%, all results were statistically significant. Table 4.27, in turn, provides the p-values resulted from the statistical tests associated with the mean number of predictions. Taking into account a level of significance of 10%, only the test concerning Example 1 was non-significant. Therefore, the *lbest* model is prone to require less time as well as less fitness evaluations than the *gbest* model does.

Table 4.26: Mean time per run (minutes) and Wilcoxon-Mann-Whitney test results *lbest* × *gbest*

Example	**Mean time per run (minutes)**		p**-value**
	lbest	*gbest*	
1	10.8342	13.2886	$6.3310 \cdot 10^{-2}$
2.1	1.0066	1.3956	$1.1710 \cdot 10^{-3}$
2.2	10.5648	15.3496	$6.9060 \cdot 10^{-2}$
3	4.9337	9.4805	$2.6620 \cdot 10^{-2}$
4	14.6685	16.9978	$8.4030 \cdot 10^{-2}$

Table 4.27: Mean number of predictions per run and Wilcoxon-Mann-Whitney test results *lbest* × *gbest*

Example	**Mean number of predictions per run**		p**-value**
	lbest	*gbest*	
1	116016.1667	106998.6333	$6.8340 \cdot 10^{-1}$
2.1	67989.7000	85499.2000	$3.7350 \cdot 10^{-2}$
2.2	68977.6667	82603.7000	$8.8710 \cdot 10^{-2}$
3	66563.8000	89619.9333	$1.0790 \cdot 10^{-2}$
4	67929.3000	89402.3000	$1.8560 \cdot 10^{-2}$

In this way, for the resolved examples, one can infer that the *lbest* approach is inclined to provide smaller values of test NRMSE than the *gbest* model does, or at least comparable ones. Moreover, for the considered examples, the *lbest* PSO has a tendency to converge more rapidly than the *gbest* approach.

5 CONCLUDING REMARKS

This chapter provides some concluding remarks. In addition, limitations as well as a description of the ongoing research along with some suggestions for future works are presented.

5.1 Conclusions

This work proposed a PSO+SVM methodology for solving reliability prediction problems. PSO+SVM combined with a validation set approach to guide the search for appropriate SVR parameters' values was validated with examples from literature concerning reliability prediction based on time series data. The results showed that PSO+SVM can achieve outcomes comparable to or better than the ones provided by other time series prediction tools, such as NN and ARIMA.

Moreover, PSO+SVM was applied to an example involving data from oil production wells. The TBF was predicted by considering specific characteristics of the system, differently from the other examples, which were all from literature and based on time series data. The input variables were both numerical and categorical. The latter were transformed into indicator variables, so as to be used by the SVR algorithm. In addition, the numerical input variables as well as the output variable were scaled in [1,2] in order to avoid scale problems and then to get improved solutions. Although the original database presented non-homogeneous data and some problems related to data gathering, the proposed PSO+SVM methodology was able to provide quite small error values.

PSO was adopted to tackle the SVM problem in the specific context of reliability prediction. The implemented PSO involved an empirical manner to avoid particles from exiting the feasible search space in early iterations of the algorithm by initially setting a small value for their maximum velocities. Also, both *lbest* and *gbest* communication network among particles were incorporated.

For all examples, apart from 13 runs in Example 1, the PSO runs converged before reaching the maximum number of iterations. This reflects the ability of the PSO algorithm to find good solutions in early steps of the algorithm. Also, for every example, the validation NRMSE, which guided the search for the parameters (fitness), did not present considerable differences along the 30 runs, which can be observed from the related standard deviations.

Furthermore, a comparison between *lbest* and *gbest* PSO models was performed. The results for the specific examples considered in this dissertation indicated that the *lbest* was faster and also provided test NRMSE values statistically comparable to or better than the ones yielded from the *gbest* approach.

Therefore, given the obtained results, the coupling of PSO with SVM is a promising methodology to tackle reliability prediction problems based on time series or on data related to

specific features of the system, for example, obtained by a condition monitoring procedure.

5.2 Limitations

Although the implemented PSO linked with LIBSVM is quite general and can be easily adapted to other application domains, the results achieved in this dissertation are limited to regression problems in the specific context of reliability prediction from time series data sets or from system features. Thus, they can not be generalized to other applications without a previous investigation of the behavior of the proposed methodology applied to the problem under consideration.

Additionally, PSO is a heuristic search procedure based on probabilities, then it does not guarantee the convergence to optimum point. However, it is a useful tool for complex objective functions that, for example, have no defined derivatives in their domains or whose search for their optimum is very burdensome. It is rather a mechanism to go in the "right" direction so as to obtain good solutions. In many contexts these good solutions are indeed valuable.

Furthermore, the generalization capability in all provided examples was assessed by the test NRMSE. This is an indication of such ability but by no means guarantees that the corresponding trained machine will have a good performance when predicting outputs from inputs not in the training, validation or test sets. If some time after the prediction of an outcome the real value could be observed, this new observation may be incorporated in the data set. With this procedure, the SVM can be periodically retrained and thus improve its performance in predicting the phenomenon under analysis.

This work considered that failure times and TBF predictions were related to systems subject to a single failure mode. The modeling of various failure modes can be tackled analytically by means of a competitive risks framework (COOKE, 1996), in which the different failure modes represent risks that compete for leading the system to a failed state. To handle different failure modes with an SVM approach, it would be necessary to associate an SVM to each one of them. In this way, the estimate of the system's very next failure time or TBF would be determined by the most critical failure mode at the moment, that is, the one with the smallest failure time or TBF returned by the associated SVM.

5.3 Ongoing Research

The following items can be cited as topics of ongoing or future research effort:

- The application of the proposed methodology in the context of fault diagnostic and prognostic. The former usually demands a SVM classification or a multi-classification task and the latter, often requires a SVR based on metrics obtained via condition monitoring.
- The combination of PSO+SVM with SMDP so as to allow for the development of a more

comprehensive tool to support maintenance decisions.

- A further investigation of the multi-objective approach for SVM.
- The SVM training problem may involve huge matrices, which can render the optimization procedure burdensome. The paralellization of the SVM code can be an alternative to this issue, by dividing the entire problem in smaller ones to be solved in parallel.
- The handling of different failure modes considering a PSO+SVM modeling.

REFERENCES

ASCHER, H.; FEINGOLD, H. *Repairable Systems Reliability: modeling, inference, misconceptions and their causes*. New York: Marcel Dekker, 1984.

BANKS, J.; CARSON, J. S.; NELSON, B. L.; NICOL, D. M. *Discrete event system simulation*. 3ed. Upper Saddle River: Prentice Hall, 2001.

BARROS JR., P. F. do R.. *Uma metodologia para análise de disponibilidade de sistemas complexos via hibridismo de redes Bayesianas e processos Markovianos*. Dissertação (Mestrado) - Universidade Federal de Pernambuco, Recife, August 2006.

BEN-HUR, A.; ONG, C. S.; SONNENBURG, S.; SCHÖLKOPF, B.; RÄTSCH, G. Support vector machines and kernels for computational biology. *PLoS Computational Biology*, vol. 4, 2008.

BONDY, J. A.; MURTY, U. S. R. *Graph theory*. New York: Springer, 2008.

BOYD, S.; VANDENBERGHE, L. *Convex optimization*. Cambridge: Cambridge University Press, 2004. Available at: <http://www.stanford.edu/∼boyd/cvxbook/>.

BRATTON, D.; KENNEDY, J. Defining a standard for particle swarm optimization. In: *Proceedings of the IEEE Swarm Intelligence Symposium*. Honolulu, United States: 2007.

BURGES, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, v. 2, p. 121–167, 1998.

CHANG, C.-C.; LIN, C.-J. *LIBSVM: a library for support vector machines*. 2001. Available at: <http://www.csie.ntu.edu.tw/∼cjlin/libsvm>.

CHANG, M.-W.; LIN, C.-J. Leave-one-out bounds for support vector regression model selection. *Neural Computation*, vol. 17, n. 5, p. 1188–1222, 2005.

CHAPELLE, O.; VAPNIK, V.; BOUSQUET, O.; MUKHERJEE, S. Choosing multiple parameters for support vector machines. *Machine Learning*, vol. 46, p. 131–159, 2002.

CHEN, B.-J.; CHANG, M.-W.; LIN, C.-J. Load forecasting using support vector machines: a study on eunite competition 2001. *IEEE Transactions on Power Systems*, vol. 19, n. 4, p. 1821–1830, 2004.

CHEN, K.-Y. Forecasting systems reliability based on support vector regression with genetic algorithms. *Reliability Engineering and System Safety*, vol. 92, p. 423–432, 2007.

COOKE, R. M. The design of reliability data bases, part II: competing risk and data compression. *Reliability Engineering & System Safety*, vol. 51, p. 209–223, 1996.

CRISTINIANI, N.; SHAWE-TAYLOR, J. *An introduction to support vetor machines and other kernel-based learning methods*. Cambridge: Cambridge Universty Press, 2000.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: *Proceedings ofthe Sixth International Symposium on Micro Machine and Human Science*. Nagoya, Japan: 1995. p. 39–43.

FAN, R.-E.; CHEN, P.-H.; LIN, C.-J. Working set selection using second order information for training svm. *Journal of Machine Learning Research*, vol. 6, p. 1889–1918, 2005.

FEI, S.-W.; WANG, M.-J.; MIAO, Y.-B.; TU, J.; LIU, C.-L. Particle swarm optimization-based support vector machine for forecasting dissolved gases content in power transformer oil. *Energy Conversion and Management*, vol. 50, p. 1604–1609, 2009.

FRÖHLICH, H.; ZELL, A. Efficient parameter selection for supportvector machines in classification and regression via model-based global optimization. In: *Proceedings of the International Conference on Neural Networks*. Montreal, Canada: 2005.

FULLER, W. A. *Introduction to statistical time series*. 2ed. New York: John Wiley & Sons, 1996.

GESTEL, T. V.; SUYKENS, J.; BAESTAENS, D.-E.; LAMBRECHTS, A.; LANCKRIET, G.; VANDAELE, B.; MOOR, B.; VANDEWALLE, J. Financial time series prediction using least squares support vector machines within the evidence framework. *IEEE Transactions on Neural Networks*, vol. 12, n. 4, p. 809–821, 2001.

GOLDBERG, D. E. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.

HAYKIN, S. *Neural networks*. 2ed. Upper Saddle River: Prentice Hall, 1999.

HONG, W.-C. Chaotic particle swarm optimization algorithm in a support vector regression electric load forecasting model. *Energy Conversion and Management*, vol. 50, p. 105–117, 2009.

HONG, W.-C.; PAI, P.-G. Predicting engine reliability by support vector machines. *International Journal of Advaned Manufacturing Technology*, vol. 28, p. 154–161, 2006.

HSU, C.-W.; CHANG, C.-C.; LIN, C.-J. *A practical guide to support vector classification*. 2009. Available at: <http://www.csie.ntu.edu.tw/∼cjlin/papers/guide/guide.pdf>.

HSU, C.-W.; LIN, S.-J. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, vol. 13, p. 415–425, 2002.

ITO, K.; NAKANO, R. Optimizing support vector regression hyperparameters based on cross-valiation. In: *Proceedings of the International Joint Conference on Neural Networks*. Portland, United States: 2003.

JARDINE, A. K. S.; LIN, D.; BANJEVIC, D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, vol. 20, p. 1483–1510, 2006.

JOACHIMS, T. Making large-scale svm learning practical. In: SCHÖLKOPF, B.; BURGES, C.; SMOLA, A. J. (Ed.). *Advances in kernel methods: support vector learning*. Cambridge: The MIT Press, 1999. p. 169–184.

KECMAN, V. *Learning and soft computing: support vector machines, neural networks and fuzzy logic models*. Cambridge: The MIT Press, 2001.

KECMAN, V. Support vector machines: an introduction. In: WANG, L. (Ed.). *Support vector machines: theory and applications*. Berlin Heidelberg: Springer-Verlag, 2005, (Studies in Fuzziness and Soft Computing, v. 177). p. 1–47.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *Proceedings of the IEEE International Conference on Neural Networks*. Perth, Australia: 1995.

KENNEDY, J.; EBERHART, R.; SHI, Y. *Swarm intelligence*. San Francisco: Morgan Kaufmann, 2001.

KORB, K. B.; NICHOLSON, A. E. *Bayesian artificial intelligence*. Florida: Chapman & Hall/CRC, 2003.

KWOK, J. T.-Y. The evidence framework applied to support vector machines. *IEEE Transactions on Neural Networks*, vol. 11, n. 5, p. 1162–1173, 2000.

LEE, M. M. S.; KEERTHI, S. S.; ONG, C. J.; DECOSTE, D. An efficient method for computing leave-one-out error in support vector machines with gaussian kernels. *IEEE Transactions on Neural Networks*, vol. 15, n. 3, p. 750–757, 2004.

LEWIS, E. E. *Introduction to reliability engineering*. Singapore: John Wiley & Sons, 1987.

LEWIS, R. M.; TORCZON, V.; TROSSET, M. W. Direct search methods: then and now. *Journal of Computational and Applied Mathematics*, vol. 124, p. 191–207, 2000.

LIN, S.-W.; YING, K.-C.; CHEN, S.-C.; LEE, Z.-J. Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert Systems with Applications*, vol. 35, p. 1817–1824, 2008.

LINS, I. D.; DROGUETT, E. L. Multiobjective optimization of redundancy allocaion problems in systemas with imperfect repairs via ant colony and discrete event simulation. In: *Proceedings of the European Safety & Reliability Conference (ESREL)*. Valencia, Spain: 2008.

LINS, I. D.; DROGUETT, E. L. Multiobjective optimization of availability and cost in repairable systemas via genetic algorithms and discrete event simulation. *Pesquisa Operacional*, vol. 29, p. 43–66, 2009.

MICHALEWICZ, Z. *Genetic algorithms + data structures*. 3ed. Berlin: Springer-Verlag, 1996.

MIERSWA, I. Controlling overfitting with multi-objective support vector machines. In: *Proceedings of the 2007 Genetic and Evolutionary Computation Conference (GECCO'07)*. London, England: 2007.

MOMMA, M.; BENNETT, K. P. A pattern search method for model selection of support vector regression. In: *Proceedings of the 2002 SIAM International Conference on Data Mining*. 2002. p. 261–274.

MONTGOMERY, D. C.; PECK, E. A.; VINING, G. G. *Introduction to Linear Regression Analysis*. 3ed. New York: John Wiley & Sons, 2001.

MORETTIN, P. A.; TOLOI, C. M. C. *Análise de Séries Temporais*. São Paulo: Edgar Blücher, 2004.

MOURA, M. J. das C.; DROGUETT, E. A. L. Mathematical formulation and numerical treatment based on transition frequency densities and quadrature methods for non-homogeneous semi-Markov processes. *Reliability Engineering & System Safety*, vol. 94, p. 342–349, 2009.

MOURA, M. J. das C.; LINS, I. D.; FIRMINO, P. R. A.; DROGUETT, E. L.; JACINTO, C. M. Semi-Markov decision processes for determining multiobjective optimal condition-based replacement policies. In: *Proceedings of the European Safety & Reliability Conference (ESREL)*. Prague, Czech Republic: 2009.

MÜLLER, K.-R.; SMOLA, A. J.; RÄTSCH, G.; SCHÖLKOPF, B.; KOHLMORGEN, J.; VAPNIK, V. Using support vector machines for time series prediction. In: SCHÖLKOPF, B.; BURGES, C. J. C.; SMOLA, A. (Ed.). *Advances in kernel methods: support vector learning*. Cambridge: The MIT Press, 1999. p. 243–253.

NOCEDAL, J.; WRIGHT, S. J. *Numerical optimization*. 2ed. New York: 2006.

PAI, P.-F. System reliability forecasting by support vector machines with genetic algorithms. *Mathematical and Computer Modelling*, vol. 43, p. 262–274, 2006.

PAI, P.-F.; HONG, W.-C. Support vector machines with simulated annealing algorithms in electricity load forecasting. *Energy Conversion & Management*, vol. 46, p. 2669–2688, 2005.

PAI, P.-F.; HONG, W.-C. Software reliability forecasting by support vector machines with simulated annealing algorithms. *The Journal of Systems and Software*, vol. 79, p. 747–755, 2006.

PLATT, J. C. Fast training of support vector machines using sequential minimal optimization. In: SCHÖLKOPF, B.; BURGES, C. J. C.; SMOLA, A. (Ed.). *Advances in kernel methods: support vector machines*. Cambridge: The MIT Press, 1998.

PLATT, J. C.; CRISTINIANI, N.; SHAWE-TAYLOR, J. Large margin DAG's for multiclass classification. *Advances in Neural Information Processing Systems*, vol. 12, p. 547–553, 2000.

RAMESH, R.; MANNAN, M. A.; POO, A. N.; KEERTHI, S. S. Thermal error measurement and modelling in machine tools. part ii. hybrid bayesian network – support vector machine model. *International Journal of Machine Tools & Manufacture: Design, Research and Application*, vol. 43, p. 405–419, 2003.

RAUSAND, M.; HOYLAND, A. *System reliability theory: models and statistical methods*. 2ed. New York: John Wiley & Sons, 2004.

REIS, G. L. dos; SILVA, V. V. da. *Geometria analítica*. 2ed. Rio de Janeiro: Livros Técnicos & Científicos, 1996.

RIGDON, S. E.; BASU, A. P. *Statistical methods for the reliability of repairable systems*. New York: John Wiley & Sons, 2000.

ROCCO, C. M.; MORENO, J. A. Fast monte carlo reliability evaluation using support vector machines. *Rliability Engineering & System Safety*, vol. 76, p. 237–243, 2002.

ROCCO, C. M.; ZIO, E. A support vector machine integrated system for the classification of operation anomalies in nuclear components and systems. *Rliability Engineering & System Safety*, vol. 92, p. 593–600, 2007.

ROSS, S. M. *Introduction to probability models*. 7. ed. San Diego: Academic Press, 2000.

SAMANTA, B.; NATARAJ, C. Use of particle swarm optimization for machinery fault detection. *Engineerinhg Applications of Artificial Intelligence*, vol. 22, p. 308–316, 2009.

SAPANKEVYCH, N.; SANKAR, R. Times series prediction using support vector machines: a survey. *IEEE Computational Intelligence Magazine*, vol. 4, p. 24–38, 2009.

SCHÖLKOPF, B.; SMOLA, A. J. *Learning with kernels: support vector machines, regularization, optimization and beyond*. Cambridge: The MIT Press, 2002.

SHI, Y.; EBERHART, R. A modified particle swarm optimizer. In: *Proceedings of the IEEE International Conference on Evolutionary Computation*. 1998. p. 69–73.

SMITH, D. J. *Reliability, maintainability and risk*. 6ed. Burlington: Elsevier, 2001.

SMOLA, A. J.; SCHÖLKOPF, B. A tutorial on support vector regression. *Statistics and Computing*, vol. 14, p. 199–222, 2004.

VAPNIK, V. N. *The nature of statistical learning theory*. 2ed. New York: Springer-Verlag, 2000.

VAPNIK, V. N.; CHAPELLE, O. Bounds on error expectation for support vector machines. *Neural Computation*, v. 12, n. 9, p. 2013–2036, 2000.

WACKERLY, D. D.; III, W. M.; SCHEAFFER, R. L. *Mathematical statistics with applications*. Pacific Grove: Thomson Learning, 2002.

WEIGEND, A. S.; GERSHENFELD, N. A. (Ed.). *Time series prediction: forecasting the future and understanding the past*. Reading: Addison-Wesley, 1994.

WIDODO, A.; YANG, B.-S. Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical Systems and Signal Processing*, vol. 21, p. 2560–2574, 2007.

WU, K.-P.; WANG, S.-D. Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space. *Pattern Recognition*, vol. 42, p. 710–717, 2009.

XU, K.; XIE, M.; TANG, L. C.; HO, S. L. Application of neural networks in forecasting engine systems reliability. *Applied Soft Computing*, vol. 2, p. 255–268, 2003.

YAN, W.; SHAO, H.; WANG, X. Soft sensing modeling based on support vector machines and bayesian model selection. *Computers and Chemical Engineering*, vol. 28, p. 1489–1498, 2004.

ZIO, E.; BROGGI, M.; GOLEA, L.; PEDRONI, N. Failure and reliability predictions by Infinite Response Locally Recurrent Neural Networks. In: *Proceedings of* 5^{th} *European Congress on Computational Methods in Applied Science and Engineering (ECCOMAS)*. Venice, Italy: 2008.