POLITECNICO DI MILANO

Scuola di Ingengeria Industriale e dell'informazione

Dipartimento di Machine Design

Corso di Laurea Magistrale in Ingegneria Meccanica

Damage identification in composite panels based on a Bayesian approach and surrogate models

Relatore: Prof. Claudio Sbarufatti
Co-relatore: Prof. Francesco Cadini

Tesi di Laurea Magistrale di:
Fabio Nichetti Matr. 898993

Anno Accademico 2018-2019

Contents

Figures

Tables

Ringraziamenti

Ringrazio il Professor Sbarufatti ed il Professor Cadini per i loro consigli, per l' aiuto e disponibilitá forniti durante questi mesi.
Ringrazio i miei genitori per la comprensione e supporto ricevuto insieme a mia sorella Sara per essermi sempre stata vicina anche da lontano.
Ringrazio mia nonna perché continua a sorridere e a preoccuparsi per me. Un ricordo particolare per tutti i nonni, per avermi insegnato tanto e che continuano a vegliare su di me ogni giorno.
Un ringraziamento speciale ad Angelo & Tizi sempre pronti a sostenermi e consigliarmi.
Ringrazio i miei amici, compagni di universitá e la mia squadra che mi hanno insegnato quanto sia importante fare gruppo.

Abstract

The aim of this thesis is to develop an integrated damage identification system, able to carry out an automated diagnosis of a delamination induced in a composite structure. The design of a structural health monitoring system implies to manage information coming from a pattern of permanently installed sensors and damage diagnosis means to identify the damage parameters (e.g damage position and extension). In order to achieve this goal an inverse problem must be solved, since it is necessary to get back to the damage parameters starting from the measure of strains. The outputs of the diagnostic phase are needed as input to the prognostic one, in order to retrieve information about the residual useful life of the system. Prognosis is probabilistic in nature, this implies that the diagnostic phase cannot be deterministic but probabilistic as well. Bayesian approaches (such as the Monte Carlo algorithm) have proven to be successful for the inverse problem solution, by evaluating the posterior distribution a damage parameters vector, conditioned on the observations of some signal features dependent on the damage, such as strains. These strains are directly needed to compute the likelihood. Unfortunately, for realistic cases, the evaluation of the likelihood for each sample ideally needs to run a computationally expensive finite element model, making the computational burden not compatible with real-time application. This problem is solved by using surrogate models integrated in a Markov-Chain Monte Carlo algorithm (used as Bayesian inference tool) in the form of Metropolis-Hasting, applied to a damage identification of a composite panel affected by a delamination. An important characteristic of the surrogates employed in this work is that they are 'statistical' surrogate models, in particular artificial neural network committee and Kriging are used to predict the strains as a function of the delamination position and dimension. These surrogate models other than substituting the computationally-demanding numerical model allowing a faster likelihood evaluation, contain information about the uncertainty of the prediction since they are statistical. From a series of finite element simulation it is possible to simulate various damage examples, than this cases are learnt by surrogate models. Finally the Markov-Chain Monte Carlo algorithm is implemented to estimate the probability density function of the damage parameters.

Chapter 1

Introduction

Real-time structural health monitoring of structures is becoming more and more an ambitious goal especially in fields which require an high level safety and reliability such as the aerospace one. Real-time monitoring means creating a system able to asses autonomously the structural integrity of the system in both damage identification (position and dimension of the damage) and evaluation of the residual useful life (RUL). The former is know also as structural health monitoring (SHM) and it is related to the diagnosis of the damage, while the latter is the prognostic health monitoring (PHM). In order to obtain a good prognosis of the structure life it is indispensable correctly to detect the damage, so it is the combination of the two which allows real-time maintenance: through a permanently installed network of sensors it is possible to evaluate the structure health and make predictions about its residual useful life (RUL). In the aerospace industry composite materials have become largely used for their high performances with respect to other conventional materials employed, such as aluminum alloys. In particular, their lightness is essential to reduce the weight of aircrafts with subsequent reduction of fuel consumption. However the employment of composites rises some maintenance and strength issues. As a matter of fact composite structures exhibits really high sensitivity to localized impact damage, which could bring to a delamination (a detachment) of the plies involved in the impact. This damage, unlike for metals, is barely visible, so very expensive Non-Destructive Test must be run, increasing the maintenance time and costs. As a result, industries have been putting significant efforts in order to create a system able to automatically recognize whether in the structureis undergoing a damage or not. Furthermore, a decay of the strength of the laminate, in particular in compression [1], is present if subjected to a damage, which makes the structure unsafe with possible catastrophic failure.
At the basis of the real-time monitoring there is a sensor network installed permanently into the system, then the data retrieved by these sensors are integrated into a Statistical Bayesian Model able to give information about

the presence of a damage, in particular to detect its position and size. This part of the process is called diagnosis. Once the damage is localized in its position and size, then these data are used as input into a damage propagation model to compute the residual useful life of the component.
The direct consequence of this procedure is the ability to pass from a time-based maintenance to a condition-based maintenance, with a consistent reduction of related costs.
In the last decades SHM and PHM have been subjected to an extensive study by the scientific community, in particular two main approaches have been developed: the data-driven approach and the model-based approach [2]. The first method is based on information provided by real cases measurements, then through a pattern recognition or machine learning tool it is possible to build a statistical model, based also as an example on data gathered from an experimental campaign. The second methodology is a physics-based model: a mathematical or virtual model (such as a finite element model) which is able to well represent the behaviour of the system is run and returns data such as strains, which depend on the damage simulated, so that the goal of the model-based approach is to search for the state of the system that is more compatible with the sensors response.
To intuitively explain how SHM works, let us suppose to have a pattern of sensors, such as strain gauges or optical fiber, on a panel subjected to a load, and a damage occurs on it, for example a crack or delamination. Once the damage appears, so that the strain measured by the sensors will change based on the entity of the damage (position and dimension), the aim is to find the position of the damage by reading only the strains. By using our intuition a possible solution would be to run a series of FE simulations and extrapolate the strain at each sensor position, until the result is similar to the one signed by the gauges. To run these simulations we could follow a defined pattern or just run them randomly, but in both cases it is very time consuming, and the necessity of reducing the number simulations arises immediately, since we want to keep the computational efforts low.
In order to do that, a Markov Chain Monte Carlo Metropolis Hasting algorithm is introduced, which is able to start from a random point and then it is able to 'walk' toward the damaged location, minimizing the simulation needed. The usage of the Markov Chain Monte Carlo Metropolis Hasting algorithm is justified also by another need: in fact so far we have treated the diagnostic/prognostic problem as deterministic (some studies can be found here [3]), whereas in order to have a more useful estimation of the RUL it is more appropriate to move from a deterministic field to a probabilistic one, which means that the damage parameters (position an size) must be expressed with their probability density function (*pdf*), ergo not only with its most probable value but also with its uncertainty quantification (UQ). This aspect is well recognized from the scientific community [4]. To solve the

UQ problem Bayesian inference (BI) is employed under the form of Markov Chain Monte Carlo. Then, these results can be projected into a prognostic tool in order to obtain the RUL, but this topic is not covered by this thesis. BI tools have been subjected to an extended campaign of studies by the scientific community, some indications can be found here [5]. BI tools exploit the concept of likelihood function, which will be clearer later, but for now let us limit ourselves to say that BI compares the observed damage feature of the structure (such as strains from sensors) with the simulation result of the Markov Chain sample drawn. This means that for each sample of the chain, a FEA must be run. Of course it is quite understandable that if number the number simulations is high (thousands of samples), the computational time will be very high, not matching the real-time monitoring requirements.
From here arises the necessity to find a method, or tool, able to drastically reduce the computational time required to know the possible system response at each iteration of the chain. Previously used in the reliability field [6], [7], surrogate modeling constitute the solution to this problem, because it is able to learn the system response for a given input, increasing the computational speed. In this context machine learning in particular is a well recognized method for a fast surrogate modeling. There are many machine learning tools available nowadays such as: Linear Regression, Support Vector Machine, Artificial Neural Network (ANN), Gaussian Processes (GP) and many others, enabling a faster likelihood estimation, thus offering a probabilistic prediction together with a computational efficiency. In this work a comparison in terms of performances between ANN (the most used tool nowadays) and GP will be explored. So the final procedure arrangement starts by training the surrogate model off-line, ANN or GP, where the data needed for training it are gathered from a series of finite element analysis, in which several delaminations are simulated. Thus it is possible to learn the relation between strains registered by sensors and the damage location. This model will be then used in a Markov Chain Monte Carlo Metropolis Hasting algorithm to quickly compute the response of the system to the damage's location hypothesized (likelihood computing), and finally identify the most probable location and size of the delamination.
In this work real-time structural health monitoring is applied to automatically recognize a delamination in a composite panel, cause by a low velocity impact. The thesis is organized as follow: in Chapter 2 an overview of the state of the art is presented, then in Chapters 3, 4, 5 the methodology used to solve the inverse problem are explained. Finally in Chapter 6 all the concept introduced previously are applied to the case studied.

Chapter 2

State of the art

2.1 Introduction

The aim of this work is to provide an advanced diagnostic tool suitable for mechanical components made of composite materials, which represent one of the novelty aspect together with the type of implemented algorithm structure, based on surrogate modeling and Markov chain Monte Carlo algorithm. The final goal is to localize the position of the component's damage together with the identification of its dimension: a panel made of carbon fiber reinforced plastic subjected to a circular delamination. Two different machine learning algorithms have been studied to achieve this goal: Artificial Neural Network (ANN) and Kriging (or Gaussian Processes (GP) with integrated basis function) have been implemented and their performances compared. Regarding SHM, important innovations have been achieved during the last years, although there is a partial lack of works which regard composites and GP. This Chapter is organized as follow: 2.2 recalls the usage of composite materials in the transport industry and addresses SHM as an innovative maintenance framework with its new challenges. 2.3 explores the importance of the uncertainty problem quantification for a diagnostic tool, since its output is the input for the following prognostic phase, and finally 2.4 shows the main methodologies used till now to address the problem of real-time SHM.

2.2 An overview of structural health monitoring

Over the last decays SHM has been a persuaded goal especially from the scientific community and the aerospace industry, this can be noticed by several publications such as [8]. As already mentioned SHM allows to avoid expensive non destructive testing (NDT), and it is performed while the structure is working (on-line). Permanently installed sensors provide continuous data to the control unit, which is then capable to recognize the presence of a damage. Worth of notice is that embedded sensors are an active part of the

structure, so their presence must be taken into account during the design phase. The main advantage that distinguishes time-based monitoring from conditioned-based monitoring (based on NDT) is that, if occurs any phenomenon able to potentially cause a dangerous damage in the structure, it can be detected at that very moment, without waiting the next inspection, avoiding potential growth of the induced damage, independently on whether it was a crack or delamination. Furthermore, the inspection based on expensive NDT, occurs only in the situation in which sensors signal an anomaly in the structure, with consequence saving of money and time with respect to time-based maintenance, which may bring to unnecessary inspections. The sensors evolve together with SHM, and over the time new sensors able to guarantee lightness and low signal to noise ratio have been developed. A comprehensive study of types of sensors and potential application of structural health monitoring can be found in [9], nevertheless a quick review of the technologies available today are shown. Each sensor works on a basis of a physical phenomenon such as deformation, vibration, electromagnetism, temperature, light etc. Among those, the most used in the past is the classic electrical stain gauge, in which a strain gauge is bonded to the structure's surface and there is a variation of the electrical resistance of the component as the structure is deformed by a load. Although strain gauges are very common for industry application, optical fiber sensors have been taking their place, and are particular suited for composites, thus being more and more studied by the scientific community [10]. Their main advantage is that a multitude of sensors can be placed along the single fiber, reducing the complexity of the electrical system [9], furthermore being less sensitive to electrical noise.
Moving on, let's suppose to have a series of sensors permanently installed inside the system, the main idea behind SHM is that any phenomenon that modifies the operating condition of the structure, induces a variation of the strain registered by the sensors. Than these information are processed in order to be able to understand the healthy state of the structure, enabling the identification of the damage in its position and dimension. This information will be than the input for the prognostic phase (not covered in this thesis). As mentioned the real challenge is to be able to correctly understand the information provided by sensors, in order to extract quantitative information about the damage location and dimension. To this aim two main approaches are found by the scientific community: Data-driven and Model-based approach.

2.2.1 Data-driven approach

A delamination in a panel could be situated anywhere, this implies that the possible values of strain registered might be infinite, so we need to 'learn' the possible response of the strain gauges, as a damage is induced inside the system. Pattern recognition and machine learning algorithms are able

to learn the relation between the damage position and dimension, and the strain response of sensors, through a training process. Some examples of machine learning algorithms used are provided by [11]. The machine learning algorithms employed in this work are Artificial Neural Networks (ANN) and Gaussian Processes (GP).
In order to train correctly the surrogate model a large database is required, in the data driven approach this dataset comes from experiments or real life measurements. The advantage of this method with respect a Model-based one is that to extrapolate the training data there is no need to build a numerical or analytical model, avoiding to introduce further uncertainty in the damage diagnosis [12], but real measurements and experiments are not always available due to unconceivable costs. An example of this method is given by [13].

2.2.2 Model-based approach

The second methodology used to obtain the training database is the model-based strategy. This method implies the availability of a mathematical model (or numerical), able to well represent the behaviour of the system. In particular the model must be able to determine the strains at sensor's locations, in presence of a damage. Actually if we want to be precise the former model might be included in the model-based approach, where the model is the surrogate model itself. Once the model, or function, is available, it is just matter of finding the damage position which is more likely according to the strain registered by each sensor. To achieve this goal we can minimize the error (or as we shall see maximization of the likelihood) between strain registered and strain simulated by our model. Hence an inverse problem must be solved, because we have to get back to the system state, starting from its output. The main obstacle of this method is the model itself since the function and its variables must be well defined. Nevertheless literature offers many example such as [14]. The perfect mixing up of the two methods is represented by finite element modeling (FEM) matched up with surrogate modeling, in which through FEM a large database can be created off-line by running a series of simulations in which the position and dimension of the delaminations is changed. Than through a machine learning algorithm (among the ones already mentioned), the relation between strains and damage properties is mapped.

2.3 Uncertainty quantification in SHM

As already mention we are interested in exploiting Bayesian inference (BI) to account uncertainties related to not only the model, but also the measurements, in order to reach an acceptable result in the prognostic phase. Civil field gives numerous example about this topic such as [15], but a lack

studies regarding uncertainty quantification and classification with composite material (in particular carbon fiber reinforced plastic). In fact most of the studies with this material regard the technology used to detect the damage, such as frequency response method [16], carbon nanotubes [17] or optic fiber, just to cite a few examples. Although Bayesian neural networks are exploited by [11] and Kriging models are used by [18], there is no sign to match them up with a MCMC, to detect the damage, so the employment of this method for composites represent a part of novelty. Furthermore there is an absence in the literature about estimation of the identification performance based on the noise level in the measurement. As a matter of fact in real life situations measurements are always affected by noise and as we shall see based on the noise level, the system may be more or less accurate (and precise) in finding the damage position and dimension. The development of Bayesian inference approaches in conjunction with advanced MCMC methods and surrogate modeling remains relatively limited for model-based SHM applications. [19] gives a good example of the general methodology followed in this work but there, a very simple thin plate cracked component is used, not a CFRP, and a performance analysis, in terms of precision and accuracy, based on the noise level is missing. Finally GP and Kriging still have a limited application in SHM of complex aerospace components or panels, in this thesis a deep understand if its performances with respect to ANN committee has been explored.
Many studies, such as [25], exploit surrogates under a deterministic point of view, but as we turn to a Bayesian framework all types of uncertainties must be taken into account, including the prediction of the surrogate itself. To overcome this issue, in this work, statistical surrogate models are used, in particular ANN committee and GP are both able to provide mean and standard deviation, or a mode and dispersion in the case the distribution is not Gaussian, of the prediction.

2.4 SHM: current studies

As most of the SHM problems, the damage identification implies the solution of an inverse problem, consisting in the inference of damage parameters (e.g. position and dimension) based on the effect they have on the measured feature (such as the strain), from a practical point of view we have to make inference by observing a set of strain measurements. In order to achieve this purpose, machine learning algorithms have been largely used to map the relation between the strain at the sensor location and the damage parameters: [20] showed how a committee of neural networks can effectively detect a crack in an aluminum panel, while [21] used a Kriging surrogate model to detect a crack in a structure, just to show some examples. Although these cases

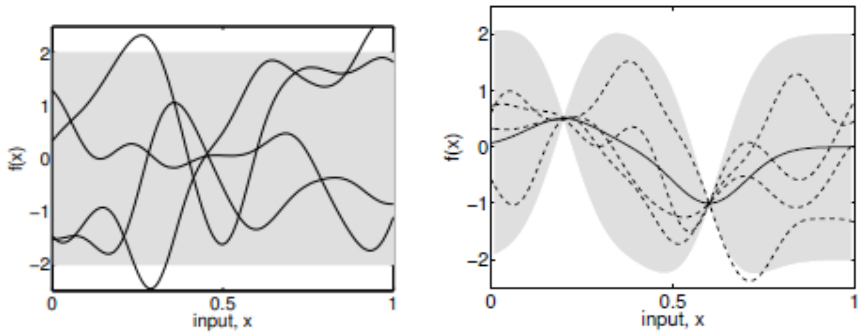
proved that machine learning and pattern recognition can localize a damage in a precise and effective way, the problem related to the uncertainty of the prediction arises. As a matter of fact multiple source of uncertainty are inevitably present in this diagnostic framework, such as discrepancies between the model and the reality and measurement noise [22]. This implies to pass from a deterministic damage identification framework to a probabilistic one, thus providing probability density functions (*pdf*) of the damage parameters.
Bayesian methods provide a statistical framework for the identification of damage in presence of uncertainties: Papadopoulos and Garcia (1998) proposed a Bayesian methodology for comparing the damaged and un-damaged structure state, surrogate based uncertainty [22] have been developed to analyze the effect of the noise in measurements and to deal with discrepancies between the model and the real case.
Among Bayesian frameworks Markov-Chain-Monte-Carlo (MCMC) sampling has been largely used to estimate the posterior probability in the form of the Metropolis-Hasting algorithm (MCMC-MH). Its strength involves the computation of a simple likelihood function in which it is compared the strain field coming from sensors with the one obtained from a model able to well represent the deformation of the system in presence of a damage. However MCMC-MH chain often requires the computation of the likelihood thousands of time, which makes the problem not practical when a closed form (analytical) solution of the link between damage state and observed feature is not available. In this case surrogate models, able to learn the link between the strain at the sensors position and the damage location, allow to evaluate quickly the likelihood, some example of an MCMC-MH algorithm coupled with surrogate model can be found here [23], [24].
Many studies, such as [25], exploits surrogates under a deterministic point of view, but as we turn to a Bayesian framework all types of uncertainties must be taken into account, including the prediction of the surrogate itself. To over come this issue in this work statistical surrogate models are used, in particular ANN committee and GP are both able to provide mean and standard deviation, or a mode and dispersion in the case the distribution is not Gaussian, of the prediction.
This thesis explores the implementation of damage identification system based on a network of strain observations and surrogate models, within a MCMC Bayesian framework, applied to a composite CFRF panel subjected to a delaminaton. Strains from multiple finite element (FE) simulations are used to train the surrogate models off-line, thus enabling a quick likelihood assessment in the MCMC-Metropolis-Hasting (MCMC-MH) algorithm. A comparison between ANN and Kriging surrogate models is explored with a final statistical analysis of the precision and accuracy of the damage identification system.

Chapter 3

Gaussian Process overview

3.1 Introduction

In this section Gaussian processes for regression are labeled. Gaussian processes, or Kriging, are widely used supervised machine learning tools for both regression and classification tasks. Supervised learning means learning an input-output relation from empirical data, called training set, the difference between a regression or classification problem lies in the nature of the output: regression has continuous outputs, while classification has discrete outputs. An example of classification is digital image recognition ([26] gives an example of GP employed for this purpose), while an example of regression is mapping strain from a damage position, which is done in this work. The input is usually denoted by $\mathbf{x}$, while the target $\mathbf{y}$. The vector $\mathbf{x}$ is in general made up of more variables and so it can be seen as a vector nxm, where n is the number of observations and m is number of input variables. The dataset D containing n observations can be written as: D=$\{(\mathbf{x}_i,\mathbf{y}_i)\ i$=1,....,$n\}$. Assuming that the training of the algorithm has been performed, in order to achieve our purpose, damage detection, we wish to make a prediction for a new input point $\mathbf{x}^*$, not included in the training set, thus we have to move from a training set D to a sort of function f able to make prediction for all possible new input values. This function f can be seen, as first step, as a linear combination of other simpler basis functions. Two approaches are available to choose the proper basis function to satisfy the supervised learning. The first is just limited to choose a class of function, such as linear, quadratic or Gaussian, of the input. This approach has an intrinsic problem related to the type of function chosen, in fact it has to well model the target function, furthermore if we try to increase the flexibility of the class of function overfitting could appear. The second considers a prior probability on each possible function, where higher probability are given to functions that represent closely the target. This method has a problem related to the fact that there is an infinite number of possible functions, so that it may seem



(a) prior (b) posterior

Figure 3.1. (a) shows four function drawn from a prior distribution, while (b) shows the posterior distribution after two points have been observed. Dashed lines are four samples from the posterior distribution, the solid one the the mean. The shaded region still represents twice the standard deviation.

to not be possible to do calculations with this infinite set: however GPs are able to address this problem. A Gaussian process, in fact, by definition is a generalization of multivariate Gaussian probability distribution of random variables which defines a Gaussian distribution over functions[27]. In a more practical manner, the relationship we want to learn from a training dataset is not represented by a single function but by a distribution of functions as we shall see. It must be precise that this is a very rude abstraction of the process. Deeper insight GPs for the interested reader may refer to: [28], [29].

3.1.1 Bayesian Modelling

In this Section an intuitive functioning of GPs (second method) is shown. Let's assume we have to solve a simple regression problem, mapping in one dimension from an input variable x to an output *f(x)*. In Figure 3.1(a) some functions from a *prior* distribution (over functions) are drawn. This prior rapresents our expectation of the types of function which may well represent the data, before seeing them. In most of the cases no prior information are available, so the mean of all the functions (which means the mean of the distribution) is set to zero for any x (but the single function has not a zero mean). In addition to the mean, we can also characterize the variability of those functions by expressing the variance in each point, this implies that the functions are Gaussian distributed. In Figure 3.1 the shaded region points out twice the standard deviation (in that very case the prior variance is constant, which means that it has a x independence). Let's suppose now that we have a training set made up two observation: $D=\{(\mathbf{x}_1,\mathbf{y}_1),(\mathbf{x}_2,\mathbf{y}_2)$, and we want to consider only realization of GP passing through these points exactly (figure 3.1(b)). The solid black line represents the mean function of the dis-

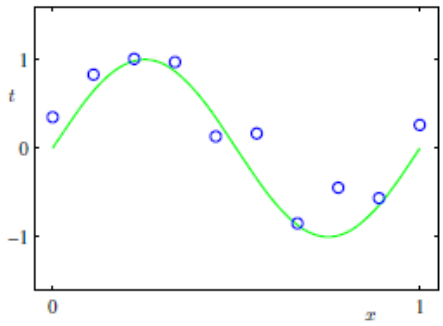


Figure 3.2. The blue points are the observed training points drawn from $sin(2\pi x)$ in green with addition of a noise in the form of $\mu(0, \sigma^2)$, x is the observed input variable and its correspondent target is t.

tribution of functions consistent with D (dashed lines). The most important thing that has to be noticed, is that the standard deviation decreases to zero as we approach the training points, while it goes back to the prior far from those points. If the standard deviation of the Gaussian distribution of functions decreases, it means that the uncertainty related to the prediction of a new input $\mathbf{x}^*$ becomes lower. The solution shown in Figure 3.1(b) is called *posterior* distribution over functions. If more points were added, the mean function would pass through all of them with the standard deviation which goes to zero in the neighborhood of the point region. To be more precise it is not mandatory that the mean function passes exactly through the points, because it is possible (as we shall see) to make it pass just near the dataset by adding an additive term called 'noise'.After this pictorial overview, we are going to introduce GP from a ore mathematical point of view, the Chapter is organized as follow: first an introduction to fitting problems with 3.2, than in 3.3 some basic concepts about statistics are introduced. Linear regression models are explored in 3.4, in section 3.5 we see the linear regression problem under a statistical point of view, this lead directly to Gaussian processes in section 3.6.

3.2 Introduction to polynomial curve fitting

A simple regression problem is presented in this Section, in order to introduce the reader to some important concepts that will be subjected to further development in the next Sections. Imagine that we have observed real values of the input variable x and target, or output, variable t as shown in Figure 3.2, and we wish to make prediction of the value of the variable t for a new input x^*. The available dataset is $\mathbf{x} = (x_1, ..., x_N)^T$ with its target values, $\mathbf{t} = (t_1, ..., t_N)^T$, is shown in Figure 3.2 (blue circles). Those points are

taken from a function $sin(2\pi x)$ and, then a random Gaussian noise $\mu(0, \sigma^2)$ is added. This process is quite similar to what happens in a real case when the measurements of a phenomenon is corrupted are by noise (such as the electrical one). It must be clear that the main objective is not to learn the exact target $\mathbf{t}$, because it is corrupted by the noise, but the real objective is to learn the unnoised underlying function $sin(2\pi x)$. For the moment we consider a very simple curve fitting, by using a polynomial function of the form:

$$y(x,w) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M = \sum_{j=0}^{M} w_j x^j \tag{3.1}$$

where M is the order of the polynomial, and $w_0, ..., w_M$ are scalar parameters that can be group in a weight vector $\mathbf{w}$. This function is a non linear function of x but is a linear function of the unknown vector $\mathbf{w}$, so these types of fitting models are called *linear models*. In order to fit the dataset we have to find the weight vector $\mathbf{w}$. This can be achieved by minimizing an *error function* which measure the misfit between the data points and $y(x, \mathbf{w})$. A simple example of this error function is given by the *sum of squared errors* between the prediction $y(x_n, \mathbf{w})$ at the data points x_n, and the target values t_n, and it is expressed as follows:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 \tag{3.2}$$

The squared is used in order to obtain a positive number. This error can be equal to zero only if $y(x,\mathbf{w})$ passes exactly through each training point (Figure 3.3 illustrates this concept graphically). $\mathbf{w}$ is chosen so that it minimize the error function. Furthermore E is a quadratic function of $\mathbf{w}$, thus its derivative is a linear function of $\mathbf{w}$, which implies that the solution $\mathbf{w}^*$ is unique. The final result will be $y(x, \mathbf{w}^*)$. The only thing we still have to do is to choose

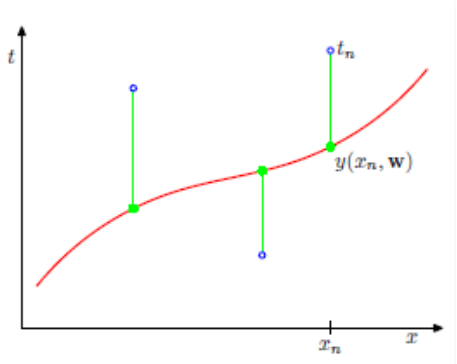


Figure 3.3. The error function is the sum of the square of the length of the green line (mismatch between real observed data and $y(x, \mathbf{w})$.

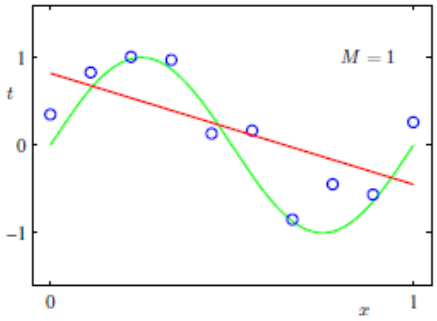


Figure 3.4. Example of linear regression with a polynomial of first order as basis function.

the order M of the polynomial function, which is known as *model selection*. By using our intuition, it is easy to understand that for low values of M such as $M = 1$, the linear model will fit the dataset with a straight line, so the resulting error will be very high, as shown in Figure 3.4. This may lead us to increase the order of the polynomial, but if M increases too much, then we encounter the problem of *over fitting*, shown in Figure 3.5. Actually this problem is strictly related to both the order M of the polynomial function and the number of data points in the training set (e.g if M is higher than the number of the dataset, for sure there will be over fitting). Figure 3.6 shows the difference in the fitting result between a smaller and larger dataset with $M = 9$, we can say that the larger the data set is, the higher is the order of the polynomial we can afford. A way to avoid over fitting is using the *regularization* procedure, which means introducing in the error function a further term proportional to the weights, but since this is not a thesis related to machine learning this topic will not be explored. By now, it is just enough to say that finding weights by minimizing the least square error,

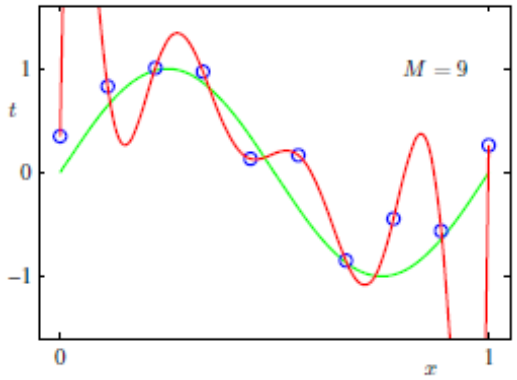


Figure 3.5. Over fitting results in a function passing exactly through data points, but with very high oscillation among them.

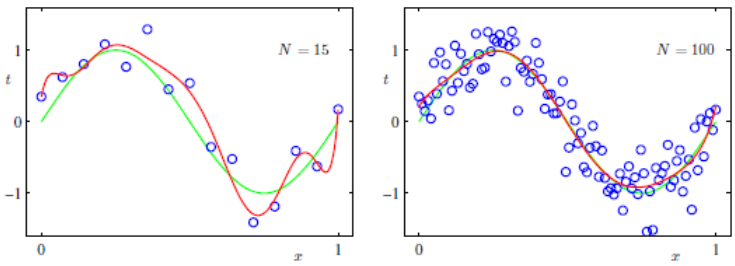


Figure 3.6. Over fitting results in a function passing exactly through data points, but with very high oscillation among them. As we can see as the number of training point passes from $N = 15$ to $N = 100$ the over fitting effect is neutralized.

means reaching the *maximum likelihood* (discussed in the next Chapter). A great advantage of adopting a *Bayesian approach* is that, to avoid overfitting or poor fitting, we do not have to take care about the number of parameters M anymore.

3.3 Introduction to statistics

The aim of this Section is to introduce the reader to some fundamental concepts about statistic and Bayesian mathematics since to understand Bayesian inference and GP these concepts are of fundamental importance.

3.3.1 Probability

In the field of machine learning quantification of uncertainties plays a key role. These can be motivated by noise measurements, limited availability of data or uncertainty due to the model employed (for example a physics based model). Furthermore the uncertainty of the surrogate model itself plays a role for the final performance of the detection system. Probability theory provides a consistent framework to deal with uncertainty. To make probability theory more practical and less mathematical a simple example is considered. Let's consider to have two boxes of two different colors: red and blue, and those boxes are filled with green balls and orange balls, in

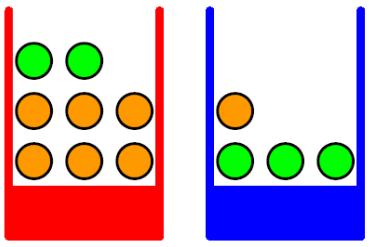


Figure 3.7. In this figure the simple example of the two boxes is shown.

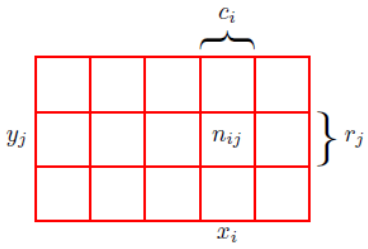


Figure 3.8. A simple example in which two variables are rapresented.

particular, in the red one there will be two green balls and six orange, while in the blue one only one orange ball and three green ones. Now imagine to select randomly one of the boxes, and than to repeat this operation ten times and taking track of how many times we select the red one rather than the blue one. Let's now suppose that we picked the red box 40% of the time, and we picked the blue box 60% of the time, we can identify the color of the boxes with the variable B. This variable can take (in this case) two possible values: r if we pick the red box and b if we pick the blue box. We can also introduce a random variable also for the ball's color: F, this can take either value g (green) or o (orange). The probability that an event can occur is the fraction between the times the event occured out of the total number of trials, thus the probability to pick up the red box 4/10 and the probability to select the blue box is 6/10. The mathematical notation to write what just explained is: $p(B{=}r){=}4/10$ and $p(B{=}b){=}6/10$. Note that the probability must lie in the interval [0,1].
Let us now suppose that every time we picked up a box we also picked up a ball inside it, with statistic we are going to be able to answer questions such as: "what is the probability to pick up a green ball?", or "by knowing we have just picked up an orange ball what is the probability that the box chosen was the blue box?", but before we have to introduce the *sum rule* and *product rule*. To derive these rules, let's consider a more general case shown in 3.8, which involves two random variable X and Y. X can take values x_i where $i{=}1,....,M$, and Y can take values y_j with j=1,...,L. Now we sample N times both variables, we can define n_{ij} as the number of time in which $X{=}x_i$ and $Y{=}y_j$. The probability that X takes the value x_i and Y takes the value y_j is called *joint probability*, and its notation form is:

$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N} \tag{3.3}$$

The probability that X takes the value x_i, independently from the value assumed by Y, is:

$$p(X = x_i) = \frac{c_i}{N} \tag{3.4}$$

where c_i is the overall number of times x_i has been extracted.
By using equations 3.3 and 3.4, it is quite easy to understand that

$$p(X = x_i) = \sum_j p(X = x_i, Y = y_j) \tag{3.5}$$

which known as *sum rule* of probability.
Than by knowing that X assumes the value $X = x_i$, the probability that Y takes the value $Y = y_j$ is written as: $p = (Y = y_j | X = x_i)$ and it is called *conditional probability* of $Y = y_j$ given x_i. The conditional probability just illustrated is given by:

$$p(Y = y_j | X = x_i) = \frac{n_{ij}}{c_i} \tag{3.6}$$

By using 3.3, 3.4, 3.6 we can derive:

$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N} = \frac{n_{ij}}{c_i} * \frac{c_i}{N} = p(Y = y_j | X = x_i) * p(X = x_i) \tag{3.7}$$

this is known as *product rule* of probability. From now on we will use a simplified notation:

$$p = \sum_Y p(X, Y) \qquad \text{(Sum rule)}$$

$$p(X, Y) = p(Y|X)p(X) \qquad \text{(Product rule)}$$

$p(X, Y)$ is the joint probability of X *and* Y. From the product rule and by exploiting the symmetry property $p(X, Y) = p(XY, X)$ we can obtain the following relationship:

$$p(Y|X) = \frac{p(X|Y) * p(Y)}{X} \tag{3.8}$$

which is called *Bayes' theorem*, The term $p(X)$ is often referred as a normalization constant, in order to make the conditional probability of 3.8 equal or lower than one. This theorem as we shall see plays a fundamental role in Gaussian processes (an in achieve learning in general), and its application allows to deal with uncertainties. The next Section will go into the heart of the theorem, explaining the meaning of each term and introducing the reader to Bayesian probability.

3.3.2 Bayes' theorem

Going back to the example of the boxes filled with balls, we can give a better interpretation of the Bayes' theorem. Let's now choose randomly a box, and than pick up a ball. The major information we caught after having picked ten times the random boxes in the previous section, is that the red one was chosen

4/10 times and so we know $p(B)$ which is called *prior probability* because it is the probability available before knowing the ball's color picked up. Once the ball's color is known we can use the Bayes' theorem the compute $p(B|F)$, which is called *posterior probability*, because we can compute it only after having observed the value assumed by F. Let us assume that the color picked is orange and we want to compute the conditional probability $p(B{=}r|F{=}o)$, we use the Bayes' theorem:

$$p(B{=}r|F{=}o) = \frac{p(F{=}o|B{=}r)p(B{=}r)}{p(F{=}o)} \tag{3.9}$$

We already know that $p(B{=}r) = \frac{4}{10}$, to compute the prior the sum rule is exploited:

$$p(F{=}o) = p(F{=}o|B{=}r)p(B{=}r) + p(F{=}o|B{=}b)p(B{=}b) = \frac{9}{20}. \tag{3.10}$$

By using 3.9 is possible to conclude that the posterior probability $p(B{=}r|F{=}o) = \frac{2}{3}$. This result accords also with intuition, since the number of orange balls is higher in the red box and we picked an orange ball, the probability that the box chosen is the red one is higher than the blue one.

3.3.3 Bayesian probability

By now probability has been explained in terms of a sequence of events, but this interpretation is useless for our purposes, in this Section we are going to turn the concept of probability in a *Bayesian* field, in which probability is able to quantify uncertainty. To have an idea of what we are talking about, take as consideration an uncertain event, such as the rate of melting of the polar ice, which is not a repeatable phenomenon as the case of the two boxes. In this example we want not only to quantify the value of the uncertainty related to the melting speed but also make a revision of our believes when new evidences such as new measurements come out. After this pictorial introduction the polynomial curve fitting shown in Section 3.2 is rediscussed under a Bayesian point of view, in particular we quantify the uncertainty related to the computation of $\mathbf{w}$. Remembering that in section 3.3.2 we discovered that the observation of the picked fruit (prior probability) can heavily affect the posterior probability, we can use the same approach to make inference of $\mathbf{w}$. In that case the Bayes' theorem takes the form:

$$p(\mathbf{w}|D) = \frac{p(D|w)p(\mathbf{w})}{p(D)}. \tag{3.11}$$

where we can incorporate our assumption about $\mathbf{w}$ through the prior probability $p(\mathbf{w})$, the effect of the observed data set $D = t_1, ..., t_N$is expressed by the likelihood function $p(D|w)$ and it expresses how probable the data set

is for a given vector of parametres $\mathbf{w}$. Recalling that $p(D)$ is a normalization factor in order to have a posterior equal or lower than one, the Bayes' theorem can be seen as:

$$posterior \propto likelihood \times prior \tag{3.12}$$

The denominator can found by integrating both sides of the equation in the following way:

$$p(D) = \int p(D|w)p(w)d\mathbf{w} \tag{3.13}$$

We have already mentioned in Section 3.2 that the sum of squared errors used as estimator to compute the weight vector $\mathbf{w}$ is actually the maximization of the likelihood function. The *maximum likelihood* is an estimator in which $\mathbf{w}$ is set to maximize the likelihood function $p(D|w)$. In machine learning literature the negative logarithm of the likelihood function is called *error function*. Since the error function is monotonically decreasing, maximize the likelihood means minimize the error (or sum of squared error). In the next Section we will apply Bayesian inference also to the prediction on the target variable t for a new input x^*, in order to find the posterior probability distribution $p(t|x^*)$.

3.4 Linear regression models

In Section 3.2 we have already discussed about regression but in that case we used a polynomial of regressors, which however belongs to a broader class called linear regression models. In these models we can take a general linear combination of *nonlinear functions* of the input called *basis functions*, recalling that this models are linear with respect to the weights and nonlinear with respect to the input variable. In the following equation a simple linear regression of both input and weights is presented:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + ... + w_D x_D \tag{3.14}$$

where D is the number of input variables. The model just presented has significant limitation, in fact being a linear function of the input variable x, it can only well fit linear input-output relationship. To overcome this issue we consider a linear combination of nonlinear function of the input variable, as shown below:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) \tag{3.15}$$

where $\phi_j(\mathbf{x})$ are known as *basis function*, M is the total number of parameter and w_0 is called *bias* since it provides an offset. In order to simplify the the

notation a further basis function $\phi_0(\mathbf{x}) = 1$ is introduced, in this way we can write:

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) \tag{3.16}$$

where $\mathbf{w} = (w_0, ..., w_{M-1})^T$ and $\phi = (\phi_0, ..., \phi_{M-1})^T$, so they are both column vectors containing all the weights and basis functions. By using nonlinear basis functions $y(\mathbf{x},\mathbf{w})$ can be nonlinear function of the input too. It is worth noticing that in section 3.2 we had just one input variable x, and in that case $\phi(x) = x^j$.
There are many common basis functions, such as the 'Gaussian' basis function:

$$\phi(x)_j = exp\{-\frac{(x - \mu_j)}{2s^2}\} \tag{3.17}$$

where μ_j govern the mean of the function in the input space, and s is the spacial scale.
Another may be the sigmoid basis function:

$$\phi(x)_j = \sigma\frac{(x - \mu_j)}{s} \tag{3.18}$$

or the logistic sigmoid:

$$\sigma(a) = \frac{1}{1 + exp(-a)} \tag{3.19}$$

In the previous sections we mentioned that minimizing a sum-of-squared error function means maximizing the likelihood under the assumption that the model is affected by a Gaussian noise, it is now time to explain this concept more in detail.
Let's assume now that our observed target variable t is affected by a Gaussian noise ξ with zero mean and precision (inverse variance $\beta^{-1} = \sigma^2$) β and

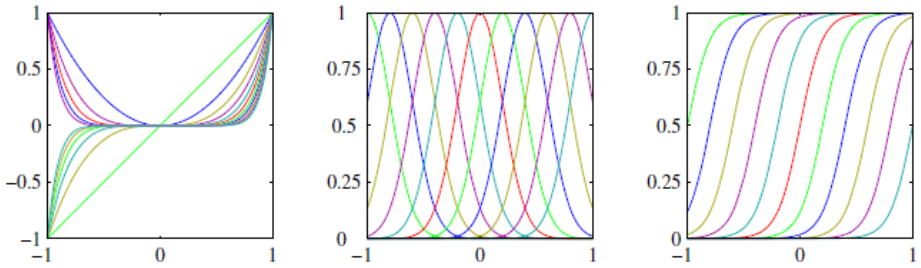


Figure 3.9. On the right polynomials basis functions, Gaussian basis function in the middle while on the left there is sigmoid basis function

$y(\mathbf{x}, \mathbf{w})$ is the uncorrupted function we want to learn, we can write:

$$t = y(\mathbf{x}, \mathbf{w}) + \xi \tag{3.20}$$

Since the noise has a Gaussian distribution, also the target variable t has a Gaussian distribution with mean $y(\mathbf{x},\mathbf{w})$ and precision β, thus we can write:

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = N(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}) \tag{3.21}$$

So we are interested in modeling the distribution of the output variables. Now consider the dataset of inputs $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ with its corresponding target values $\mathbf{t} = \{t_1, ..., t_N\}$, making the assumption that the data points are drawn independently from the distribution 3.21, we obtain the following expression for the likelihood function:

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} N(t_n|\mathbf{w}^T\phi(\mathbf{x}_n), \beta^{-1}) \tag{3.22}$$

It is now convenient to write likelihood function in the logarithm form:

$$\begin{aligned} p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \sum_{n=1}^{N} lnN(t_n|\mathbf{w}^T\phi(x_n), \beta^{-1}) = \\ \frac{N}{2}ln\beta - \frac{N}{2}ln(2\pi) - \frac{\beta}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2 \end{aligned} \tag{3.23}$$

To determine the optimal polynomial coefficients $\mathbf{w}^*$ we can use the maximum likelihood criterion, which means maximizing 3.23 with respect to $\mathbf{w}$. In this way, the first two terms can be neglected, since there is no dependency on $\mathbf{w}$, furthermore we substitute the coefficient $\frac{\beta}{2}$ with $\frac{1}{2}$ since this scaling

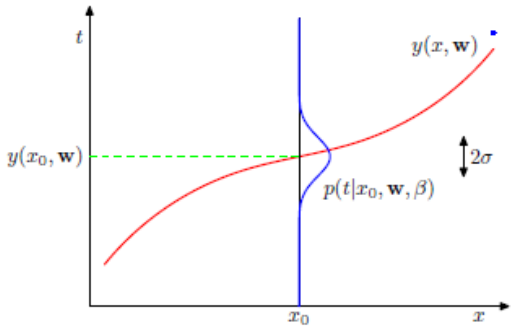


Figure 3.10. Graphical distribution of the variable t for a given x (one input variable only) in which the mean is given by $y(x,\mathbf{w})$ and the standard deviation is given by the precision β, for which we have $\beta^{-1} = \sigma^2$

factor does not alter the position of the maximum with respect to $\mathbf{w}$. Finally instead of maximizing the log likelihood, we can minimize the negative log likelihood, thus as far as we are looking for finding the vector $\mathbf{w}$ maximizing the likelihood means minimizing the *sum-of-square error function*.
Setting at zero the gradient with respect to $\mathbf{w}$ we obtain:

$$0 = \sum_{n=1}^{N} t_n \phi(\mathbf{x}_n)^T - \mathbf{w}^T (\sum_{n=1}^{N} \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T). \tag{3.24}$$

Solving with respect to $\mathbf{w}$ we obtain:

$$\mathbf{w}^* = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t} \tag{3.25}$$

Φ is a $N \times M$ matrix called *design matrix* composed by:

$$\Phi = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1).... & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2).... & \phi_{M-1}(\mathbf{x}_2) \\ . & . & . \\ . & . & . \\ . & . & . \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N).... & \phi_{M-1}(\mathbf{x}_N) \end{bmatrix} \tag{3.26}$$

Once the vector $\mathbf{w}^*$ has been found we can proceed and determine the precision parameter β, the maximization leads to the following result:

$$\frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n=1}^{N} \{t_n - \mathbf{w}_{ML}^T \phi(\mathbf{x}_n)\}^2 \tag{3.27}$$

3.5 Bayesian linear regression

We have seen that the model complexity such as the number of basis function, or better the number of weights, is led by the size of the of the data set because just maximizing the likelihood may lead to over-fitting. So deciding the model complexity plays a crucial role for the final result, and this cannot be reached by simply maximizing the likelihood function. Treating the linear regression under a Bayesian point of view can avoid the over-fitting problem of maximum likelihood, and will lead also to an automatic method for determining the model complexity using the training data alone.

3.5.1 Parameter distribution

The discussion begins with introducing a prior probability distribution over the parameters $\mathbf{w}$, and β is treated as a known constant. Since the likelihood function $p(\mathbf{t}|\mathbf{w})$ in 3.22 is Gaussian, the conjugate prior over $\mathbf{w}$ is Gaussian as well:

$$p(\mathbf{w}) = N(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0) \tag{3.28}$$

where $\mathbf{m}_0$ is the mean and $\mathbf{S}_0$ is the covariance. The product between likelihood and the prior can give us the posterior distribution. Since the prior distribution is Gaussian, the posterior will be Gaussian as well in the form:

$$p(\mathbf{w}|\mathbf{t}) = N(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N) \tag{3.29}$$

where:

$$\mathbf{m}_N = \mathbf{S}_N(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\Phi^T\mathbf{t}) \tag{3.30}$$

$$\mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta\Phi^T\Phi \tag{3.31}$$

To simplify the case under assumption, we can assume that the prior distribution over the weight vector is a zero-mean Gaussian, with precision parameter α so that:

$$p(\mathbf{w}|\alpha) = N(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \tag{3.32}$$

and the posterior distribution over $\mathbf{w}$ is given by:

$$\mathbf{m}_N = \mathbf{S}_N(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\Phi^T\mathbf{t}) \tag{3.33}$$

$$\mathbf{S}_N^{-1} = \alpha\mathbf{I} + \beta\Phi^T\Phi \tag{3.34}$$

It is now time to compute the log of the posterior distribution, given by the sum of the log likelihood function and the log of the prior:

$$lnp(\mathbf{w}|\mathbf{t}) = -\frac{\beta}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2 - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + cons \tag{3.35}$$

The maximization of the log of the posterior distribution with respect to $\mathbf{w}$ is equivalent to minimize the sum-of-square error function with the addition of a quadratic term.

3.5.2 Predictive distribution

In real case application we are not interested in computing the value of $\mathbf{w}$ but we want to make prediction of t for a new input value x^*, this implies to evaluate the *predictive distribution* :

$$p(t|\mathbf{t}, \alpha, \beta) = \int p(t|\mathbf{w}, \beta)p(\mathbf{w}|\mathbf{t}, \alpha, \beta)d\mathbf{w} \tag{3.36}$$

where $\mathbf{t}$ is the target vector of the training set. It can be demonstrated that the predictive distribution is:

$$p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = N(t|\mathbf{m}_N^T\phi(\mathbf{x}), \sigma_N^2(\mathbf{x})) \tag{3.37}$$

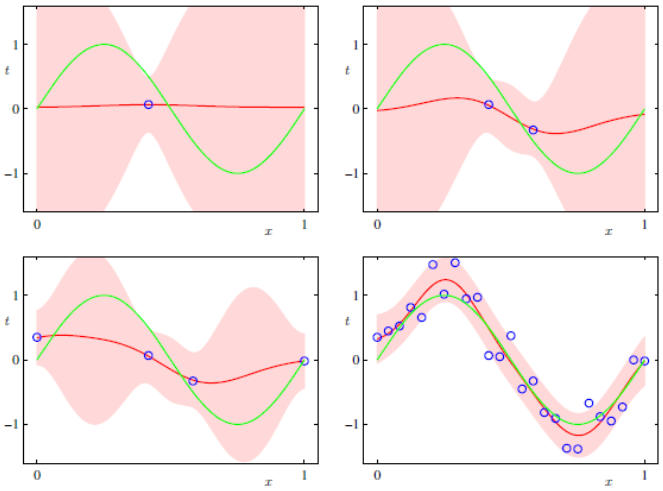


Figure 3.11. Example of predictive distributions as the number of training points increases. Note that the standard deviation increases a lot far from the training points, while reduces near them.

It means that the prediction of the target variable for a new input is a Gaussian distribution characterized by a mean and a variance. The variance $\sigma_N^2(\mathbf{x})$ is given by:

$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \phi(x)^T \mathbf{S}_N \phi\mathbf{x} \tag{3.38}$$

where $\frac{1}{\beta}$ represents the noise on the data, while the second part of the equation represent the uncertainty associated with the parameter $\mathbf{w}$. Figure 3.11 shows the predictive distribution of a Bayesian linear regression in which data are drawn from a sinusoid. Data are fitted using a Gaussian basis function. For each plot, the blue circles are the data points, the red line is the mean of the Gaussian predictive distribution, the green one is the sinusoidal function, and finally the red shaded region is one standard deviation. We can than draw samples from the posterior distribution over $\mathbf{w}$, and than plot $y(x, \mathbf{w})$ as shown in the Figure 3.12.

3.5.3 Equivalent kernel

By substituting eq. 3.30 in equation 3.16, we can derive the predictive mean in the following form:

$$y(\mathbf{x}, \mathbf{m}_N) = \mathbf{m}_N^T \phi\mathbf{x} = \beta\phi(\mathbf{x})^T \mathbf{S}_N \Phi^T \mathbf{t} = \sum_{n=1}^{N} \beta\phi(\mathbf{x})^T \mathbf{S}_N \phi(\mathbf{x}_n) t_n \tag{3.39}$$

where $\mathbf{S}_N$ is defined in 3.31. The mean of the predictive distribution at a certain $\mathbf{x}$ can be seen as a linear combination of the target t_n and the so

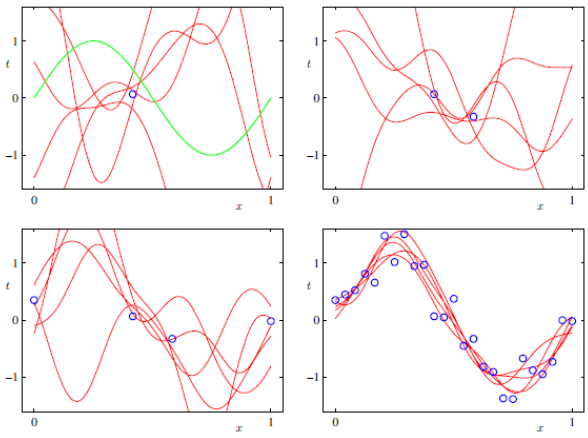


Figure 3.12. Plots of functions $y(($x),$\mathbf{w}$).

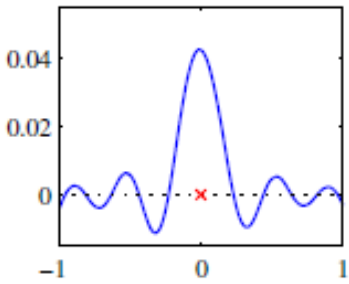


Figure 3.13. Example of a kernel function contained from a sigmoid basis function. Note that it is a localized function even if $\mathbf{x}'$ is non local.

called *equivalent kernel* or *smoother matrix* defined as:

$$k(\mathbf{x}, \mathbf{x}') = \beta\phi(\mathbf{x})^T \mathbf{S}_N \phi(\mathbf{x}') \tag{3.40}$$

Finally the the predictive mean can be computed as:

$$y(\mathbf{x}, \mathbf{m}_N) = \sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}') t_n \tag{3.41}$$

Note that the equivalent kernel is a function of the input training set $\mathbf{x}_n$, in figure 3.13 a kernel function is computed from sigmoid basis function. One of the most popular covariance function is the *squared exponential covariance function*:

$$k(\mathbf{x}, \mathbf{x}') = exp(-\frac{1}{2}|\mathbf{x} - \mathbf{x}'|^2) \tag{3.42}$$

It can be shown that the squared exponential covariance function corresponds to a Bayesian linear regression with an infinite number of basis function.

3.6 Gaussian processes

In Section 3.4 we saw a linear regression in the form of $y(\mathbf{x},\mathbf{w}) = \mathbf{w}^T\phi(\mathbf{x})$. We have seen that a prior distribution over $\mathbf{w}$ induces a prior distribution over functions $y(\mathbf{x}, \mathbf{w})$, then given a training data set, we than evaluate the posterior distribution of both $\mathbf{w}$ and $y(\mathbf{x}, \mathbf{w})$, which than implies predictive distribution $p(t|\mathbf{x})$ for a new input $\mathbf{x}$. We have also discovered kernel functions, so from Gaussian point of view we define a prior over functions directly and not over weights.

3.6.1 Linear regression revisited

The discussion starts with a quick review of a linear combination in the form:

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\phi(\mathbf{x}) \tag{3.43}$$

where $\mathbf{x}$ is the input vector and $\mathbf{w}$ is the M-dimensional weight vector. We then put a prior distribution over $\mathbf{w}$ in an isotropic Gaussian form:

$$p(\mathbf{w}) = N(\mathbf{w}|0, \alpha^{-1}\mathbf{I}) \tag{3.44}$$

where α^{-1} is the inverse variance (or called precision), that could be a non-diagonal matrix $\sum_p$. The prior probability on $\mathbf{w}$ induces a probability over $y(\mathbf{x}, \mathbf{w})$. We want now to compute the posterior probability using the information provided by the training set $\mathbf{x}_1, ..., \mathbf{x}_N$ and $\mathbf{y}_1, ..., \mathbf{y}_N$; we denote with vector $\mathbf{Y}$ the vector containing all target observed. Through eq. 3.43 we can compute that vector as:

$$\mathbf{Y} = \Phi\mathbf{w} \tag{3.45}$$

The design matrix Φ is a matrix with elements $\Phi_{nk} = \phi_k(\mathbf{x}_n)$. $\mathbf{Y}$ is a linear combination of Gaussian distributed variables $\mathbf{w}$ so it is Gaussian as well, thus it is completed described by its mean and covariance (since it is a multiple input variable), by:

$$\mathbf{E}[\mathbf{Y}] = \Phi\mathbf{E}[\mathbf{w}] = 0 \tag{3.46}$$

$$cov[\mathbf{Y}] = \mathbf{E}[\mathbf{Y}\mathbf{Y}^T] = \Phi\mathbf{E}[\mathbf{w}\mathbf{w}^T]\Phi^T = \frac{1}{\alpha}\Phi\Phi^T = \mathbf{K}(\mathbf{x}, \mathbf{x}) \tag{3.47}$$

where $\mathbf{K}$ is called Gram matrix or *covariance matrix* with elements:

$$K_{nm}(\mathbf{x}, \mathbf{x}) = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha}\phi(\mathbf{x}_n^T)\phi(\mathbf{x}_m) \tag{3.48}$$

where $k(\mathbf{x}_n, \mathbf{x}_m)$ is the kernel function. In general there is no prior knowledge available about $\mathbf{w}$ so the mean is set to zero, to define completely the Gaussian process we need to determine the covariance matrix by computing the covariance between any two values of $\mathbf{x}$ through the kernel function

$k(\mathbf{x}_n, \mathbf{x}_m)$, this function can be computed either by using the formula 3.48 or can be directly on our choice, a typical choice of this function is the *squared exponential*:

$$k(\mathbf{x}_n, \mathbf{x}_m) = exp\Big(-\frac{1}{2}|\mathbf{x}_n, \mathbf{x}_m|^2 \Big) \tag{3.49}$$

It must be precise that if we choose the covariance function by our own, the function $y(\mathbf{x}, \mathbf{w})$ will not be a function of the weights $\mathbf{w}$ anymore, but it can be written as a Gaussian process with mean and covariance function as already defined:

$$y(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}_n, \mathbf{x}_m)) \tag{3.50}$$

3.6.2 Gaussian processes for regression

To apply Gaussian processes to deal with regression, we have to take into account noise in the observed data (as we have already seen in the Bayesian linear regression):

$$t_n = y_n + \epsilon_n \tag{3.51}$$

where $y_n = y(\mathbf{x}_n)$, while ϵ_n is a random noise variable independent from the observation and normally distributed as follow:

$$\epsilon \sim N(0, \sigma_n^2) \tag{3.52}$$

where σ_n^2 is the variance of the noise. Since it has a Gaussian distribution, it induces a Gaussian distribution over the observations:

$$p(t_n|y_n) = N(t_n|y_n, \sigma_n^2) \tag{3.53}$$

Since we have multiple observations, the joint distribution of the target values $\mathbf{t} = (t_1, ..., t_N)^T$ conditioned on the values $\mathbf{y} = (y_1, ..., y_N)^T$ is given by an isotropic Gaussian distribution:

$$p(\mathbf{t}|\mathbf{y}) = N((\mathbf{t}|\mathbf{y}, \sigma_n^2\mathbf{I}_N) \tag{3.54}$$

where $\mathbf{I}_N$ is the identity $N \times N$. By definition of a Gaussian process the marginal distribution of the observation $p(\mathbf{y})$ is Gaussian with mean zero and covariance matrix $\mathbf{K}$

$$p(\mathbf{y}) = N(0, \mathbf{K}(\mathbf{x}, \mathbf{x})) \tag{3.55}$$

The marginal distribution $p(\mathbf{t})$ but it must take into account the independent Gaussian noise source, so probability takes the zero as mean but with a change in the covariance function:

$$\mathbf{t} \sim N(0, \mathbf{k}(\mathbf{x}, \mathbf{x}) + \sigma_n^2 I) \tag{3.56}$$

where each element of the covariance matrix is in the form:

$$cov(t_n, t_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \sigma_n^2 \delta_{nm} \tag{3.57}$$

where δ_{nm} is equal to zero if $n = m$ since the noise on the observations is independent. So it can be written in the following form:

$$cov(\mathbf{t}) = \mathbf{k}(\mathbf{x},\mathbf{x}) + \sigma_n^2 I \tag{3.58}$$

Thus the squared exponential covariance function can be written as:

$$k_t(x_n, x_m) = \sigma_f^2 exp(-\frac{1}{2l}(x_n - x_m)^2) + \sigma_n^2 \delta_{nm} \tag{3.59}$$

The three unknown parameters are called *hyperparameters*, and as we shall see, each one plays a significant role in learning the function wanted. Anyhow, by now it is just enough to say that l is called *correlation length* or *length scale*, σ_f^2 is a *scale factor* of the Gaussian function, and σ_n^2 is the *variance of the error* (which is the dispersion of our observation around the function $y(\mathbf{x})$). Since now we have discussed about the joint distribution of the observed data set, but actually our goal in regression is to make prediction for a new input give a set of training points. Let's suppose that $\mathbf{t}_N = (t_1, ..., t_N)^T$ are the target of the corresponding input $\mathbf{x}_1, ..., \mathbf{x}_N$ our purpose is predict the target variable t_* for a new input $\mathbf{x}_*$, which means that we have to evaluate the predictive distribution $p(t_{N+1}|\mathbf{x}_{N+1})$ and it can we written in the following form:

$$\Phi = \begin{bmatrix} \mathbf{t} \\ y_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} \mathbf{K}(\mathbf{x},\mathbf{x}) + \sigma_n^2 I & \mathbf{K}(\mathbf{x},\mathbf{x}_*) \\ \mathbf{K}(\mathbf{x}_*,\mathbf{x}) & \mathbf{K}(\mathbf{x}_*,\mathbf{x}_*) \end{bmatrix}\right) \tag{3.60}$$

To simplify the notation we can introduce a new matrix:

$$C_{N+1} = \begin{bmatrix} \mathbf{K}(\mathbf{x},\mathbf{x}) + \sigma_n^2 I & \mathbf{K}(\mathbf{x},\mathbf{x}_*) \\ \mathbf{K}(\mathbf{x}_*,\mathbf{x}) & \mathbf{K}(\mathbf{x}_*,\mathbf{x}_*) \end{bmatrix} \tag{3.61}$$

We can assume that we have only one test point and simplify the notation such as: $\mathbf{K} = \mathbf{K}(\mathbf{x},\mathbf{x})$, $\mathbf{k}_* = \mathbf{K}(\mathbf{x},\mathbf{x}_*) = \mathbf{K}(\mathbf{x}_*,\mathbf{x})$ and $\mathbf{k}(\mathbf{x}_*,\mathbf{x}_*) = \mathbf{K}(\mathbf{x}_*,\mathbf{x}_*)$ Finally we are ready to compute the mean and variance of the prediction.

$$m(t_*) = \mathbf{k}_* \mathbf{K}^{-1} \mathbf{t} \tag{3.62}$$

$$\sigma^2(t_*) = \mathbf{k}(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_* \mathbf{K}^{-1} \mathbf{k}_* \tag{3.63}$$

A very important thing to be noticed is that the prediction is based on an inversion of a $N \times N$ matrix where N is dimension of the data set. It means that to avoid numerical problems the size of the data point cannot be

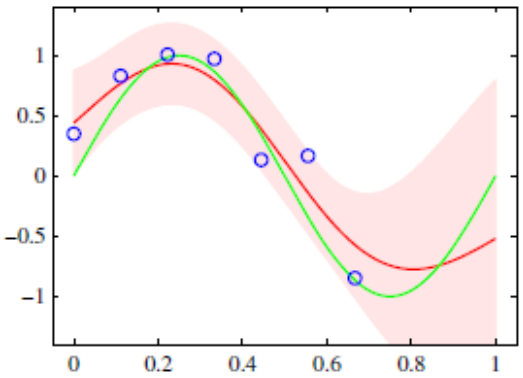


Figure 3.14. GP regression of a sinusoid. The green curve is the sinusoidal function, in blue there re the training points, the red line is mean of the Gaussian Process, and finally the shaded region is plus minus two standard deviation.

very large, usually the maximum number of points employed is 3000. A way to avoid this inversion is using the Cholesky decomposition, in this way we obtain a faster an numerically more stable process. Please note that $\sigma^2(t_*)$ is the uncertainty of the prediction for a new input point, which is different from the value of the hyperparameter σ_n^2 related to the noise the observation. We have already seen that $\sigma^2(t_*)$ is lower near the training points, and becomes larger far from them or we are out from the data set as shown in Figure 3.14.

3.6.3 Learning the hyperparametres

From a practical point of view the learning process depends upon the values the hyperparameters θ take. To learn them we have to evaluate the likelihood function $p(\mathbf{t}|\theta)$ and maximize it. The log likelihood function for a Gaussian process regression model is easily evaluated using the form for a multivariate Gaussian distribution.

$$lnp(\mathbf{t}|\theta) = -\frac{1}{2}ln|\mathbf{K}| - \frac{1}{2}\mathbf{k}_*^T\mathbf{K}^{-1}\mathbf{k}_* - \frac{N}{2}ln(2\pi) \tag{3.64}$$

We than compute the derivative with respect to a given hyperparameter:

$$\frac{\partial}{\partial\theta_t}lnp(\mathbf{t}|\theta) = \frac{1}{2}\mathbf{Tr}(\mathbf{K}\frac{\partial\mathbf{K}}{\partial\theta_t}) + \frac{1}{2}\mathbf{k}_*^T\mathbf{K}\frac{\partial\mathbf{K}}{\partial\theta_t}\mathbf{K}^{-1}\mathbf{k}_* \tag{3.65}$$

3.6.4 Incorporating explicit basis function

It common to consider a GP with zero mean function, this is not a big a limitation since the mean of the posterior is not zero. However there are advantages (mainly from a computational point of view) to include a mean function. Incorporating explicit basis function is a way to incorporate a non-zero mean over function. This can be achieved by still using a zero mean

Gaussian Process, but adding also a *basis function*, in this case $y(\mathbf{x})$ takes the following form:

$$y(\mathbf{x}) = f(\mathbf{x}) + \mathbf{h}(\mathbf{x})^T\beta \tag{3.66}$$

where $f(\mathbf{x}) \sim GP(0, k(\mathbf{x}, \mathbf{x}))$, $\mathbf{h}(\mathbf{x})$ are a set of basis functions, usually it is a polynomial vector such as $\mathbf{h}(\mathbf{x}) = (1, x, x^2, ..)$ and β is inferred from data. The basic concept is that we do linear a regression (which gives the mean) in which the residuals are learnt through a GP.

3.6.5 Effect of hyperparmeters: GP in practice

As already mentioned the squared-exponential covariance function has the following form:

$$k(\mathbf{x}_n, \mathbf{x}_m) = \sigma_f^2(-\frac{1}{2l}(\mathbf{x}_n - \mathbf{x}_m)^2) + \sigma_n^2\delta_{nm} \tag{3.67}$$

has three free parameters to be learnt or defined by our own: l is the length scale, σ_f^2 is the process variance or scale factor, and σ_n^2 is the noise variance. We must precise that the log-likelihood function is very complex since it depends upon three parameters, this implies that the minimum solution of its derivative is not unique, as a matter of fact there are multiple local minima, and each one gives different results in terms of learning. In Figure 3.15 it is shown a parabola in which in which are drawn 15 points smeared with a Gaussian noise with standard deviation $\sigma_n^2 = 50000$. The blue line represents the actual parabola, the blue points are the data points, the continuous red line is the prediction of the Gaussian process and finally the dotted red line is plus-minus two standard deviation predicted. In this case, the length scale l found is 806, the process variance is $8.24 * 10^6$ and the estimated noise variance $\sigma_n^2 = 3.8 * 10^4$ which is actually very close to $5 * 10^4$, if we put and higher number of training points, such as 30, its value would become very alike to the real one. Let's now change the length scale and see what happens: Figure 3.16 shows the results when the length scale becomes much larger than the previous one. In that very case l was set to 9000, the other two parameter values are found by maximizing the log-likelihood, $\sigma_f^2 = 10$ and $\sigma_n^2 = 7.12 * 10^5$. As we can see the prediction approximates to the regression only and the noise variance is much higher than the real one. When the length scale increases the general behaviour is 'smoother' and it approximates to the basis function (in this case it was linear), this type of result can be explained by the fact that since the length is very high, one point is correlated with many others, so it is a sort of generalized mean. In general, when the computed σ_n^2 is much higher than the real one, the prediction passes very far from the data points (even with). On the other hand, if we set the length scale to a smaller value, we obtain a greater flexibility of the prediction with a consistent reduction of the noise standard deviation, in fact in Figure 3.17 l has been reduced to 60, the remaining other two parameters were set

to maximize the likelihood again, ($\sigma_f^2 = 6.59 * 10^5$ and $\sigma_n^2 = 7.37 * 10^3$) the noise variance turns out to be an order of magnitude lower than the optimal one. This implies that the prediction passes very near the training points and its trend has an higher variability, this behaviour can be compared to over-fitting in linear regression and neural networks, in which the error with respect the data set is quite low, but the result in terms of generalization is very inefficient. In Figure 3.18 the length scale has been further reduced down to $l = 0.001$. This means that each prediction point is correlated with no one of the training data, with the result that the overall prediction follows the linear regression, except for the trained points, in which it goes very near them. In this case $\sigma_f^2 = 6.64 * 10^5$, $\sigma_n^2 = 6.67 * 10^5$.

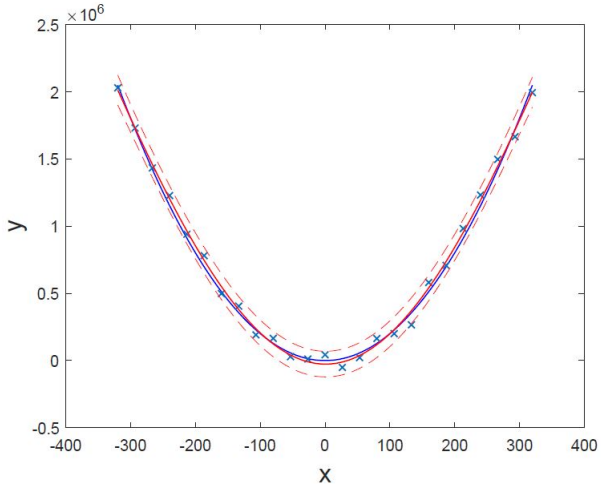


Figure 3.15. A Gaussian process prediction.

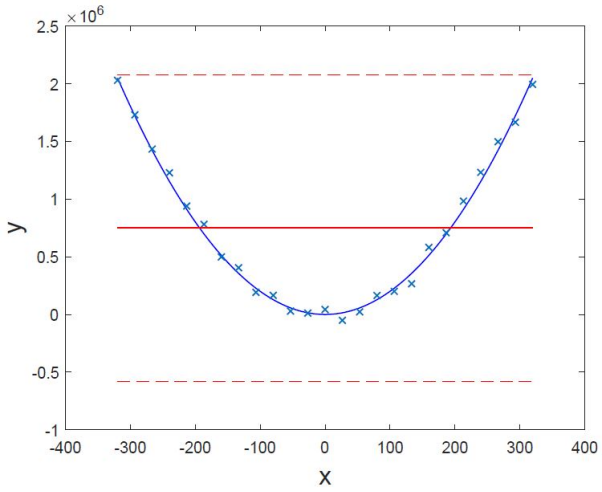


Figure 3.16. Effect of having an high length scale.

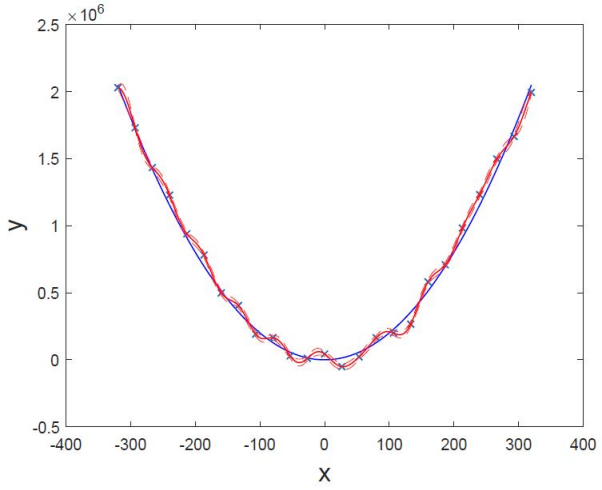


Figure 3.17. Effect of having a low length scale.

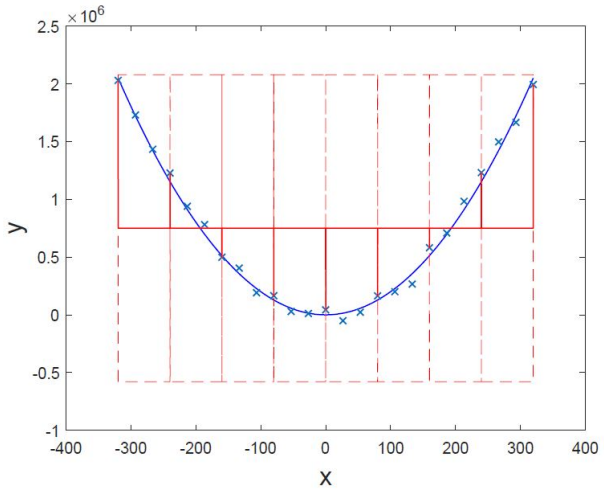


Figure 3.18. Effect of having a very low length scale.

Chapter 4

Artificial Neural Network

4.1 Introduction

Artifical Neural Network (ANN) is probably the most spread technique for machine learning problems because of its mathematical simplicity, and capacity to deal with complex problems unmanageable by simpler techniques such as polynomial regression. Born with the aim to reply the biological brain and nervous human system in terms of structure and learning capabilities, research about ANN became very active since 1940_s. The aim of this Chapter is to introduce the reader to ANN theory in order to make him aware of the training process and ANN structure meaning, not all topics will be treated since the literate is full of book and references, if the reader is interested in going deeper about some topics he can refer to [30].
The Chapter is organized as follow: first the working principle of a single neuron is shown in 4.2, then these concepts are extended to a more complex artificial neural network structure 4.3. In 4.4 the learning process is shown, and finally the advantages of neural networks committee are explored in 4.5.

4.2 The artificial neuron: single perceptron

As already mention ANN is a bio-inspired algorithm. Human brain is composed by a huge number of single units called neurons interconnected with each others as shown in Figure 4.1. Each neuron is a sort of black box which receives inputs, elaborates them, and then it provides an output. This output is an information entering in the system which will be then processed. A single neuron does not work alone but this procedure is run in parallel with all other neurons. For neurons the activation signal is in the form of electrical stimulation, and it returns a signal as well, whose intensity depends upon the level it was activated. Human's learning activity goes through the continuous stimulation of neurons, by means of continuously repetition of inputs, there is a strengthening of neuron connection, making the network more and

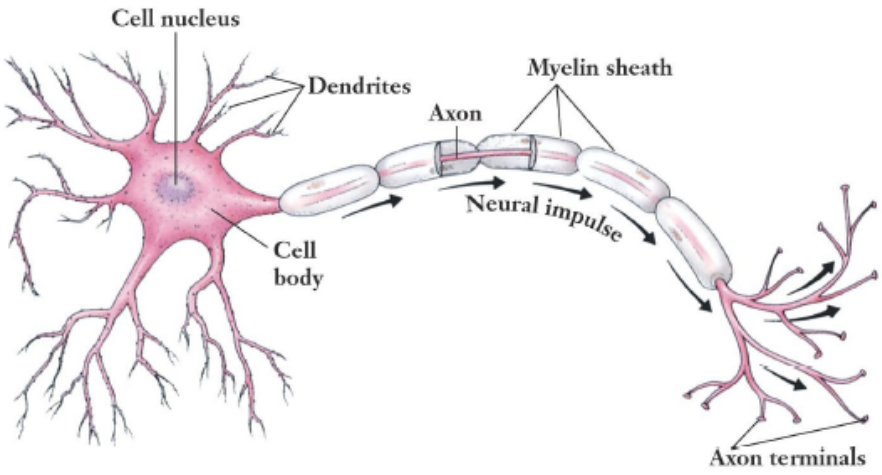


Figure 4.1. Rapresentation of a biological neuron.

more capable to correctly respond for a given input. From a practical point of view, when we approach a routine the first time, probably we will not be able to do that task correctly, but if repeat that task many many time we will able to carry out that job in the right way.
The biological neuron (Fig. 4.2.a) structure is constituted by a cell, where the biochemical reactions occur,and synapses, which are connections among neurons. The cell body sums all the electrical impulses coming from all other connected neurons. If the summation exceeds a certain threshold, the neuron will be activated, sending an electrical impulse outwards, thus stimulating other neurons. Through a continuous excitement of neurons synapses become stronger. A first explanation of the learning process was given by Hebb (1949): when a cell A excites a cell B in a repetitive and persistent manner, a change in metabolism or growth in one or both cells happens, such that the effectiveness of A in stimulating B is increased with respect to all other connected cells. In a more practical way, all neurons are interconnected among each other, they can send and receive electrical impulses. If a neuron receives impulses over a certain threshold it is activated and sends a new electrical input. Similarly, the mathematical model of a neuron (Fig. 4.2.b) consists in a computing cell, which receives information (s_i) from other neurons, sums them and returns an informative impulse (S_j) which is modulated through an activation function $h_j(.)$ and then passed to a consecutive computing unit. The training, or programming, for artificial neural networks consists in a presentation of a sequence of stimulus-response pairs so that the network can learn the appropriate relationship by reinforcing some of its internal connections. In practice, a weight w_{ji} is assigned to each incoming dendrite (each one transferring a signal s_i) and it is optimized based on training data

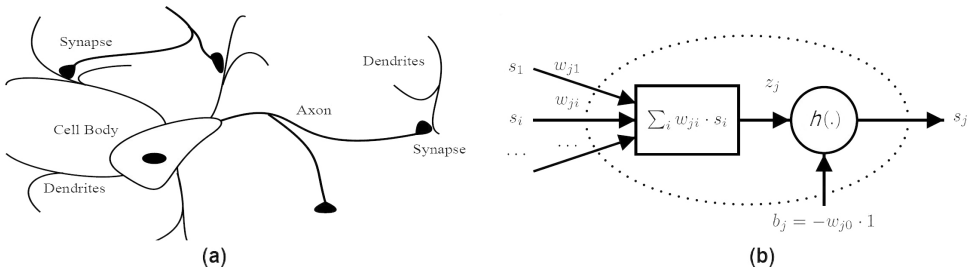


Figure 4.2. Structure for a (a) biological and (b) artificial neuron.

samples. The following parameters might be identified inside the j^{th} neuron model presented inside Fig. 4.2.b:

- The *synaptic weights* or simply weights w_{ji}, associated to the i^{th} incoming link for the j^{th} neuron.
- The *activation value* $z_j = \sum_{i=1}^{N^i} w_{ji} s_i$.
- The bias (*activation threshold*) $b_j = -w_{j0} \cdot 1$
- The *activation function* $h_j(.)$, as will be explained later.

In the case we have only one neuron $j = 1$, while i is equal to the number of synapses connected with it (or number of input from a mathematical point of view). Again j is the neuron index while i is the synapse index. The output of the j^{th} neuron can thus be written as follows:

$$S_j = h_j \left(\sum_{i=1}^{N^i} w_{ji} s_i - b_j \right) = h_j \left(\sum_{i=0}^{N^i} w_{ji} s_i \right) \tag{4.1}$$

being N^i the number of inputs contributing to the calculation of the activation value. It should be noted that the bias is not associated with the signal, it is a fixed constant in order to allow an offset of the excitation z_j. The activation function h_j will be better explained in the next section, by now it is just enough to say that it depends upon the type of problem, whether regression or classification, and upon the network structure, the hidden layer will have a different activation function with respect the output one. For the moment it is important to say that it must introduce a non-linearity in the regression or classification scheme (since we want to model non linear problem) and must be differentiable, an example is the sigmoid function (in the reference given there are many other types of activation functions):

$$S_j = \frac{1}{1 + e^{-z_j}} \tag{4.2}$$

An Artificial Neural Network (ANN) is an ensemble of many elementary artificial neurons (perceptron), as the one presented above. The general idea of a neural network is to represent a non-linear function mapping between a set of input variables into a set of output variables. Artificial Neural Networks are algorithms able to learn from experience (either experimental or virtual, the latter being the one adopted in this thesis), which means that after having see a certain number of cases they are able to elaborate answers in the case of new inputs, exactly like human being. The main advantage of ANN algorithms is that no mathematical or physical modeling effort is needed to evaluate the functions underlying the phenomenon to be described. Furthermore, the system is able to perform an automatic parameter adjustment in the training phase based on available input-output data coming from a dataset obtained either from FE models or from joint FE and experimental simulations. The drawback is that ANNs are difficult to interpret in terms of physical processes and they sometimes require extensive input-output data for proper training, increasing with the number of parameters involved, the extent of non-linearity and the complexity of the approximated functions. For this reason, the ANN may not be used as a "black box", as its success depends much on the knowledge of the problem domain that can be incorporated into the design and training of the neural network.
Many different Artificial Neural Network topologies exist in literature, suitable for different kind of application. The following classification can be done, according to the learning paradigm:

- *Supervised learning*: the algorithm receives input patterns $x_1, x_2, \ldots, x_n$ as well as the associated outputs $y_1, y_2, \ldots, y_n$, and through a training and learning process it produces the correct output given a new input is provided.
- *Unsupervised learning*: it exploits regularities in the input data to build a representation that can be used for reasoning or prediction (no output target data are used during training).
- *Reinforcement learning*: producing actions $a_1, a_2, \ldots, a_n$ which affect the environment, and receiving rewards $r_1, r_2, \ldots, r_n$, it learn to act in a way that maximizes rewards in the long term.

The focus inside this thesis is on *supervised learning* since from numerical simulations a large amount of data regarding input-output damage information is available.

4.3 The feed-forward multi-layer perceptron (MLP)

The ANN structure which is most often used is the so-called *Multi-Layer-Perceptron* (MLP), in which neurons are organized in layers (Fig. 4.3). In

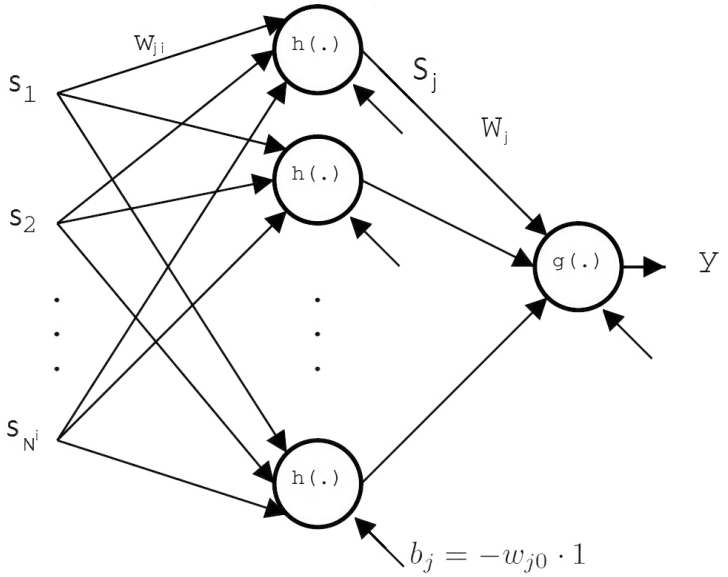


Figure 4.3. Structure for multi-layer perceptron.

particular, a layer is a set of neurons and three types of layer can be identified:

- *Input layer*: neurons that receive as input the data to be processed and distribute the input signals to the first available layer of computing neurons
- *Hidden layer(s)*: intermediate neurons that process data from other neurons and provide input to a consecutive layer. They do not communicate with the outside world.
- *Output layer*: neurons that provide the final output of the network

In a *feed-forward* network topology the signals pass from the *input layer*, progress forward through the *hidden layers* and finally emerge from the *output layer*, feed-forward means the information can go only toward one direction, so every neuron can receive information only from the previous layer unlike the *feed-back* network such as recurrent neural networks. Feed-forward ANN are easy and fast to train, but they have intrinsic limits, for example they are not able to deal dynamic inputs such as time series. Fig. 4.3 shows that each node in a layer is connected through some weights to each node of the consecutive layer. Moreover, each node (except those belonging to the input layer) is also connected to a bias node by means of a special threshold weight (w_{j0}). At each computational node inside the hidden layer, the activation value is calculated as in Eq. 4.3, taking into account also the bias

weight s_0 and N^i inputs to the computational node:

$$z_j = \sum_{i=0}^{N^i} w_{ji} s_i \tag{4.3}$$

Then, the ability of an ANN to learn depends on the choice of the activation function ($h(\cdot)$ and $g(\cdot)$ in Fig. 4.3) for the hidden and output layer respectively. Various choices for the activation function are possible, depending on the problem to be modeled, as well as the considered layer:

1. *hidden layers*: if the problem to be modeled with ANN is non-linear (as most of the times), the choice usually falls upon non linear functions such as the *hyperbolic tangent function* (Eq. 4.4) or the *sigmoid function* (Eq. 4.5), respectively providing an output in the interval $[-1, 1]$ and $[0, 1]$.

$$S_j = \frac{e^{z_j} - e^{-z_j}}{e^{z_j} + e^{-z_j}} \tag{4.4}$$

$$S_j = \frac{1}{1 + e^{-z_j}} \tag{4.5}$$

2. *output layer*: the activation value can be expressed as in Eq. 4.6 according to the nomenclature used inside Fig. 4.2 and Fig. 4.3:

$$Z = \sum_{j=0}^{N^{hl}} W_j S_j \tag{4.6}$$

where N^{hl} is the number of hidden neurons belonging to the hidden layer. Three forms of activation function are usually considered to generate the output values y, depending on the problem under analysis:

- For regression problems, an appropriate choice is the *linear* function of the form:

$$y = Z \tag{4.7}$$

- For classification problems, a *sigmoidal* or a *softmax* activation function can be successfully applied, as in Eq. 4.8 and Eq. 4.9 respectively.

$$y = \frac{1}{1 + e^{-Z}} \tag{4.8}$$

$$y_k = \frac{e^{-Z_k}}{\sum_{k=1}^{N^o} e^{-Z_k}} \tag{4.9}$$

  Eq. 4.9 is mostly recommended in the case one has to infer over two or more mutually exclusive classes, being N^o the number of output nodes for the considered structure (nevertheless if one single output node is used as in Fig. 4.3, 4.8 is the correct choice).

Finally, the ANN output y in Fig. 4.3 can be expressed through Eq. 4.10:

$$y = g\left(\sum_{j=0}^{N^{hl}} W_j \cdot h\left(\sum_{i=0}^{N^i} w_{ji} s_i\right)\right) \tag{4.10}$$

where $h(\cdot)$ and $g(\cdot)$ are the activation functions for the hidden and output layers respectively, W_j is the synapse weight connecting the j^{th} hidden neuron to the output node, w_{ji} is the synapse weight connecting the i^{th} input node to the j^{th} hidden neuron, N^i and N^{hl} are the number of input and hidden nodes respectively. The goal is now to approximate a target function t given a finite number of observations. This is done during the *training* process.

4.4 Learning in multi-layer perceptron

The aim of *supervised training* is to compute weights and biases in order to achieve a good generalization (which means that the network is able to respond not only to the training points but also to new inputs) of the data, this step can be interpreted as the strengthening process of synapses in human nervous system. In particular, at each step during training, a set of inputs is passed forward through the network and a trial output is obtained, and then compared with the desired known output. If a significant error is found, it is passed backward through the network and is used to adjust the synapse weight in order to minimize the error. This procedure is called *error back-propagation*. Thus, the first thing to be considered is the definition of an error function to be minimized during training. In particular, the ANN learning can be associated to the problem of *Maximum Likelihood Estimation* or *Sum of squared error minimization*, which are the same thing as shown in 3. Focusing on *regression problems*, suppose to have a set of $t_1, t_2, \ldots, t_N$ independent identically distributed samples, which are output observations for the process to be modeled through the ANN. Observations can be thought as *normally distributed*, centered in y (the output of the ANN, dependent on the unknown weight variables) with standard deviation σ (also unknown). The likelihood L of the observation, with respect to the ANN model, can be defined as follows:

$$L(w) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t_n - y_n)^2}{2\sigma^2}} \tag{4.11}$$

The problem now turns in finding weights that maximize the likelihood parameter in Eq. 4.11. It already shown that the optimal weights are those which minimize the cost function (the error function E):

$$\arg\max_w L(w) = \arg\min_w \sum_{n=1}^{N} (t_n - y_n)^2 = \arg\min_w E \tag{4.12}$$

4.4.1 The error back-propagation

Error back-propagation is a *gradient descent* optimization method, it is used to iteratively minimize the network error E by adjusting its weight parameters w.

$$\begin{cases} w^{n+1} = w^n + \Delta w \\ \Delta w = -\eta \cdot \dfrac{\partial E}{\partial w} \end{cases} \tag{4.13}$$

One can express the back-propagation update rule for the W_j as follows, being k the iteration index:

$$W_j^{k+1} = W_j^k + 2\eta \sum_{n=1}^{N} \left(t_n - g\left(Z\right)\right) \cdot \left(-g'\left(Z\right)\right) \cdot S_j \tag{4.14}$$

Nevertheless, also the synapse weight connecting the i^{th} input to the j^{th} hidden neuron has to be updated at each iteration during training. The gradient of the error function with respect to the identified weight $\frac{\partial E}{\partial w_{ji}}$ is the following:

$$\begin{aligned} \frac{\partial E}{\partial w_{ji}} &= \sum_{n=1}^{N} 2\left(t_n - g\left(Z\right)\right) \cdot \left(-g'\left(Z\right)\right) \cdot \frac{\partial Z}{\partial w_{ji}} \\ &= \sum_{n=1}^{N} 2\left(t_n - g\left(Z\right)\right) \cdot \left(-g'\left(Z\right)\right) \cdot W_j \cdot \frac{\partial S_j}{\partial w_{ji}} \\ &= \sum_{n=1}^{N} 2\left(t_n - g\left(Z\right)\right) \cdot \left(-g'\left(Z\right)\right) \cdot W_j \cdot h'\left(z_j\right) \cdot \frac{\partial z_j}{\partial w_{ji}} \\ &= \sum_{n=1}^{N} 2\left(t_n - g\left(Z\right)\right) \cdot \left(-g'\left(Z\right)\right) \cdot W_j \cdot h'\left(z_j\right) \cdot s_i \end{aligned} \tag{4.15}$$

In the same way, the procedure to obtain the updating rule for the weight w_{ji} can be synthesize:

$$w_{ji}^{k+1} = w_{ji}^k + 2\eta \sum_{n=1}^{N} \left(t_n - g\left(Z\right)\right) \cdot \left(-g'\left(Z\right)\right) \cdot W_j \cdot h'\left(z_j\right) \cdot s_i \tag{4.16}$$

In general terms, the updating rule for a synapse weight connecting i^{th} neuron with j^{th} neuron in two consecutive layers can be summarized as follows:

$$w_{ji}^{k+1} = w_{ji}^k + \eta \delta_j s_i \tag{4.17}$$

Where δ_j is the error in the output from the j^{th} node of the hidden layer. This error is not known a priori and it is constructed from the errors that can be easily calculated at the output layer $\sum_{n=1}^{N} \left(t_n - g\left(Z\right)\right)$.

4.4.2 Issues in ANN learning

Tough gradient descent methodology for weight optimization is rather a simple process, many problem could arise during ANN training, among which:

- Selection of the best ANN structure
- Convergence of the learning process
- Generalization and overfitting
- Existence of local optima

4.4.3 ANN structure definition

The MLP is capable of approximating a function with arbitrary accuracy even if it only possesses one single hidden layer (as in Fig. 4.3). However no guidelines for the a priori definition of the network complexity for a given function are available. As a consequence, one has to try different ANN structures (one or more hidden layers with varying number of hidden neurons), with increasing complexity, thus selecting the best one, which is the one that minimizes the error with respect to the validation set (refer to Section 4.4.5). Nevertheless, for the specific case reported inside this thesis it has been decided to use one single hidden layer, however optimizing the number of neurons to obtain the best performances of the network.

4.4.4 The momentum coefficient

The learning coefficient drives the convergence speed of the learning process. If η is too small convergence may take too time. On the other hand if η is too large, the learning process will be rapid but there may be the risk that parameters may diverge, as shown in Fig. 4.4. The introduction of a *momentum coefficient* α can mitigate this problem, which is an additional inertia term which depends on the weight modification at previous iteration step, as shown in Eq. 4.18:

$$w_{ji}^{k+1} = w_{ji}^{k} + \eta \delta_j s_i + \alpha \Delta w_{ji}^{k} \tag{4.18}$$

4.4.5 Generalization and overfitting

Generalization is the capability of an ANN to learn the significant features of a proposed dataset, without learning characteristic feature. By recalling the example of polynomial curve fitting, in Figure 4.5 are shown the three sin cases in which we have a poor fitting (or excess of generalization), a good fitting and finally overfitting. Overfitting problem is not only characteristic of polynomial curve fitting but also of ANN since to obtain weights there is a minimization of an error function. In order to avoid overfitting three

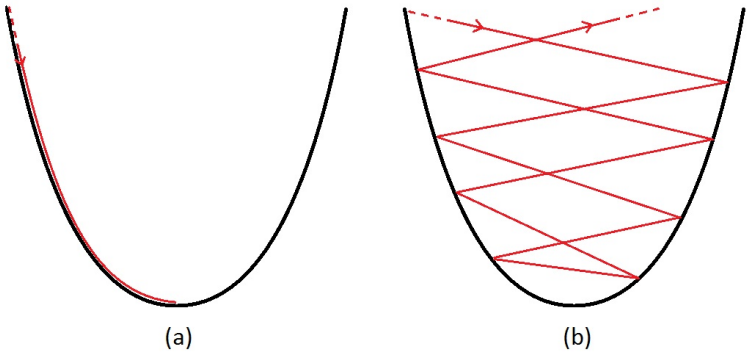


Figure 4.4. Schematic representation of convergence to minimum error (a) and divergence (b) during the weight optimization problem. The adoption of a low learning rate allows for smaller weight step, thus slower converence, While the second figure shows that divergence may occur.

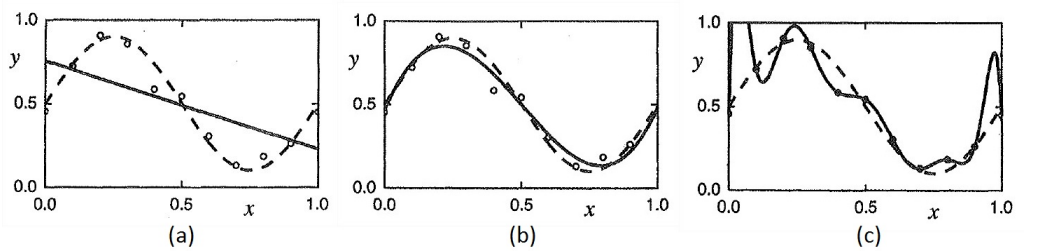


Figure 4.5. Fitting of a dataset generated from a sinusoidal function with additive noise. (a) A linear curve provides a poor fitting. (b) A cubic curve provides the best fitting. (c) A 10^{th} order polynomial over-fits data.

main methodologies are commonly used: *Early stopping*, *training with noise* and *regularization*. Since only one of these techniques are used in this thesis, only *Early stopping* is presented and explained. *Early stopping*: as explained above, the training of ANNs corresponds to an iterative reduction of the error function defined by comparing the ANN output with given set of weights with respect to training data.It is common practice to divide the training dataset in at least two parts:

- *Training set*: data used for weight optimization.
- *Validation set*: independent data not involved in the weight optimization but used to avoid overfitting.

During the training procedure the error is measured also on the validation set, as shown in Figure 4.6. In the first part of the training process both training and validation error decrease but after a certain number of the training set continues to decrease but the validation one starts increasing, which means that overfitting is starting.

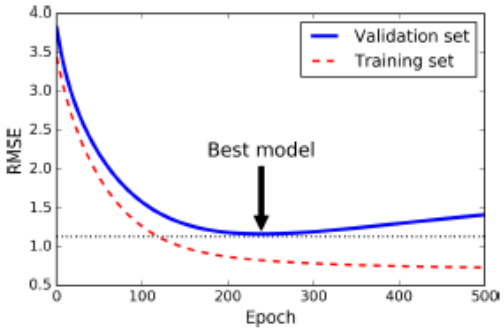


Figure 4.6. Error trend of the validation set during the training process. As we can see after a certain number of iteration the error starts to grow, which means that overfitting is taking place, thus the learning process must be stopped.

4.5 Uniqueness of solution and ANN committees

The problem related to error minimization is that the error function is very complex and so it includes many local minimum, the ideal situation is to reach the global minimum. In order to '*escape*' from local minimum *randomized algorithms* add an amount of energy at each iteration, this energy decreases over the number of iterations, in order to reach convergence.
Multiple restarts with different weights initialization is another tool. From a practical point of view we train the network multiple times with a defined training and validation sets which remain the same for each training. For each training the weights are chained randomly in order to explore different local optima, finally we take the network with the lower root-mean-square error.
In order to achieve a better performance than *multiple restarts* which is just one single isolated neural network, [30] proved that ANN *committee* reaches a lower error. The concept behind ANN committee is that multiple ANN models can be trained, also with different architecture such as different activation function, number of layers, number of neurons and different local minimum, using the training and validation sets, and once each network has been trained, the prediction for a new input value will be just the average of the predictions of each individual ANN model. This procedure is done because, as we are going to see, the error of the committee is always equal or lower than the average associated with the single neural network, on the other hand the computational effort increases. Suppose to have a set of L trained models $y_i(\mathbf{X})$ where $i = 1, .., L$. $h(\mathbf{X})$ is the true regression function we want to learn. The function $y_i(\mathbf{X})$ can be written as the desired function with an additional error:

$$y_i(\mathbf{X}) = h(\mathbf{X}) + \epsilon_i(\mathbf{X}) \tag{4.19}$$

The sum-of-square error for model y_i can be written as :

$$E_i = \xi[y_i(\mathbf{X}) - h(\mathbf{X})^2] = \xi[\epsilon_i^2] \tag{4.20}$$

where ξ is the expectation, which is the integration over $\mathbf{X}$ weighted by the probbility density of $\mathbf{X}$ as follow:

$$\xi[\epsilon_i^2] = \int \epsilon_i^2(\mathbf{X}) p(\mathbf{X}) \tag{4.21}$$

from 4.20 the average error of the single neural networks is given by:

$$E_{AV} = \frac{1}{L} \sum_{i=1}^{L} E_i = \frac{1}{L} \sum_{i=1}^{L} \xi[\epsilon_i^2] \tag{4.22}$$

The committe prediction is just the average of the outputs of the L networks, thus it can be written as a simple mean:

$$y_{COM}(\mathbf{X}) = \frac{1}{L} \sum_{i=1}^{L} y_i(\mathbf{X}) \tag{4.23}$$

The error of the committee can be written as:

$$E_{COM} = \xi\Big[\Big(\frac{1}{L} \sum_{i=1}^{L} y_i(\mathbf{X}) - h(\mathbf{X})\Big)^2\Big] = \xi\Big[\Big(\frac{1}{L} \sum_{i=1}^{L} y_i \epsilon_i\Big)^2\Big] \tag{4.24}$$

Making the assumption that $\epsilon_i(\mathbf{X})$ are uncorrelated and having zero mean:

$$\xi[\epsilon_i] = 0 \xi[\epsilon_i \epsilon_j] = 0 if j \neq i \tag{4.25}$$

Than using 4.22 we obtain the relation between the committee error and the average error of the single network:

$$E_{COM} = \frac{1}{L^2} \sum_{i=1}^{L} \xi[\epsilon_i^2] = \frac{1}{L} E_{AV} \tag{4.26}$$

This interesting result says that the average sum-of-square error of the single network can be reduced by a factor L (number of networks in the committee) by averaging the prediction of the networks. However this result is based on strong assumptions: zero mean and uncorrelated, in fact in practice the reduction is much smaller than that, since errors $\epsilon_i(\mathbf{X})$ of different models are highly correlated, anyway it is possible to demonstrate the averaging process with committee cannot increase the expected error value by using Cauchy's inequality:

$$\Big(\sum_{i=1}^{L} \epsilon_i\Big)^2 = L \sum_{i=1}^{L} \epsilon_i^2 \tag{4.27}$$

an thus

$$E_{COM} \leqslant E_{AV} \tag{4.28}$$

Chapter 5

Markov chain Monte Carlo

We start our discussion by doing a recall about the Bayes theorem under a more practical point of view, so some information will be briefly repeated. The goal is to make inference (in terms of probability density function) about the parameter vector ϑ as a set of observation x of the variable X are available, as an example in our case we have to know the damage position and dimension once the strain measurements are available. Different values of ϑ can lead X to take different probability distribution $p(X = x|\vartheta)$, however this is not a conditional distribution since θ is a discrete deterministic variable, so it is just $P(X = x)$ with the model affected by ϑ.
In classical statistics $p(X = x|\vartheta)$ is called likelihood, because it indicates how much the parameter ϑ gives the vector x similar to the one observed, and the real value of theta is the one which makes $p(X = x|\vartheta)$ maximum. It has to be clear that the likelihood is not a probability about ϑ.
However under a Bayesian point of view the likelihood function is a real probability, the variable Θ is assigned to the parameter vector ϑ. Θ is characterized by a pdf $p(\Theta|\vartheta)$, known as *prior distribution* of Θ, furthermore $p(X = x|\Theta = \vartheta)$ is now a conditional probability. As already explain in Chapter 3, the Bayes' theorem is used to compute the *posterior probability* as follow:

$$p(\Theta = \vartheta|X = x) = \frac{p(X = x|\Theta = \vartheta) * p(\Theta|\vartheta)}{p(X = x)} \tag{5.1}$$

The goal is to compute the posterior probability taking into account information coming from the observation of the phenomenon and the prior knowledge about the system. The *normalization constant* $p(X = x)$ makes sure that the posterior is lower than one and it is computed as follow:

$$p(X = x) = \sum_{j=1}^{N} p\left(X = x \mid \Theta = \vartheta_j\right) \cdot p\left(\Theta = \vartheta_j\right) \tag{5.2}$$

where N is the length of the parameter variable vector ϑ. This marginalization constant for real case application is never known since it must be

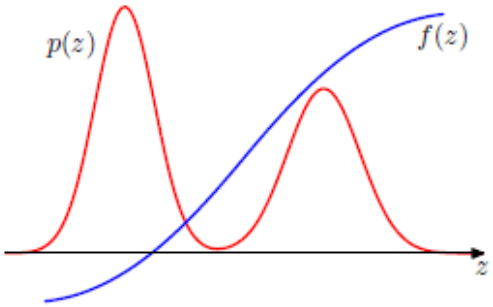


Figure 5.1. Illustration of the function $f(\mathbf{z})$ and the probability distribution $p(\mathbf{z})$.

evaluated for all parameters N of the vector ϑ, thus the Bayes' theorem becomes just a simple proportionality:

$$p\left(\Theta = \vartheta \mid X = x\right) \propto p\left(X = x \mid \Theta = \vartheta\right) \cdot p\left(\Theta = \vartheta\right) \tag{5.3}$$

The computation of the posterior probability requires the calculation of integral that cannot be solved analytically. Since an analytical solution is not available, an approximate inference method based on numerical sampling known as *Monte Carlo* techniques are considered.
By now let us limit ourselves to use the sampling methods to be computed the expectation of a function $f(\mathbf{z})$ with a probability distribution $p(\mathbf{z})$, where $\mathbf{z}$ is a set of whether continuous or discrete variables. In the case of continuous variables:

$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z} \tag{5.4}$$

This situation is shown in Figure 5.1, and we now make the assumption that this expectation is too complex to compute analytically. The main idea behind the sampling methods is that we can obtain a set of sample $\mathbf{z}^{(l)}$ drawn independently from the distribution $p(\mathbf{z})$ in order to approximate the integral with a finite sum:

$$\widetilde{f} = \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{(l)}) \tag{5.5}$$

If $\mathbf{z}^l$ is drawn from the distribution $p(\mathbf{z})$ then $\mathbb{E}[f\hat{}] = \mathbb{E}[f]$. It is important to say that the accuracy of the estimation does not depends upon the dimension of $\mathbf{z}$ and so high accuracy may be reached with also a low value of samples.The main problem which may affect the result is that samples $\mathbf{z}^l$ may not be independent, and so the meaningful samples may be lower than the number actually drawn. Another problem is very easy to see and understand in Figure 5.1, where we have zones in which $f(\mathbf{z})$ is small in regions in which $p(\mathbf{z})$ is large and vice versa, in that case a very large number of samples must

be drawn in order to estimate the expectation with sufficient accuracy.
The problem is now turned to find algorithms able to sample effectively from a (complex) distribution that, as we shall see, will be the key to compute the posterior probability $p(\Theta = \vartheta \mid X = x)$ through a sequence of samples. In our applied case we will have to find the pdf_s of the damage parameters once observations coming from sensors are available. Numerous texts provides explanation about Monte Carlo method such as: [31], [32], [33].

5.1 Rejection sampling

Rejection sampling allows to sample from relatively complex distributions. Let's consider to sample from an univariate distribution $p(\mathbf{z})$, and sampling directly from that is difficult. Furthermore suppose that we cannot evaluate $p(\mathbf{z})$ but we can evaluate $\widetilde{p}(z)$ in such a way that:

$$p(\mathbf{z}) = \frac{1}{Z_p}\widetilde{p}(z) \tag{5.6}$$

where Z_p is unknown.
In order to apply rejection sampling we need a simpler distribution $q(z)$ called *proposal distribution* from which we can easily draw samples. Than we introduce a scaling factor k chosen in order to have $kq(z) \geqslant \widetilde{p}(z)$ for any value of z. The function $kq(z)$ is called *comparison function* and it shown in Figure 5.2. The sampling procedure involves generating two random numbers: first we draw a number z_0 from the distribution $q(z)$ then a number u_0 from the uniform distribution over $[0, kq(z_0)]$. Finally, if $u_0 > \widetilde{p}(z_0)$ the sample is rejected, otherwise it is kept. This means that if the sample pair lies in the gray shaded region shown in Figure 5.2 it will be rejected, the remaining pairs will have a uniform distribution under the curve $\widetilde{p}(z)$. The number z_0 was generated from the proposal $q(z)$ and the acceptance probability is $\widetilde{p}(z)/kq(z)$, thus the number of sample rejected depends on the ratio between

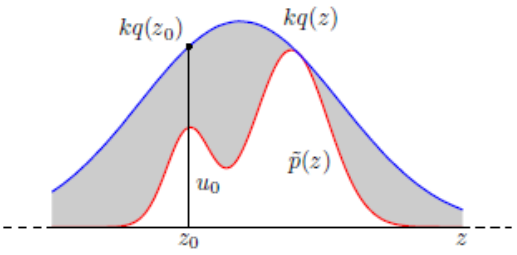


Figure 5.2. In rejection sampling samples are drawn from a proposal distribution $q(z)$ and rejected if they fall in the gray area. The remaining samples are distributed as $p(z)$ which is the normalized distribution of $\widetilde{p}(z)$.

the unnormalized distribution $\widetilde{p}(z)$ and the area under the curve $kq(z)$, so the multiplication constant k must be as small as possible but $kq(z) > \widetilde{p}(z)$.

5.2 Importance sampling

As already seen we wish to sample from a complex distribution in order to evaluate integrals or expectation as shown in equation 5.4. *Importance sampling* is able to approximate integrals but it does not provide a method to draw samples from a given distribution $p(\mathbf{z})$. The approximation given in 5.5 depends on the ability to draw samples from $p(\mathbf{z})$. Let us suppose that it is not practical to draw samples directly from $p(\mathbf{z})$ but it can be evaluated for any value of $\mathbf{z}$. One simple strategy to evaluate the expectation would be to discretized the $\mathbf{z}$-space uniformly and compute the integral as a finite sum as follow:

$$\mathbb{E}[f] \simeq \sum_{l=1}^{L} p(\mathbf{z}^{(l)}) f(\mathbf{z}^{(l)}) \tag{5.7}$$

The first problem related to this approach is that the number of samples increases a lot as the dimensionality of $\mathbf{z}$ increases. Second, the probability distribution often has a small region of the space parameter $\mathbf{z}$ in which it is confined the major of its mass, so uniform sampling will be inefficient, because very few samples will give a contribution to the sum. We wish to sample were $p(\mathbf{z})$ is large, or better, where $p(\mathbf{z})f(\mathbf{z})$ is large.
As the case of rejection sampling, importance sampling is based on a proposal distribution $q(\mathbf{z})$ from which it is easy to draw samples. The expectation can be written with a finite sum:

$$\begin{aligned} \mathbb{E}[f] = \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z} &= \int f(\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} q(\mathbf{z}) d\mathbf{z} \\ &\simeq \frac{1}{L} \sum_{l=1}^{L} \frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})} f(\mathbf{z}^{(l)}) \end{aligned} \tag{5.8}$$

Where $r_l = p(\mathbf{z}^{(l)})/q(\mathbf{z}^{(l)})$ are known as *importance weights* and are needed to correct the bias introduced by sampling from the wrong distribution, unlike rejection sampling all of samples will be retained.
Also in this case, the result strongly depends on how much the proposal distribution $q(\mathbf{z})$ matches the desired distribution $p(\mathbf{z})$. If $p(\mathbf{z})f(\mathbf{z})$ has a significant portion of its mass concentrated in a small region of the $\mathbf{z}$ space, there will be few importance weights having r_l giving a significant contribution, thus the sample size may be much smaller than L.

5.3 Markov Chain Monte Carlo

In the previous section we have seen that rejection sampling and importance sampling have strong limitations for high dimensionality and when $p(\mathbf{z})$ is large where $f(\mathbf{z})$ is small, thus we now turn to the Markov chain Monte Carlo (MCMC) algorithm, in our case it is important to remember that we want to make inference over some parameters distribution once observations are available, in particular we are interested in finding $p\left(\Theta = \vartheta \mid X = x\right)$ through the Markov chain Monte Carlo sampling. As in rejection and importance sampling we draw samples from a proposal distribution, but we now take a record of the current state $\mathbf{z}^t$ and the proposal $q(\mathbf{z}|\mathbf{z}^t)$ depends upon this current state. By an acceptance criterion we decide whether to retain or not the sample, and finally a sequence of samples $\mathbf{z}^1, \mathbf{z}^2, ...$ forms a Markov chain. Again we have $p(\mathbf{z}) = \widetilde{p}(z)/Z_p$, where $\widetilde{p}(z)$ can be easily evaluated, Z_p is unknown, and a proposal distribution $q(\mathbf{z})$ is simple in order to easily draw samples. At each cycle we draw a sample $\mathbf{z}^*$ from the proposal, than this sample is accepted based on a criterion. In the basic framework the proposal is symmetric so $q(\mathbf{z}_A|\mathbf{z}_B) = q(\mathbf{z}_B|\mathbf{z}_A)$. The candidate sample at the iteration $t+1$ will be accepted with probability:

$$A(\mathbf{z}^*, \mathbf{z}^t) = min\Big(1, \frac{\widetilde{p}(\mathbf{z}^*)}{\widetilde{p}(\mathbf{z}^t)}\Big) \tag{5.9}$$

By choosing a random number u with uniform distribution in the interval $[0, 1]$ the sample will be accepted if:

$$A(\mathbf{z}^*, \mathbf{z}^t) > u \tag{5.10}$$

It is important to note that if passing from sample $\mathbf{z}^t$ to $\mathbf{z}^*$ we have an increase of $p(\mathbf{z})$ the sample will accepted for sure.
If the candidate is accepted then $\mathbf{z}^{t+1} = \mathbf{z}^*$, otherwise if the candidate is discarded $\mathbf{z}^{t+1} = \mathbf{z}^t$, and then a new candidate is drawn from $q(\mathbf{z}|\mathbf{z}^{t+1})$ building in this way the sample chain. However it must be noted that if we take the whole chain as it is, the samples are not independent one from each other, and so once the entire chain is computed only one sample each ten is generally kept, in order to have independent samples.

5.4 Markov chain definitions

In this section some general properties of Markov chains are explained, in particular we show under what circumstances a Markov chain will converge toward the desired distribution, which means reaching a *stationary* and *unique* solution, these properties can be achieved by satisfying the *detailed balanced condition*, in order to reach the chain *reversibility* or by satisfying the *ergod-*

icity of an *irreducible* Markov chain. Deeper studies about this topic can be found in [34].

Def. 1: Markov Chain A *Markov chain* is a sequence of random variables that can be imagined as evolving in time It is mathematically defined in terms of its transition kernel. Thus, given a kernel K, a sequence of random variables $\mathbf{z}_n$ is a Markov chain if:

$$p(\mathbf{z}^{(n)}|\mathbf{z}^{(1)}, ..., \mathbf{z}^{(n-1)}) = p(\mathbf{z}^{(n)}|\mathbf{z}^{(n-1)}) \tag{5.11}$$

Def. 2: Transition Kernel A *transition kernel* K is a conditional probability density function that relates two consecutive steps in a Markov chain:

$$K(\mathbf{z}^{(n-1)}, \mathbf{z}^{(n)}) \approx p(\mathbf{z}^{(n)}, \mathbf{z}^{(n-1)}) \tag{5.12}$$

A Markov chain is called homogeneous if the transition kernel are the same for all m.

Def. 3: Stationarity (or invariant) A distribution is said to be invariant, or stationary, with respect to a Markov chain if each step in the chain leaves the distribution invariant. Which means that an *invariant distribution* π exists in a way that $\mathbf{z}_k \approx \pi$, $\mathbf{z}_{k+1} \approx \pi, \ldots$, which means the marginal distribution of $\mathbf{z}_k$ is independent on k. MCMC method is based upon the possibility to meet this requirement. The invariant solution is a candidate solution for the final algorithm convergence, being the marginal probability distributions at various steps all equal to π. Nevertheless, sometimes it does not exist or it is not unique. The *ergodicity* and *irreducibility* of the Markov chain are the conditions that guarantee the existence and uniqueness of the equilibrium solution (invariant solution).

Def. 4: Detailed Balance Condition A sufficient but not necessary condition for ensuring that the required distribution $\pi(\mathbf{z})$ is to choose the transition kernel to satisfy the *detailed balance* condition:

$$p(\mathbf{z})K(\mathbf{z}, \mathbf{z}^{'}) = p(\mathbf{z}^{'})K(\mathbf{z}^{'}, \mathbf{z}) \tag{5.13}$$

If a Markov chain respects the detailed balance is said *reversible*. Eq. 5.13 indicates that the entry rate in a certain state is equal to the exit rate. More precisely, for a process at equilibrium, the number of times the process enters in a state must be equal to the times it exits from the same state (in a fixed step interval).
However, the verification of the DBC implies the knowledge of the equilibrium distribution π. In practice one usually calculates π from the DBC. As a matter of fact, if an irreducible Markov chain satisfies the detailed balance condition with π being a *pdf*, then the chain is positive recurrent and re-

versible and the density π is the invariant equilibrium distribution for the process.

Def. 5: Reversibility A stationary Markov chain is *reversible* if the distribution of $\mathbf{z}_{k+1}$ conditional on $\mathbf{z}_{k+2} = x$ is the same as the distribution of $\mathbf{z}_{k+1}$ conditionally on $\mathbf{z}_k = x$.

Def. 6: Ergodicity The aim of the Markov chain is to sample from a given distribution and order to achieve this purpose the desired distribution must be invariant. It must also be required that for $m \to \infty$ the distribution $p(\mathbf{z}^{(m)}$ converges toward the invariant distribution $\pi(textbfz)$. This property is called *ergodicity* and the invariant distribution is called *equilibrium* distribution. An ergodic distribution can have only one equilibrium distribution.

Def. 10: MCMC A *Markov chain Monte Carlo* method for the estimation of a *pdf* distribution π is any method producing an ergodic Markov chain whose stationary distribution is π.

5.5 Metropolis-Hastings algorithm

Metropolis-Hastings algorithm is used to generate a Markov chain. The aim is to draw samples that admit as stationary distribution the target posterior distribution, as an example the *pdf* associated with the system parameters conditioned on the system observations. Two min consideration must be taken into account:

- If the Markov chain is irreducible, ergodic and stationary, it converges towards a unique invariant distribution. This means that it is possible to choose randomly the initial conditions of the chain (the first sample) as the chain will always converge toward the target distribution. However the first part of the Markov chain (the first samples) must be discarded because it was a transient, and samples do not belong to the posterior target distribution. This transient window is usually named *burn-in* period.

- As already mentioned, a Markov chain samples are dependent (any state of the chain depends only on the previous one). As a matter of fact, the adopted transition kernel usually allows for small jumps which causes a correlation between subsequent samples. Nevertheless, to build a *pdf* one has to refer to independent samples. Thus, after eliminating the *burn-in* period, only one sample every n_t is retained; this process is named *thinning*.

As we have already seen in 5.3 that the drawing process is based on a proposal distribution $q(\cdot)$ and an acceptance probability $\alpha(\cdot)$, where $\alpha(\cdot) \cdot q(\cdot)$ is the transition kernel that links two consecutive steps of the Markov Chain. The acceptance probability will take the following form:

$$\alpha\left(\vartheta_{k+1} \mid \vartheta_k\right) = min\left(\frac{\pi\left(\vartheta_{k+1} \mid y\right) \cdot q\left(\vartheta_k \mid \vartheta_{k+1}\right)}{\pi\left(\vartheta_k \mid y\right) \cdot q\left(\vartheta_{k+1} \mid \vartheta_k\right)} \quad , \quad 1\right) \tag{5.14}$$

The MH algorithm can be summarized through the following steps:

1. Initialize the chain by selecting an arbitrary starting point for the inferred parameter vector ϑ_0, this is the step $k = 0$.
2. At step k, draw a proposal sample $\widetilde{\vartheta}_{k+1}$ from the proposal density $q\left(\vartheta_{k+1} \mid \vartheta_k\right)$
3. Evaluate $\alpha\left(\widetilde{\vartheta}_{k+1} \mid \vartheta_k\right) = min\left(\frac{\pi\left(\widetilde{\vartheta}_{k+1} \mid y\right) \cdot q\left(\vartheta_k \mid \widetilde{\vartheta}_{k+1}\right)}{\pi\left(\vartheta_k \mid y\right) \cdot q\left(\widetilde{\vartheta}_{k+1} \mid \vartheta_k\right)} \quad , \quad 1\right)$
4. Acceptance criterion: the new state $\widetilde{\vartheta}_{k+1}$ will be accepted with probability $\alpha\left(\widetilde{\vartheta}_{k+1} \mid \vartheta_k\right)$ if by generating a random sample r from a uniform distribution $U(0,1)$ we have $\alpha > r$, in tht case $\vartheta_{k+1} = \widetilde{\vartheta}^{k+1}$. Otherwise if $\alpha < r$ the sample will be rejected and so $\vartheta^{k+1} = \vartheta^k$.
5. Repeat steps 2 to 4 until the desired chain length is reached
6. Skip samples relative to the *burn-in* period, activate *thinning* and compute the required estimates.

Note that if a symmetric transition proposal density is selected a strong simplification of Eq. 5.14 is obtained:

$$\alpha\left(\vartheta_{k+1} \mid \vartheta_k\right) = min\left(\frac{\pi\left(\vartheta_{k+1} \mid y\right)}{\pi\left(\vartheta_k \mid y\right)} \quad , \quad 1\right) \tag{5.15}$$

A typical symmetrical distribution is the Gaussian one.
Secondly, $\pi\left(\vartheta_k \mid y\right)$ is the target of the investigation and one has to recall to Bayes theorem in order to solve the problem, as shown in Eq. 5.16.

$$\pi\left(\vartheta_k \mid y\right) = \frac{\pi\left(y \mid \vartheta_k\right) \cdot \pi\left(\vartheta_k\right)}{\int \pi\left(y \mid \vartheta_k\right) \cdot \pi\left(\vartheta_k\right) d\vartheta_k} \propto \pi\left(y \mid \vartheta_k\right) \cdot \pi\left(\vartheta_k\right) \tag{5.16}$$

As always the *normalization constant* is not available, in particular if many parameters have to be estimated. Anyway it appears in both numerator and denominator, so it is neglected. The *prior probability* is $\pi\left(\vartheta_k\right)$ and $\pi\left(y \mid \vartheta_k\right)$ is the *likelihood*. If we do not have a prior knowledge about the system we can use a uniform prior probability, thus non informative. In this case a

further simplification can be done, and the acceptance probability becomes only function of the likelihood function as follow:

$$\alpha\left(\vartheta_{k+1} \mid \vartheta_k\right) = min\left(\frac{\pi\left(y \mid \vartheta_{k+1}\right)}{\pi\left(y \mid \vartheta_k\right)} \quad , \quad 1\right) \tag{5.17}$$

As we shall see, since the measurement are supposed to be affected by a noise $\mu \sim N(0, \sigma^2)$ normally distributed with zero mean and standard deviation σ, the likelihood function can take the following form:

$$\pi\left(y \mid \vartheta_k\right) = L\left(\vartheta_k \mid y\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{costfunction}{2\sigma^2}} \tag{5.18}$$

Again σ is the *standard deviation* of the uncertainty or noise which is inside the observations and the *cost function* is the *square error* calculated between the observations and the available model with the proposed parameter inside.

Chapter 6

Case studied

6.1 Introduction

In this section the diagnostic framework is going to be exploited to localize a delamination in a composite panel caused by a low velocity impact. The diagnostic system is based on sensors (such as strain gauges) placed on the surface of the panel, these sensors gives information about the strain in the position they are located. These strains are needed to compute the likelihood function in the MCMC algorithm, in this way it is possible to build the Markov chain and finally to solve the inverse problem by computing the *pdf* of the damage parameters. Both Kriging and ANN surrogate models will be implemented in the MCMC algorithm and a statistical analysis of the result will be carried out in order to know how performances change as a function of the damage size and noise level. The effectiveness of the tool is verified through a series of finite element simulations and all the concepts explained in the previous Chapters will be applied for this specific application.
The Chapter is organized as follow: first MCMC-MH is explained fo this very application 6.2, then an explanation about how the FE model of the composite materials shown together on how the training database has been obtained 6.3. In 6.4 and 6.5 is shown the result of the training process of the two surrogate models employed. Finally the application to some simulated cases is shown in 6.6.

6.2 Metropolis-Hasting Markov Chain Monte Carlo: likelihood calculation and adaptive proposal

Suppose that a composite panel is mounted aircraft wing and during its operation a low velocity impact occurs causing a delamination among plies of the material, the goal is to identify the damage parameters (position and dimension) real-time, in particular we do not want a deterministic result, but

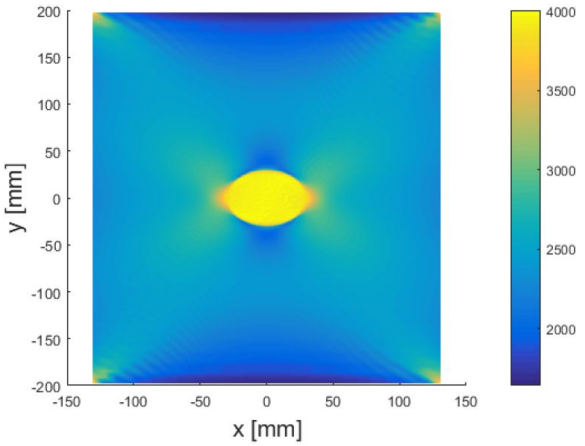


Figure 6.1. Example of the strain field induced in a composite panel, as a consequence of a delamination.

a probabilistic one, which means that we are interested in the computation of the probability density function of the damage parameters. At this scope we can draw samples from the posterior distribution.
As we have seen in Chapter 5, the Markov Chain Monte Carlo Metropolis-Hasting algorithm is able to sample from the posterior probability, so this powerful tool can be used for our purpose. Let us suppose the panel we are talking about is shown in Figure 6.1, the damage parameters we are interested in are: x, y and r, where x, y are the center coordinates of the delamination and r is its radius (because as we shall see we suppose to have circular delamination). We can assign the variable ϑ to the damage parameters in a way that $\vartheta = [x\ y\ r]$. A set of strain measurements is collected from sensors, the variable z is assign to them, thus the aim is to sample from the multivariate posterior distribution $p(\vartheta|z)$. It is now proposed again the likelihood function already shown in Section 5.5 in order to make some consideration about its form and its error function:

$$p\left(z \mid \vartheta_i\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{costfunction}{2\sigma^2}} \tag{6.1}$$

First it is now time to define clearly what is the cost function: it is the discrepancy of the real measurements coming from sensors, and the strain z_i^* a delamination with damage parameters sampled by the MCMC at each i would give. Being ϑ^* the sample at iteration i, z_i^* will be:

$$z_i^* = f(\vartheta^*) \tag{6.2}$$

It is easy to understand that the measurements are always corrupted by a noise ε. It is reasonable to assume that ε is Gaussian distributed with zero mean and standard deviation σ as follow:

$$\varepsilon \sim N(0, \sigma^2) \tag{6.3}$$

thus since z_i^* is not corrupted by noise, we can write:

$$z_i = f(\vartheta^*) + \varepsilon \tag{6.4}$$

Through these considerations and having more than one sensor, the likelihood function takes the form of a Gaussian function with error function as shown below:

$$p\left(z \mid \vartheta_i\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{\left(-\frac{\sum_i (z_i - z_i^*)^2}{2\sigma^2}\right)} \tag{6.5}$$

To compute this function we need to know how to obtain the unnoised strain estimation z_i^*, for example through an analytical or numerical model.
To model a delamination in a panel an analytical model is not available, so finite element (FE) analysis must be used. The most intuitive way to compute z_i^* is, once ϑ^* has been drawn, running a FE analysis in which the damage simulated takes the damage parameters contained in the vector ϑ^*. Although through this methodology we are able to converge toward the desired distribution, in order to compute an entire Markov chain the likelihood function must be assessed thousands of times, and for each sample a FE simulation should be run, which implies that thousands of FE simulations must be carried out. This would increase the computational burden, making unfeasible a real-time identification. To overcome this issue surrogate models come to help us. If we train off-line a surrogate model, whether ANN or Kriging, then we can use it to obtain z_i^* as the sample ϑ^* is drawn.
Till now we have never discussed about the drawing process and the proposal distribution, except that it is useful to be symmetric in order to simplify the way to compute the acceptance probability, the acceptance criterion is shown in Figure 6.2.
Gaussian distribution represents a function from which it is easy to draw samples and it is symmetric, thus we will use it as proposal distribution. Since we have sample three parameters the distribution must be a multivariate one, so we cannot define a single standard deviation that characterizes the distribution but we have to define a *covariance matrix*. The covariance matrix is a $n \times n$ matrix where n is the number of damage parameters (3 in our case) which define the orientation of the multivariate Gaussian distribution. Since it is suppose that there is no correlation between the damage parameters it will be a diagonal matrix as follow:

$$C = \begin{bmatrix} c_1 & 0 & 0 \\ 0 & c_2 & 0 \\ 0 & 0 & c_3 \end{bmatrix} \tag{6.6}$$

were c_i are the variances of each damage parameter.

M-H Algorithm: acceptance criterion

Initialize the algorithm:

draw a candidate in the parameters space: $\boldsymbol{\vartheta}^* \sim q\left(\boldsymbol{\vartheta}^* | \boldsymbol{\vartheta}^{(j-1)}\right) = N(\boldsymbol{\vartheta}^{(j-1)}, \sum_q)$

evaluate the likelihood: $p(\boldsymbol{z}|\boldsymbol{z}^*)$

accept and store $\boldsymbol{\vartheta}^*$ with probability

$$\alpha\left(\boldsymbol{\vartheta}^*, \boldsymbol{\vartheta}^{(j-1)}\right) = \min\left(1, \frac{p(\boldsymbol{z}|\boldsymbol{z}^*)\, q(\boldsymbol{\vartheta}^{(j-1)}|\boldsymbol{\vartheta}^*)\, p(\boldsymbol{\vartheta}^*)}{p\left(\boldsymbol{z}\middle|\boldsymbol{z}^*_{j-1}\right) q(\boldsymbol{\vartheta}^*|\boldsymbol{\vartheta}^{(j-1)})\, p(\boldsymbol{\vartheta}^{(j-1)})}\right)$$

If $\alpha\left(\boldsymbol{\vartheta}^*, \boldsymbol{\vartheta}^{(j-1)}\right)$>rand[0;1]

Sample accepted

Figure 6.2. Metropolis-Hasting algorithm acceptance criterion.

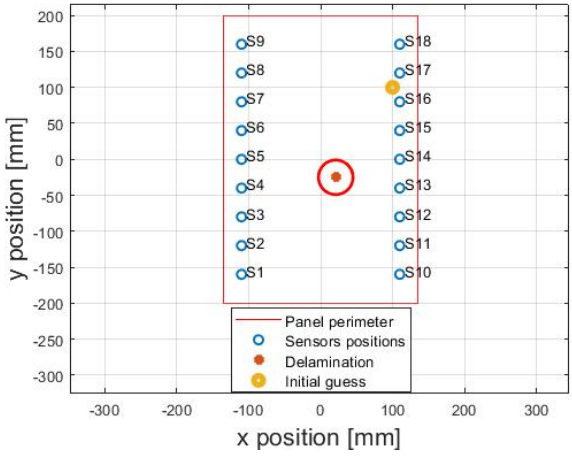


Figure 6.3. Case in which it is shown a delamination in the panel and the first sample.

A main problem is related to the tuning of this matrix, in fact C can be seen as how far from the previous sample (in term of space, since we are talking about spacial coordinates) a new one can be drawn, the larger the variance is, the farer could be the new sample and vice versa. By taking as reference Figure 6.3, we can see that the first sample may be far from the delamination center, we have to be able to converge toward the desired distribution without consuming too many iterations. If the variance is too large the number of samples accepted will be too low, thus the algorithm will not converge toward the most probable damage parameters distribution. On the other hand if the variance is too small the acceptance rate will be very high, as a consequence the number of iteration needed to the stationary distribution will be too high. The ideal situation requires to have a larger variance at the initial stage in order to have larger proposal steps, and then having a lower variance as we approach the zone with an higher likelihood.This issue is overcame by adopting an adaptive proposal covariance matrix [35]. This method, shown in Figure 6.4, allows to recompute the covariance matrix af-

ter a certain number of iteration U (called update frequency) by using the residuals of the last H samples (memory).

M-H Algorithm with adaptive proposal

Initialize the algorithm:

$\boldsymbol{\vartheta}_0$

$U = U_0, H = H_0$

for j =1: N

draw a candidate in the parameters space: $\boldsymbol{\vartheta}^* \sim q\left(\boldsymbol{\vartheta}^* | \boldsymbol{\vartheta}^{(j-1)}\right) = N(\boldsymbol{\vartheta}^{(j-1)}, \sum_q)$

evaluate the likelihood: $p(\boldsymbol{z}|\boldsymbol{z}^*)$

accept and store $\boldsymbol{\vartheta}^*$ with probability

$$\alpha\left(\boldsymbol{\vartheta}^*, \boldsymbol{\vartheta}^{(j-1)}\right) = \min\left(1, \frac{p(\boldsymbol{z}|\boldsymbol{z}^*)\, q(\boldsymbol{\vartheta}^{(j-1)}|\boldsymbol{\vartheta}^*)\, p(\boldsymbol{\vartheta}^*)}{p\left(\boldsymbol{z}\middle|\boldsymbol{z}^*_{j-1}\right) q(\boldsymbol{\vartheta}^*|\boldsymbol{\vartheta}^{(j-1)})\, p(\boldsymbol{\vartheta}^{(j-1)})}\right)$$

If the remainder of (j/U) is null

Generate the residual of the chain: $\boldsymbol{R}(\boldsymbol{\theta}_H) = \boldsymbol{\theta}_H - \boldsymbol{E}(\boldsymbol{\theta}_H)$ where $\boldsymbol{\theta}_H$ is the matrix of the last H samples of the Markov chain and $\boldsymbol{E}(\cdot)$ is the expected value

Update the covariance matrix of the proposal:

$$\sigma^2 = \sigma^2_{\theta_H} = \frac{C_d^2}{H-1} \boldsymbol{R}(\boldsymbol{\theta}_H)^T \boldsymbol{R}(\boldsymbol{\theta}_H)$$

where $C_d^2 = \frac{2{,}4}{\sqrt{d}}$ and d is the number of parameters to be estimated

Figure 6.4. Metropolis-Hasting algorithm adaptive proposal algorithm.

6.3 Composite FE model

As we have seen, running a FE analysis for each sample of the Markov chain is computationally prohibitive, thus a surrogate model is needed for a fast evaluation of the likelihood. Independently from the surrogate chosen a data set is needed to be created in order to train the neural network or the Kriging model, and so to learn the relationship between the input parameters $\vartheta = [x\ y\ r]$ and the observed strain $\varepsilon = [\varepsilon_1, .., \varepsilon_i, ..\varepsilon_n]$. This data set can come either from experiments or simulations. The path chosen in this work is to run a series of FE simulations off-line, in which the damage position and dimension is varied and strains at the sensors' locations are extracted. The layup of the carbon fiber reinforced plastic (CFRP) is an 18 symmetric plies as follow $[45, -45, 0, 45, -45, -45, 45, 90, 0]_S$, the feature of the panel are shown in table 6.1 while the mechanical properties in table 6.2.

The panel detailed model shall establish a relationship between the parameters space θ and the observable variables space ε. Once the relationship is identified, one should be able through the M-H MCMC algorithm integrated with a surrogate model, trained based on a detailed (FE) panel model, to get back and estimates the parameters posterior *pdf* when a measure of the variables is given. Naturally, the chosen variables have to exhibit some sensi-

Panel length [mm]	Panel width [mm]	Panel thickness [mm]	Ply thickness [mm]
400	270	3.6	0.2

Table 6.1. Panel features.

E11 [MPa]	E22 [MPa]	G12 [MPa]	G13 [MPa]	G23 [MPa]	ν [MPa]
157487	9946	4950	4950	3209	0.24

Table 6.2. Panel properties.

tivity to the parameters variation. Ergo, while the detailed surrogate models allow to relate the parameters space to the observable variables space, M-H MCMC enables to solve the inverse solution. The panel features the damage parameters θ, to be identified are those which univocally characterize the impact damage [36], and the observed variables are represented by some of the strain field components on the panel outermost plies. On the surface of the panel two strain components are present ε_{11} and ε_{22}. Since we have to decide which component we have to consider, a sensitivity analysis in which it is studied how strains are affected as a function of the distance from a delamination is carried out. These results are shown in Figure 6.6 in which the strains trend are shown for a central delamination for points which moves along the y axis with $x = -40$ as shown in Figure 6.5. From this study it is clear that the external ply strain component which exhibits the highest sensitivity to the damage parameters is ε_{22}, i.e. the strain along the panel longitudinal direction, thus to train the surrogate model only this component has been taken into account, since in real word application the sensor coverage must be minimized for cost and weight saving.
The sensor location is chosen in order to avoid an overlapping between a sensor and the delamination as shown in Figure 6.7. The number of sensor chosen is 18 and the placement is equally spaced among the edge. As proved by [36],when an impact occurs and a delamination is created its shape is not constant over the panel thickness, and it strongly depends upon the layup. In order to avoid this type of model complexity, a simplified model of the impact damage has been considered:

- The delamination size is equal for all plies
- The delamination has a circular shape so the size is characterized by its radius R_{DEL}

In order to well characterized the damage not only a delamination has been introduced, but also a damage of the fiber and the matrix respectively are introduced (intra-laminar damage). In order to take into account the intra-

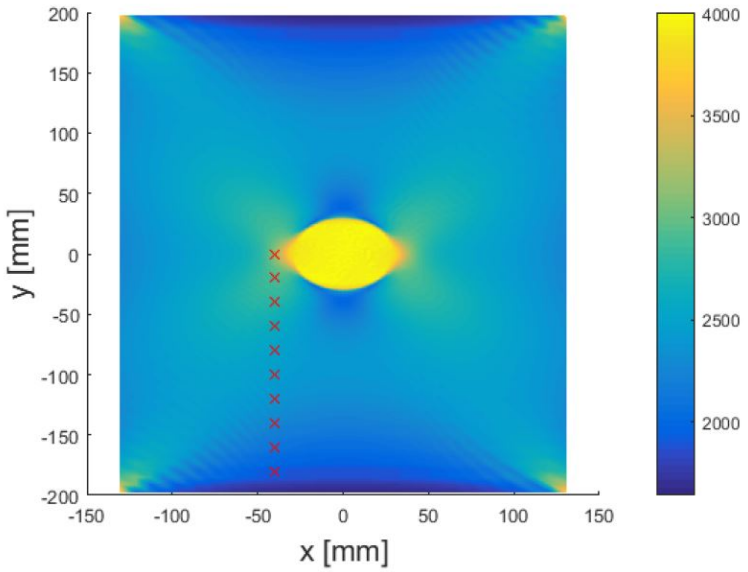
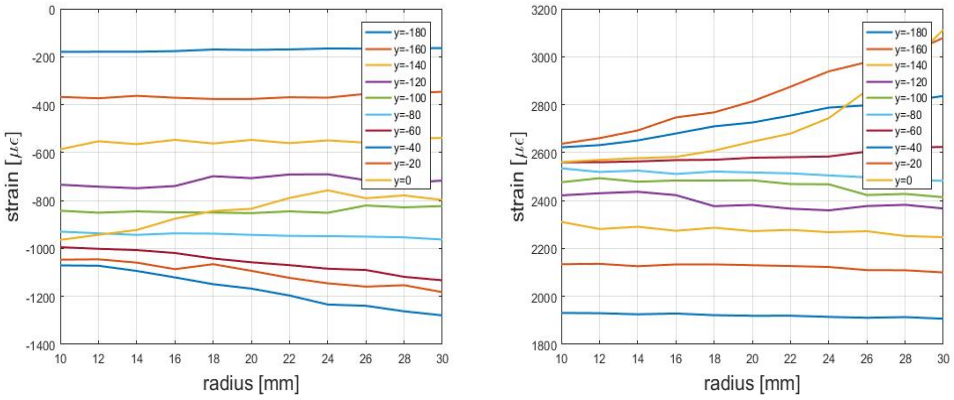


Figure 6.5. The red x are the points in which strains are measured as a function of the delamination radius un a tensile load.



(a) (b)

Figure 6.6. (a) ε_{11} trend for various y distances and for different damage of the fiber, while (b) shows ε_{22} trend for various y distances and for different damage of the fiber.

laminar damage two coefficients d_f, d_m related to the damage of the fiber and matrix are introduced. A directed consequence of these damages is that the stiffness of the panel reduces, the following stiffness degradation scheme is considered:

$$\tilde{E}_{11} = (1 - d_f)E_{11} \tag{6.7}$$

$$\tilde{E}_{22} = (1 - d_f)(1 - d_m)E_{22} \tag{6.8}$$

$$\tilde{G}_{12} = (1 - d_f)(1 - d_m)G_{12} \tag{6.9}$$

The damaged area shown in Figure 6.5 is modeled as follow :

- A circular shaped delamination area is introduced over the whole thickness of the panel having radius R_{DEL} in which the damage indexes assumes the values shown in table 6.3.
- A smaller circular area (impact location area), concentric with the delamination one, having radius R_0 equal to $R_{DEL}/3$ in which the damage indexes are the highest (table 6.3).

	d_f	d_m	R
Delamination area	0.5	0.5	R_{DEL}
Impact location	0.9	0.9	$R_{DEL}/3$

Table 6.3. Damage feature.

The finite element (FE) model is built up with Abaqus, and being the layup symmetric and the loading and boundary conditions of the panel are symmetric too, the two outermost surfaces are being characterized by the same strain field. The load the panel is subjected, is kept constant for each damage

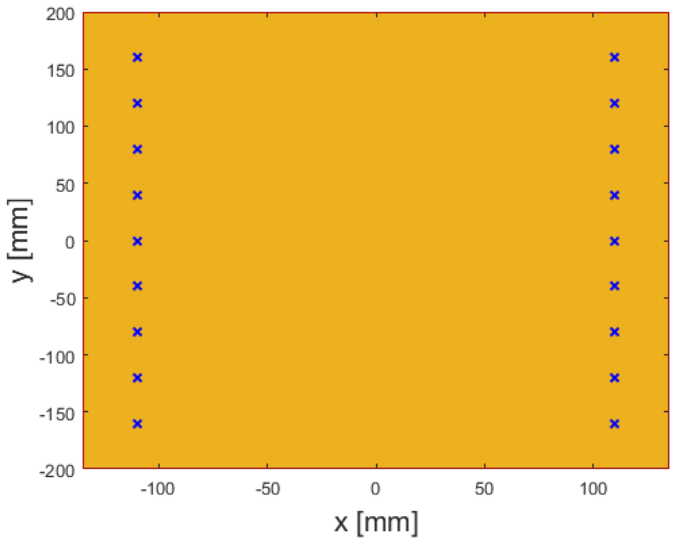
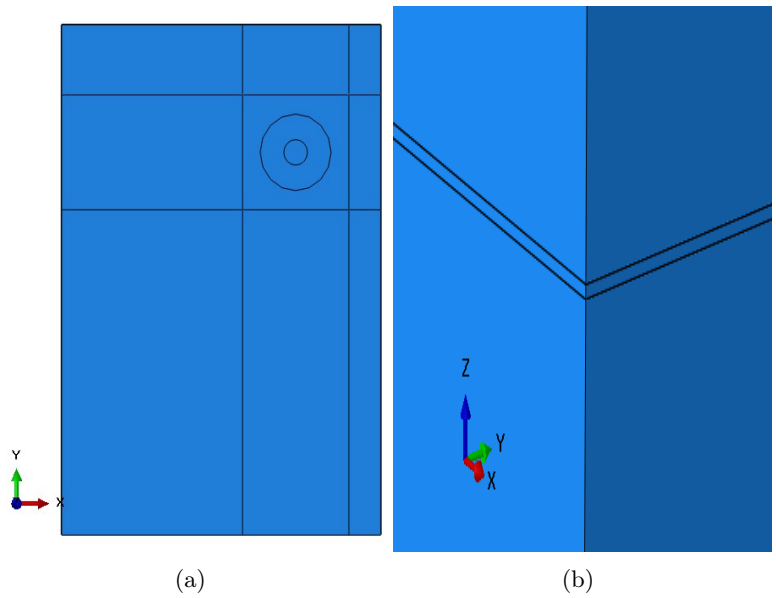


Figure 6.7. Sensors position on the panel.

simulated, thus it is not considered in the damage parameters matrix. To model each ply continuum shells (SC8R element) has been used. Between adjacent plies a narrow-band of vanishing thickness (0,001mm), named the cohesive zone, is inserted (Figure 6.8). Adjacent plies are connected to these cohesive layers via tie constraints, except for the zones in which delamination has occurred, which are tie-free. Here, the cohesive zone modeling (CZM) is exploited for the delamination modeling only [37]. Impact damage is taken into consideration by adjusting the plies stiffness in the impact area, according to matrix and fiber damage indexes and considering the stiffness degradation scheme described previously. The force is 10kN and it is applied longitudinally at one of the free ends, while the other one is clamped. An adaptive mash is adopted in order to have a smaller mesh size in the neighborhood of the delamination and a coarser one far from it, as shown in Figure 6.9.
The training set is the same for both ANN and Kriging, and it is generated from several FE simulations in which position and dimension of the delamination is changed, in particular the damage position parameters (x, y) are the delamination center, while the size is R_{DEL}. In order to achieve a good training a specific surrogate model is trained for each sensor, so we will have eighteen committees of neural networks and eighteen Kriging models. The dataset is composed by:

- A $N \times M$ input matrix ϑ^{INPUT} in which N is the number of simulated



(a) (b)

Figure 6.8. (a) Panel in which the inner circle is the impact zone, the external one is the delamination induced zone, (b) cohesive layer modeling.

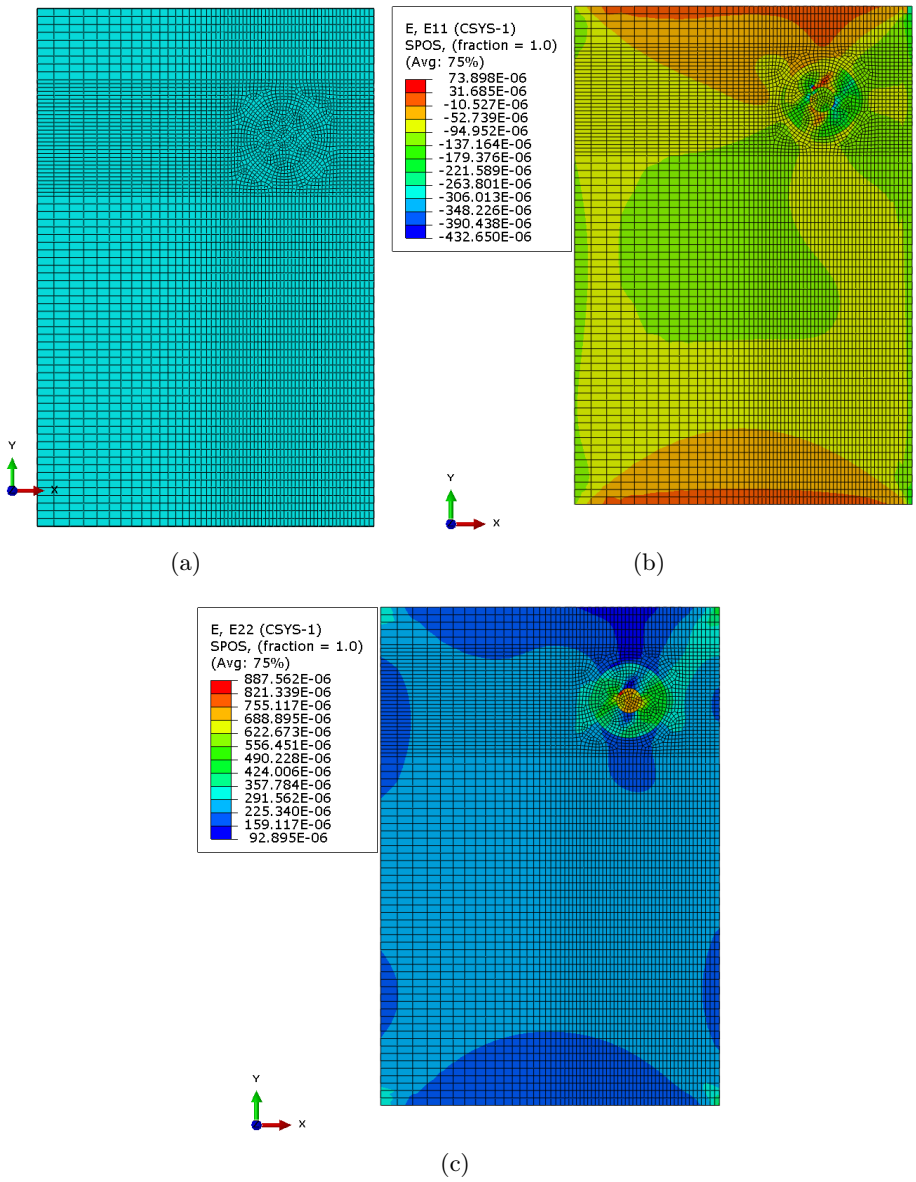


Figure 6.9. (a) FE adaptive mesh, (b) external ply resulting ε_{11} strain field, (b) external ply resulting ε_{22} strain field.

delamination (data set size) and M is the number of damage parameters (x, y, R_{DEL}).

- An output $N \times 1$ vector Z^{OUTPUT} which contains the strain response caused by each delamination at the specific sensor location.

Both of them are obtained via finite element. In table 6.4 it is shown the step in which the delamination parameters are varied and Figure 6.10 it is

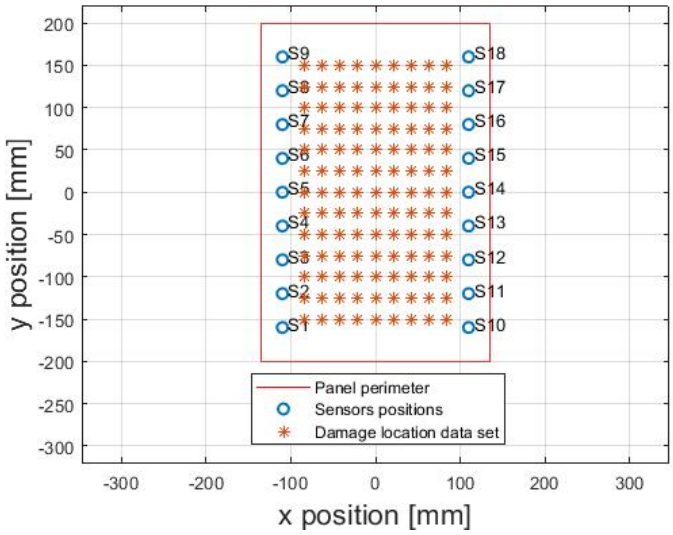


Figure 6.10. Delamination positions used to generate the training data set.

R_{DEL}	x[mm]	y [mm]	Combinations
10:2:30	-84:21:84	-150:25:150	1507

Table 6.4. Bounds and step size of the discretized parameter space.

possible to see them from a graphical point of view, while in eq. 6.10 and 6.11 it is shown how the input-output matrices are formed. It must be precise that not all the delaminations present in the table have been simulated by FE, in fact they have been simulated in only a quarter of the panel, then through symmetries with respect to the axises the complete dataset has been created.

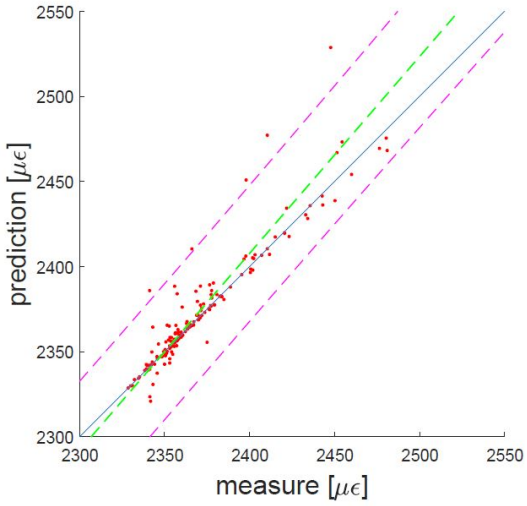
$$\vartheta^{INPUT} = \begin{bmatrix} x_1 & y_1 & R_1 \\ . & . & . \\ . & . & . \\ . & . & . \\ x_N & y_N & R_N \end{bmatrix} \tag{6.10}$$

$$Z^{OUTPUT} = \begin{bmatrix} \varepsilon_{22}^1 \\ . \\ . \\ . \\ \varepsilon_{22}^N \end{bmatrix} \tag{6.11}$$

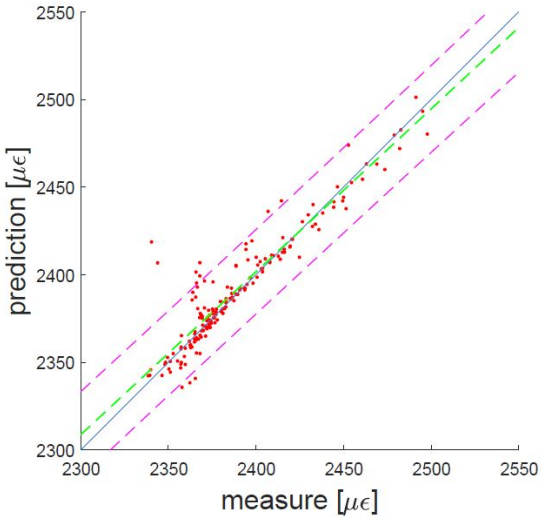
where N is equal 1507 (number of total delaminations simulated). Again the training is done for each single sensor. Now we are able to train our

surrogate models in order to learn the relationship $z_i = f(x_i, y_i, R_i)$, between the damage parameters matrix ϑ^{INPUT} and Z^{OUTPUT}.

6.4 Kriging training



(a)



(b)

Figure 6.11. (a) Scatter plot of the test delaminations of sensor 4 , (b) Scatter plot of the test delaminations of sensor 5. The dotted green line is the fitting line, while the dotted magenta lines represent the 95% confidence interval.

The data set created with FE includes a number of 1507 delaminations of various positions and dimensions, however not all of them were used as data training points but 15% of them (randomly selected) were used as testing points, which means points used to check the goodness of the training. A worth consideration must be done about Kriging training. The prediction strongly depends upon the values found in the optimization process shown in equation 3.65 in which the likelihood function is derived with respect hyperparameters. Actually the likelihood is a very complex function since it is a function $f : \mathbb{R}^3 \to \mathbb{R}^1$ thus multiple local minimum are present, and each one can return different results in the prediction process. As a consequence of this, the optimum point returned by the software strongly depends on the starting point, so it was up to us finding the proper optimum point. To check the precision of the training scatter plots of the test set were plotted for each sensor, some examples are given in Figure 6.11.

6.5 ANN committee training

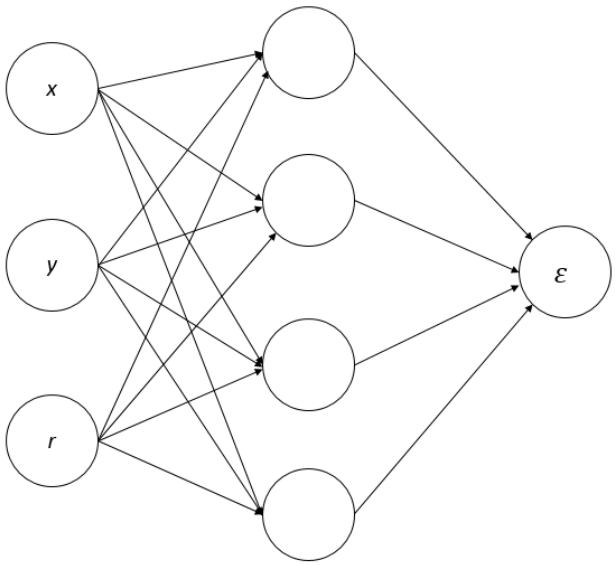


Figure 6.12. ANN structure.

A ANN multilayer perceptrons is fed with data obtained from FE simulations, the layout chosen is the simplest one: a sigmoid function in the hidden one, and a linear function in the output layer as shown in Figure 6.12.
Since the damage parameters-strain relation is quite complex, one single neu-

ral network is not able to learn accurately the relationship, thus a committee is required, in fact as shown in section 4.5, the committee error is lower than the single neural network, furthermore through committee we are not employing a deterministic surrogate model, but we are switching to a statistical model, since we are able to compute mean and variance of the distribution of networks, and then we are able to compare these features with respect to Kriging.
To make up our the surrogate model we need to define: the number of neuron, and the number networks in the committee.
The number of neuron should minimize the root-mean-squared error (RMSE) with respect to the training and testing points, and contemporary avoid overfitting. In order to avoid over fitting early-stopping is adopted, thus the dataset is divided into three subsets: training, validation and test sets with a 70%, 15%, 15% proportions respectively. To choose the number of neuron the trend of the RMSE as a function of the neuron number has been studied and as shown in Figure 6.13 from 60 neurons on, we have that the RMSE remains constant, thus 60 will be the number of hidden neurons chosen.

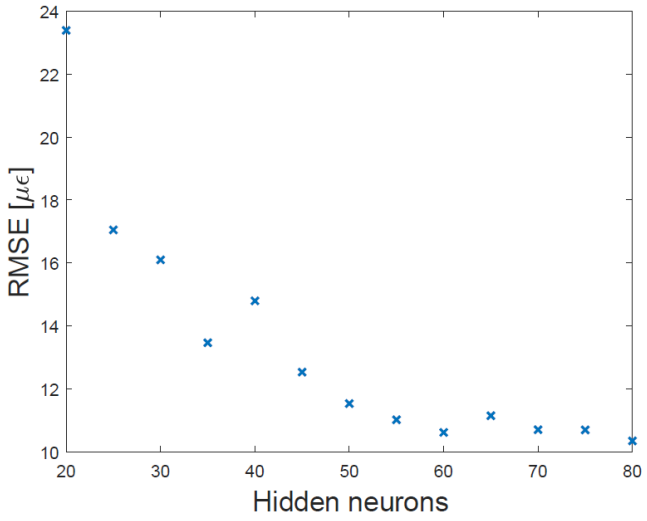
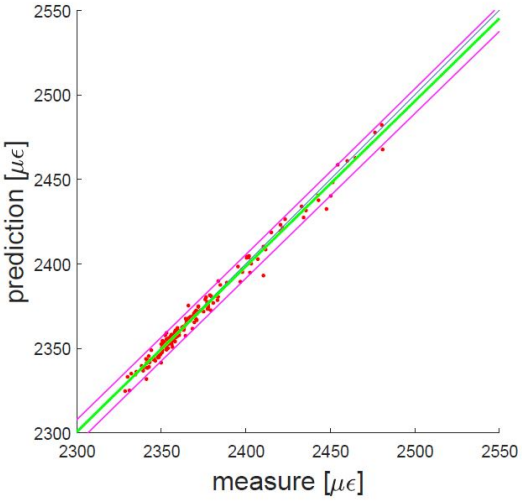
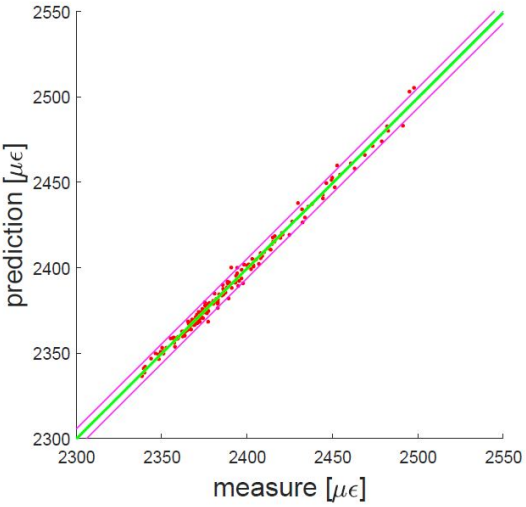


Figure 6.13. RMSE versus hidden units.

The number of networks per committee has been decided upon two conflicting requirements: first we need to have a good generalization and to reduce the RMSE, second we need to maintain low the computational effort, since running more ANNs is much more time consuming. To satisfy both of them 20 networks have been employed. In order to check if the training process has achieved a good result, an example of the scatter plot of the test set for



(a)



(b)

Figure 6.14. (a) Scatter plot of the test delaminations of sensor 4 , (b) Scatter plot of the test delaminations of sensor 5. The dotted green line is the fitting line, while the dotted magenta lines represent the 95% confidence interval.

sensor 4 and 5 are shown in Figure 6.14.
We can observe that ANN committee has a better fitting results than Kriging.

6.6 Application to impact damage

It is now time to apply all the concepts explained to a simulated delamination damage. As already said MCMC is used to solve the inverse problem: computing the posterior *pdf* of each damage parameter $p(\vartheta|z)$ once observations of z in form of strain measurement on the surface of the panel are available. In order to generate a wide range of cases of various position and dimension, 100 delamination were retrieve from FE simulations in which strain measurements at the sensor positions were collected and then smeared with Gaussian noise. In particular half of the delamination chosen where taken from the dataset used to trained the surrogates and then smeared with Gaussian noise, the other half was obtained by running new FE simulations in order to have a completely new delamination locations not coincident with the one used to train the surrogate. In order to understand the performance of the diagnostic system when data are affected by noise (measurements always are), three level of noise were investigated: 1%, 2%, 3%, these noises represent two times the standard deviation of a normal distribution with mean centered in the measure itself, as follow:

$$\sigma = \varepsilon_{meas} * 1\% \tag{6.12}$$

$$\varepsilon^*_{meas} \sim N(\varepsilon_{meas}, \sigma^2) \tag{6.13}$$

Let's start by showing in Figure 6.15 the Markov chain obtain from a simulation by using both Kriging surrogate model, together with the *pdf* we can

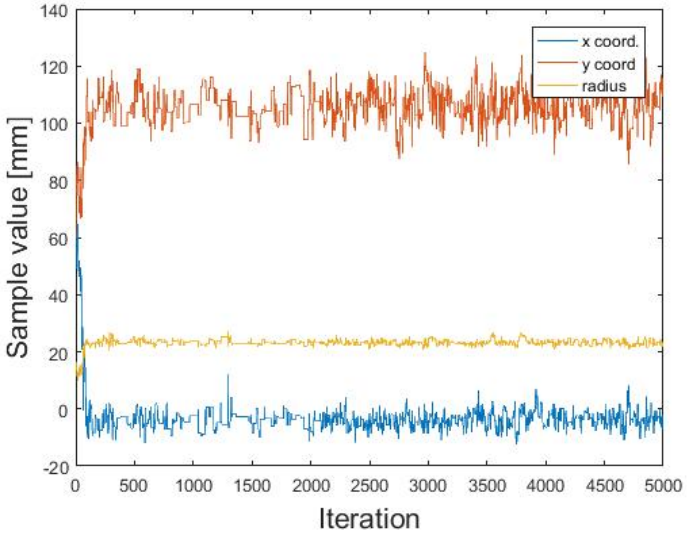


Figure 6.15. Markov chains of the three damage parameters using Kriging surrogate model.

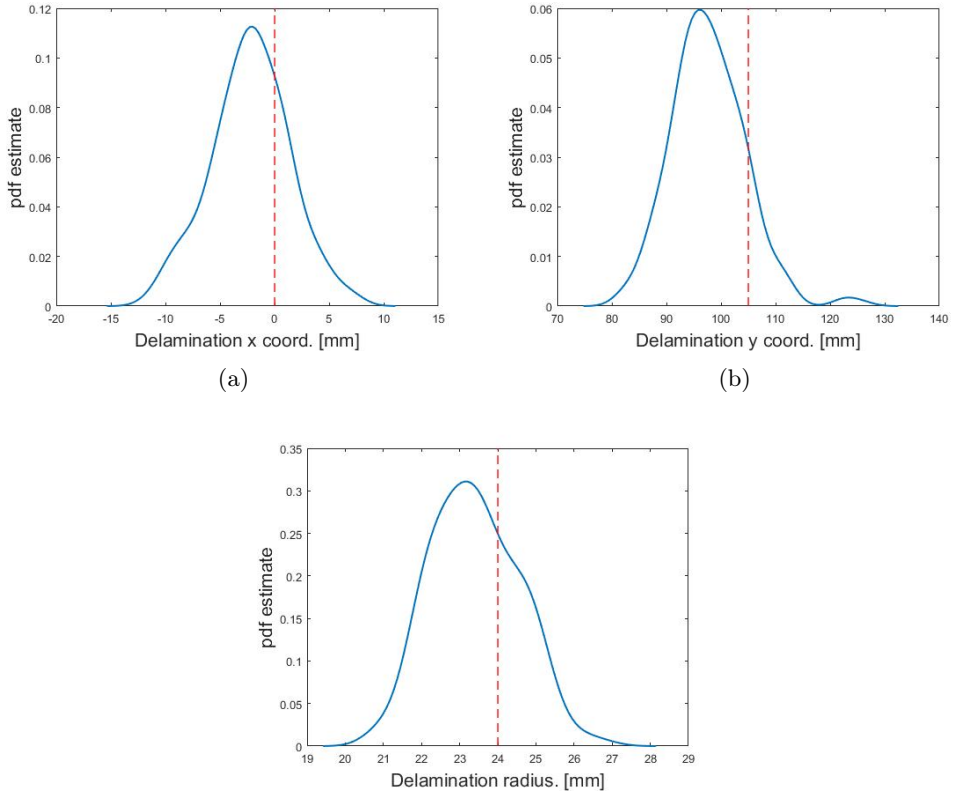


Figure 6.16. (a) x coord. distribution, (b) y coord. distribution, (c) radius distribution with noise level of 1%, the red dotted line is the true value, Kriging surrogate model is used.

derive from. In the example shown in Figure 6.15 there is a MCMC chain of a delamination with the following damage parameter: $\theta = [42, 45, 26]$ and noise level equal to 1%, and in Figure 6.16 it is possible to see the *pdf* of the three damage parameters obtained from those chains.
To compute the *pdf* the burn in period has been discarded, only half of the chain has been kept, and only one sample each 10 has been considered (thinning process).
Two main information can be extrapolated from each distribution:

- First we can compare the real feature with respect the mode of the distribution, in this way we can know the accuracy of the system for that specific case.
- We can compute the interval dispersion around the mode, which is the interval in which 95% of the data are contained, in order to compute the precision of the system. As a matter of fact the result may be accurate

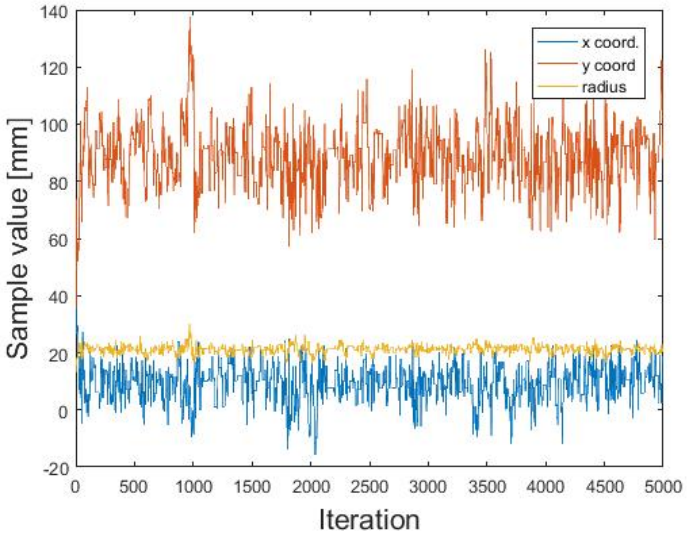


Figure 6.17. Markov chains of the three damage parameters with noise level of 2%, using Kriging surrogate model.

but at the same time not precise. It means that if the distribution was a Gaussian one, the standard deviation would be quite high, if the distribution is not Gaussian, the two percentiles at 5% and 95% are needed to compute the dispersion intervals.

One of the aim of this work is to understand the noise effect on performances. As first observation in Figures 6.17 and 6.18 are shown the MCMC chains of the three damage parameters of the same delamination shown in Figure 6.15 but this time with an higher level noise. By comparing these two cases it is clear that the probability distributions become larger and less tall, which means that at least the precision (dispersion around the mode) becomes worse. In this case the likelihood is not able to reach a location with a real maximum.
In order to know how noise on measurement affects both accuracy and precision, we can run all the 100 delaminations with each noise value, and it is possible to see how precision and accuracy change as a function of it. In particular the statistical database is built as follow:

- Select the noise level wanted.
- Run all the 100 delaminations ten times, in order to sample a different noise combination for each sensor each times.
- For each *pdf* obtained, compute the error between the mode and real value, and the dispersion interval
- Change the noise level and repeat step 2.

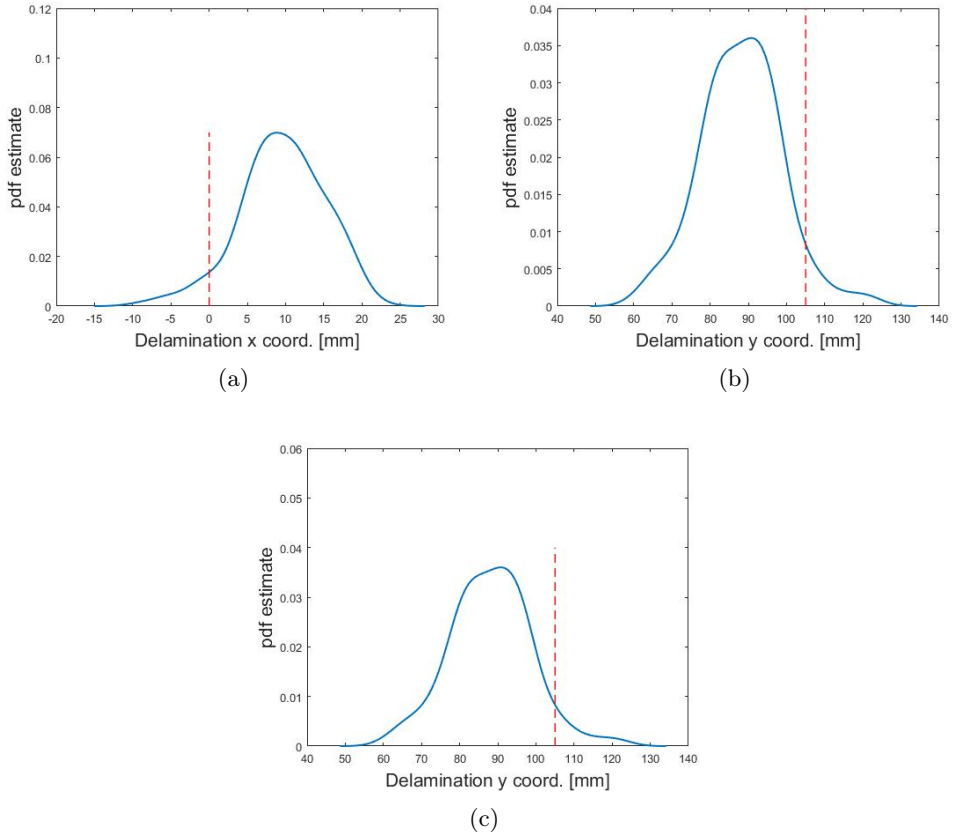
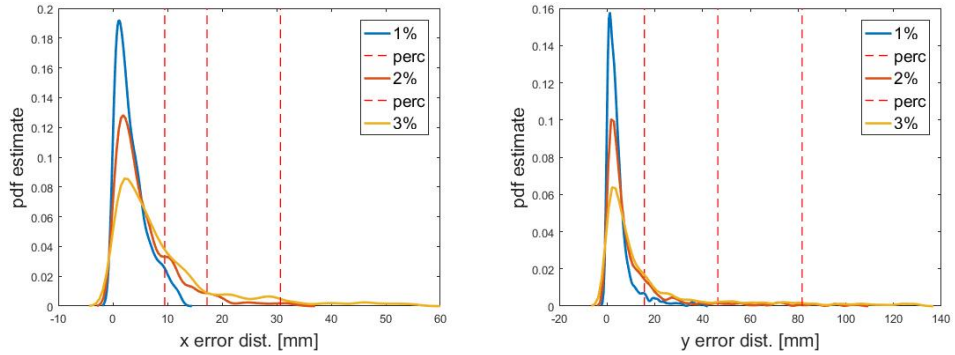


Figure 6.18. (a) x coord. distribution, (b) y coord. distribution, (c) radius distribution with noise level of 2%, the red dotted line is the true value, Kriging surrogate model is used.

Following this procedure the total number of delaminations simulated is 3000, and we have a database of errors and dispersions as a function of the noise, by observing from Figures 6.19 to 6.22 we can conclude that :

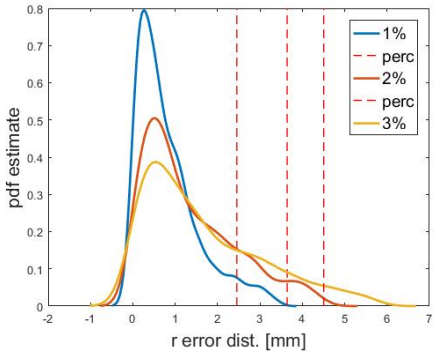
- As the noise level increases the dispersion and error percentile becomes more and more larger.
- The dispersion along the y direction is much wider than the other two even though the error is not very high.
- These trends are quite similar between ANN and Kriging.

Even though the mean value of a Markov chain could be similar to the actual one (good accuracy), having an high dispersion means that the MCMC is not able to converge toward a region in which the likelihood is maximum which implies that the simulated noised strains are very different from those returned by the surrogate. This situation is particularly true when we are in



(a)

(b)



(c)

Figure 6.19. (a) x coord. error distribution, (b) y coord. error distribution, (c) radius error distribution, the red dotted line is the 95% percentile, Kriging surrogate model is employed.

presence of small delaminations placed in the middle of the panel far from the sensors, so the strains registered by them are low, and the noise injected can have a great influence on measurements. By excluding progressively small delaminations from the data base, it is possible to obtain a sensitivity analysis of the system to the delamination size. In the following tables are shown the trends of the 95% percentile of the error e and dispersion d as a function of the radius of the delamination. From the graphs shown from Figure 6.23 to Figure 6.28, in which it is shown the 95% percentile trend of the error and dispersion of each damage parameter, it is clear that the damage parameter which is more affected by the noise and the radius size is the y coordinate. In particular the best performance in terms of accuracy and precision is achieved when the noise level is minimum and the delamination dimension range is high, but as we approach an higher noise level and we take into account also smaller delamination we have a strong increment of both error and dispersion, confirming that low dimension delamination are

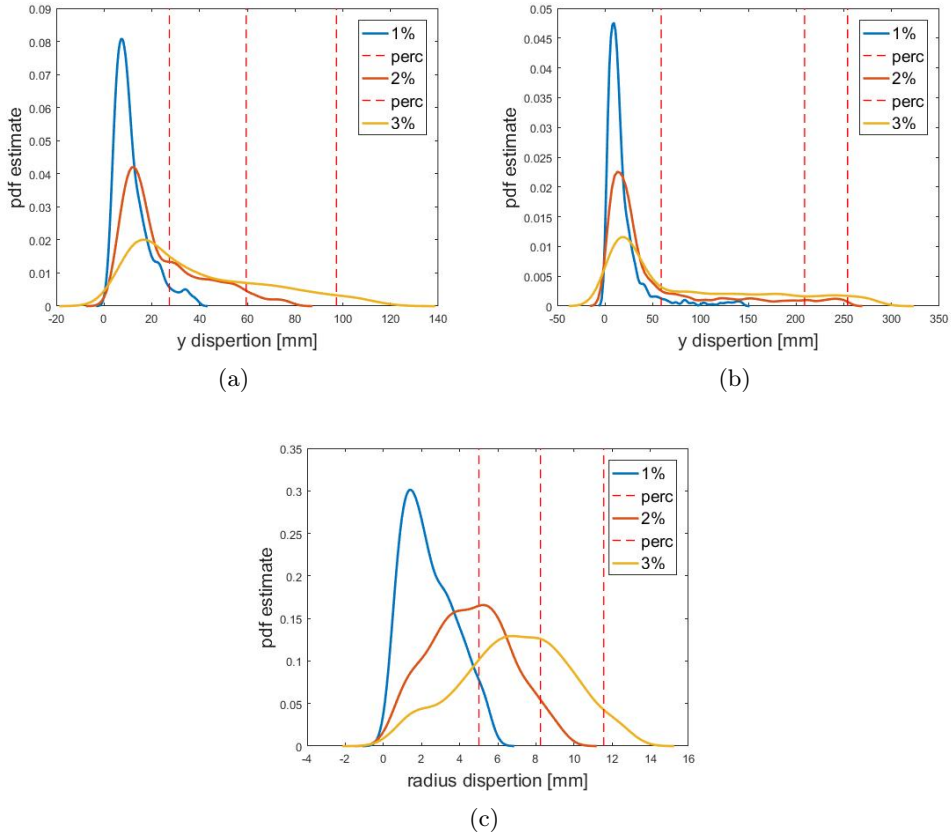
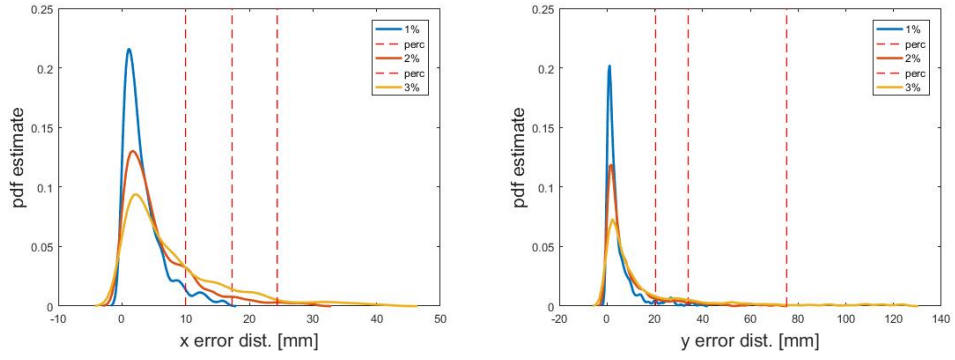


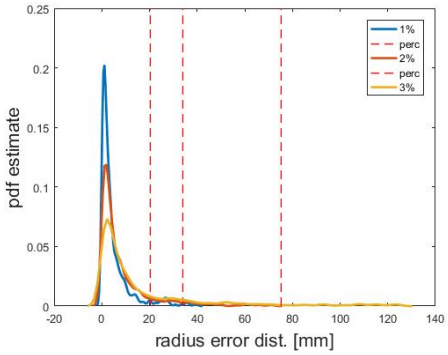
Figure 6.20. (a) x coord. dispersion, (b) y coord. dispersion, (c) radius dispersion, the red dotted line is the 95% percentile, Kriging surrogate model is employed.

difficult to be identified with a good level of accuracy and precision, in fact even though the noise level is high (which means bad measure) but the radius is high as well we still have an acceptable result in terms of localization performance.
About the surrogate model employed, we can observe that there is no a significant difference between using Kriging or ANN committee from a performance point of view, but a significant difference is present in terms of computational time required to localize the damage. In fact the time required to ANN is much higher than Kriging to compute a Markov chain. In table 6.5 are shown the time require for both of them to compute a Markov chain with 5000 samples.



(a)

(b)



(c)

Figure 6.21. (a) x coord. error distribution, (b) y coord. error distribution, (c) radius error distribution, the red dotted line is the 95% percentile, ANN surrogate model is employed.

Kriging time [s]	ANN time [s]
52.8	377.4

Table 6.5. Time required to compute an entire Markov chain.

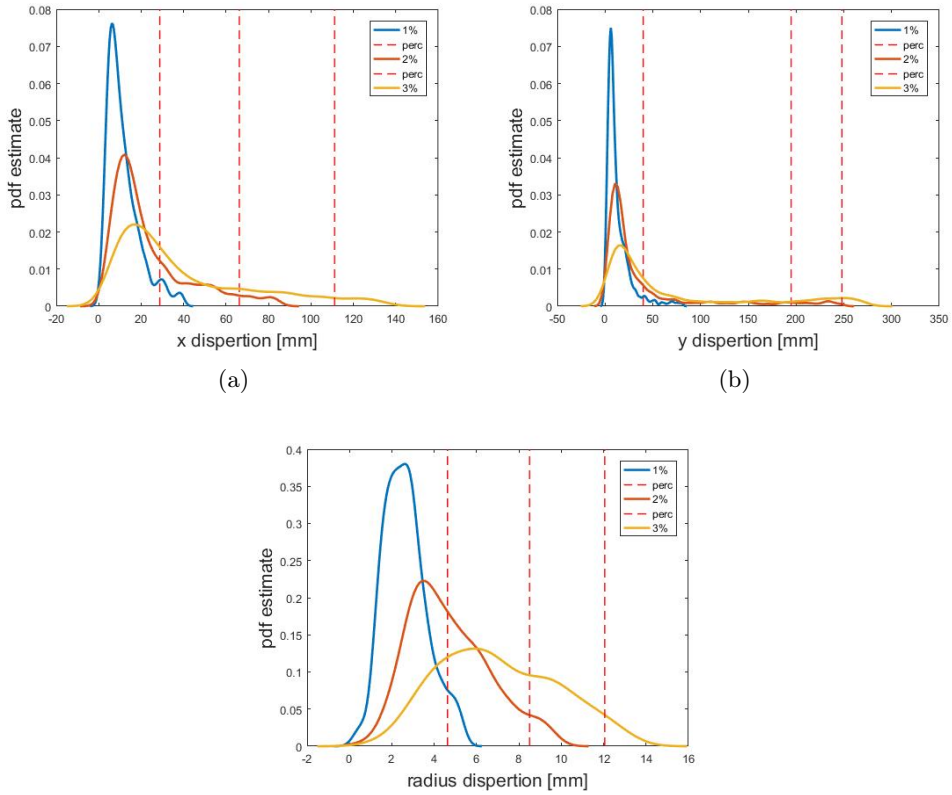


Figure 6.22. (a) x coord. dispersion, (b) y coord. dispersion, (c) radius dispersion, the red dotted line is the 95% percentile, ANN surrogate model is employed.

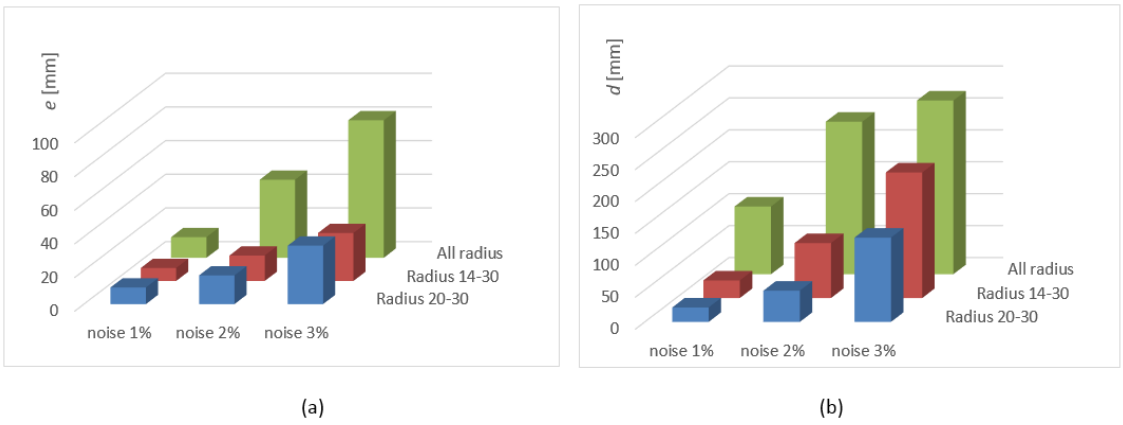
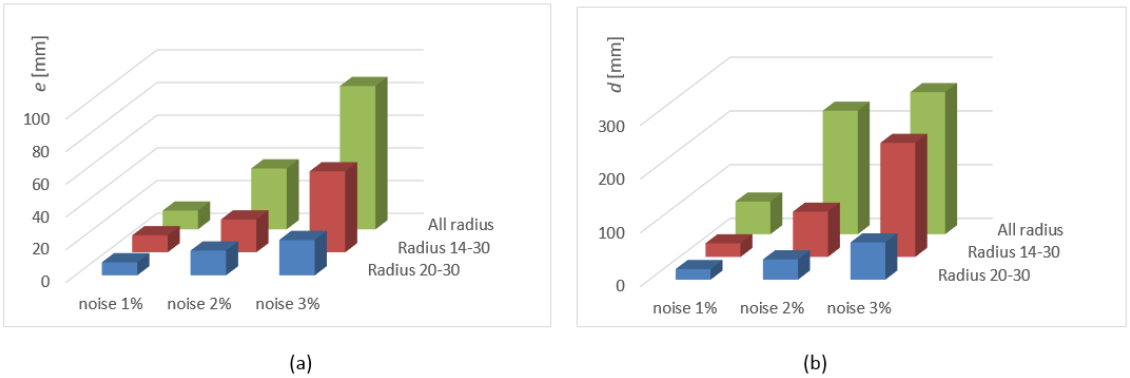
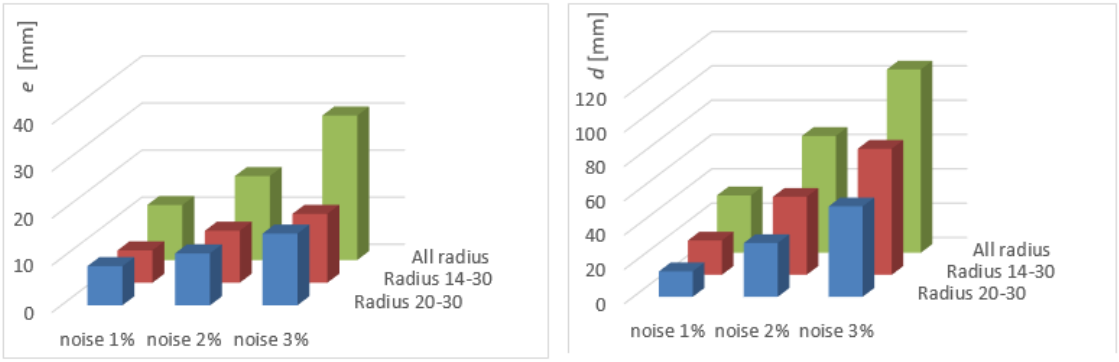


Figure 6.23. (a) Error percentile trend along the y coordinate for different noise level and delamination dimensions considered using Kriging surrogate model, (b) dispersion percentile trend along the y for different noise level and delamination dimensions considered using Kriging surrogate model.



(a) (b)

Figure 6.24. (a) Error percentile trend along the y coordinate for different noise level and delamination dimensions considered using ANN committee surrogate model, (b) dispersion percentile trend along the y for different noise level and delamination dimensions considered using ANN committee surrogate model.



(a) (b)

Figure 6.25. (a) Error percentile trend along the x coordinate for different noise level and delamination dimensions considered using Kriging surrogate model, (b) dispersion percentile trend along the y for different noise level and delamination dimensions considered using Kriging surrogate model.

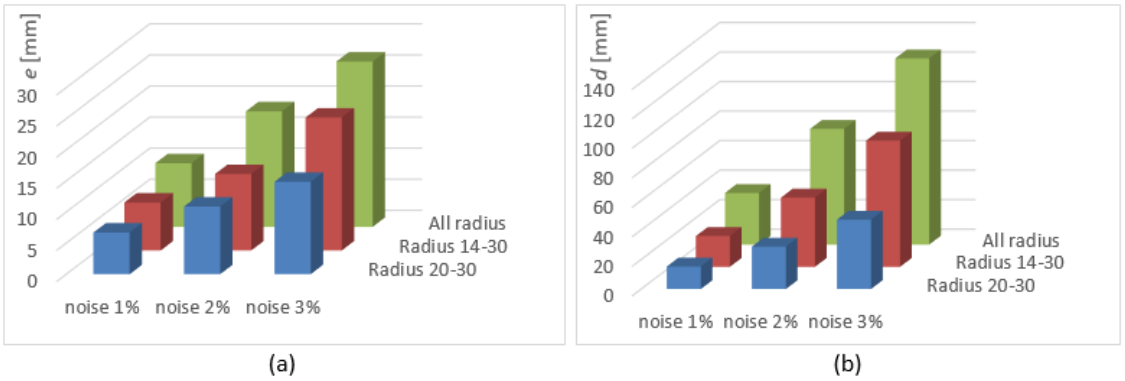


Figure 6.26. (a) Error percentile trend along the x coordinate for different noise level and delamination dimensions considered using ANN committee surrogate model, (b) dispersion percentile trend along the x for different noise level and delamination dimensions considered using ANN committee surrogate model.

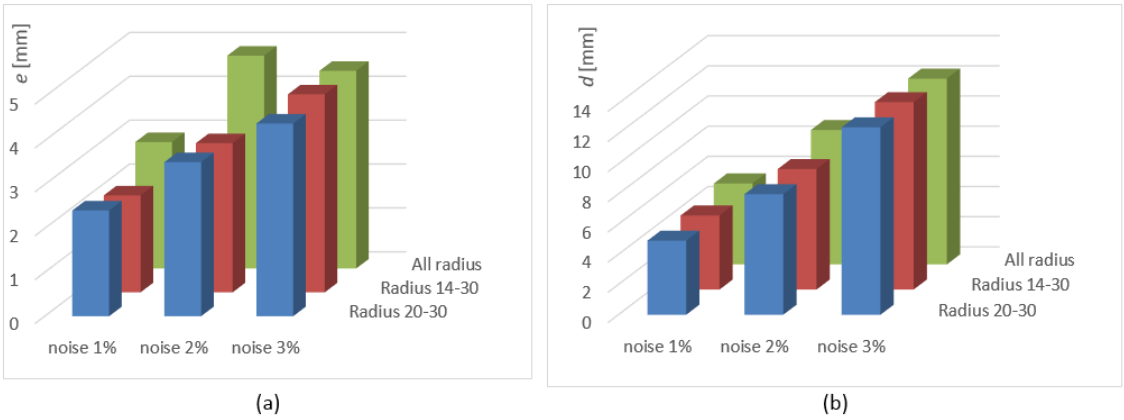


Figure 6.27. (a) Error percentile trend of the radius r for different noise level and delamination dimensions considered using Kriging surrogate model, (b) dispersion percentile trend along the radius r for different noise level and delamination dimensions considered using Kriging surrogate model.

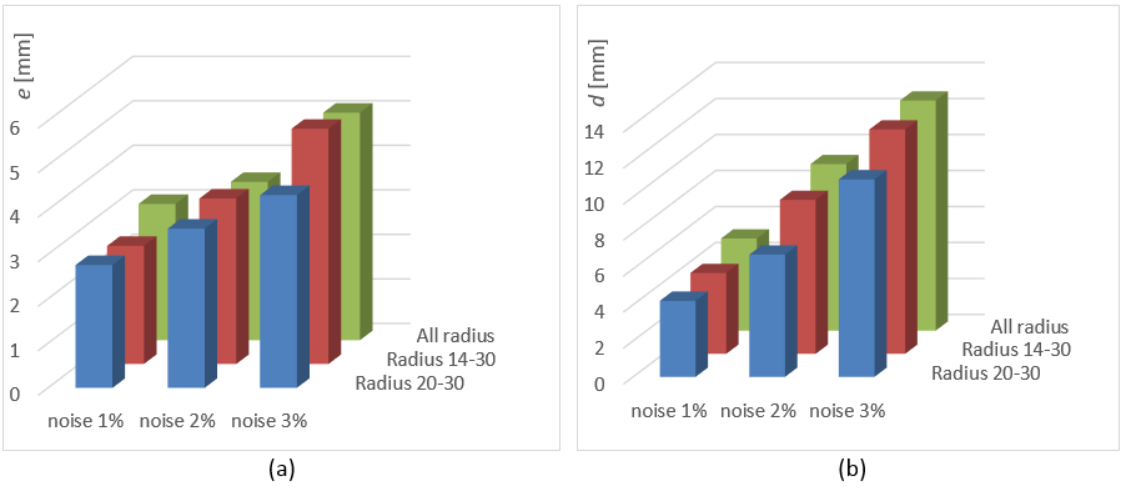


Figure 6.28. (a) Error percentile trend of the radius r for different noise level and delamination dimensions considered using ANN committee surrogate model, (b) dispersion percentile trend along the radius r for different noise level and delamination dimensions considered using ANN committee surrogate model.

Chapter 7

Conclusion

7.1 Summary of the algorithm and results

In this final chapter a review of all observations caught over the entire work has been done, furthermore some potential future development will be proposed.
First it is useful to recall the elements of novelty of the work: coupling surrogate models to a MCMC algorithm in order to localize and quantify a delamination in a CFRP and analyze the performances of the system in terms of accuracy and precision as a function a the damage size. The usage of surrogate models instead of a numerical tool such as FE method is indispensable in order to reduce the computational time and make the real-time diagnosis possible.
The algorithm can be summarized as follow: at each iteration a sample is drawn attempting to identify the damage parameters (position and dimension). Each time a sample is drawn, a likelihood function is evaluated, this function gives an indication of the similarity between the real strain measurements and those simulated by the surrogate model receiving as input the damage parameters just drawn. In this work two surrogate models are employed: Kriging and artificial neural networks committee. Iteration after iteration it is possible to build up a chain of samples which converges toward the support partition with the highest likelihood, enabling to solve the inverse problem. It is important to remind that the damage localization has been applied under a probabilistic framework and not deterministic, and the real-time application is guaranteed by the surrogate model and not by the physics-based model.
Both ANN and Kriging were trained using the same training points, and scatter plots were done upon the same testing points. These graphs show that ANN committee has a slight better learning performance than Kriging. This is due to the fact that Kriging, being a kernel based algorithm, may show a lack of generalization when it has to deal with high variability func-

tions, this situation can occur with large dimension delaminations located near the sensors.
To check the ability of the framework to localize a delamination, several delamination cases smeared with random noise of various levels have been run. For each case the error between the real damage parameter values and the estimated ones by the MCMC was computed together with the interval in which 95% of the data were included. The next step was to make a statistical analysis of these results as shown at the end of chapter 6. The results show that as the noise level which affects the measurements increases, both accuracy and precision become worst. This fact is particularly accentuate if we have to localize small size delaminations, while if the delamination size becomes higher we still have an acceptable results although the noise level may be high.
Although the performances in terms of accuracy and precision are quite similar between ANN committee and Kriging, a big discrepancy is present in terms of identification time, in fact implementing a Kriging surrogate model in a Markov-Chain Monte Carlo algorithm requires a time over than five times lower than using ANN committee.
Another main differences between ANN committee and Kriging are, in my opinion, constituted by the implementation process of these models, rather than the detection performance. ANN committee, other than having a longer Markov-Chain time consumption, requires a longer training time, in fact 20 ANN must be trained (for each sensor) with respect just one Kriging model to learn the same function. On the other hand being Kriging based on a matrix inversion, the training set cannot be too large (not higher than 10000), while ANN does not show this kind of problem (it must be said, however, that there exist some approximation techniques, such as the 'Subset of Regressors', which enable Kriging to handle also large datasets).

7.2 Future development

A list of possible future developments is proposed:

- Verify experimentally that the damage simulated in the panel could be real, in fact the sensitivity achieved in this work depends upon the values of damage induced in the fiber and the matrix, if the fiber and matrix damages are much different the sensitivity will change a lot.
- Validate experimentally the diagnostic system.
- Incorporate to this framework a prognostic one through a particle filter, in which a damage evolution model is included, able to make prognosis and to estimate the RUL of the component.

References

[1] S. Sanchez-Saez, E. Barbero, R. Zaera, and C. Navarro, "Compression after impact of thin composite laminates," *Composites Science and Technology*, vol. 65, no. 13, pp. 1911–1919, 2005.

[2] T. Loutas, N. Eleftheroglou, and D. Zarouchas, "A data-driven probabilistic framework towards the in-situ prognostics of fatigue life of composites based on acoustic emission data," *Composite Structures*, vol. 161, pp. 522–529, 2017.

[3] J. Yang, G. Sha, Y. Zhou, G. Wang, and B. Zheng, "Statistical pattern recognition for structural health monitoring using esn feature extraction method," *International Journal of Robotics and Automation*, vol. 33, no. 6, pp. 569–576, 2018. cited By 0.

[4] A. Mondal, Y. Efendiev, B. Mallick, and A. Datta-Gupta, "Bayesian uncertainty quantification for flows in heterogeneous porous media using reversible jump markov chain monte carlo methods," *Advances in Water Resources*, vol. 33, no. 3, pp. 241 – 256, 2010.

[5] Y. Huang, C. Shao, B. Wu, J. Beck, and H. Li, "State-of-the-art review on bayesian inference in structural system identification and damage assessment," *Advances in Structural Engineering*, vol. 22, no. 6, pp. 1329–1351, 2019. cited By 3.

[6] C. Jiang, H. Qiu, L. Gao, D. Wang, Z. Yang, and L. Chen, "Real-time estimation error-guided active learning kriging method for time-dependent reliability analysis," *Applied Mathematical Modelling*, vol. 77, pp. 82–98, 2020. cited By 4.

[7] P. Wei, C. Tang, and Y. Yang, "Structural reliability and reliability sensitivity analysis of extremely rare failure events by combining sampling and surrogate model methods," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 233, no. 6, pp. 943–957, 2019. cited By 0.

[8] X. Zhao, H. Gao, G. Zhang, B. Ayhan, F. Yan, C. Kwan, and J. L. Rose, "Active health monitoring of an aircraft wing with embedded piezo-

electric sensor/actuator network: I. defect detection, localization and growth monitoring," *Smart Materials and Structures*, vol. 16, pp. 1208–1217, jun 2007.

[9] C. Boller and N. Meyendorf, "State-of-the-art in structural health monitoring for aeronautics."

[10] J. Lopez-Higuera, L. Rodriguez, A. Quintela, A. Cobo, F. Madruga, O. M. Conde, M. Lomer, and J. MIrapeix, "Fiber optics in structural health monitoring," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 7853, 11 2010.

[11] O. K. Ihesiulor and Z. Zhang, "Effectiveness of artificial neural networks and surrogate-assisted optimization techniques in delamination detection for structural health monitoring," in *10th IASTED International Conference on Visualization, Imaging, and Image Processing, July*, pp. 3–5.

[12] R. J. Barthorpe, *On model-and data-based approaches to structural health monitoring*. PhD thesis, University of Sheffield, 2010.

[13] D.-A. Tibaduiza, M.-A. Torres-Arredondo, L. Mujica, J. Rodellar, and C.-P. Fritzen, "A study of two unsupervised data driven statistical methodologies for detecting and classifying damages in structural health monitoring," *Mechanical Systems and Signal Processing*, vol. 41, no. 1, pp. 467 – 484, 2013.

[14] E. Z. Moore, J. M. Nichols, and K. D. Murphy, "Model-based shm: Demonstration of identification of a crack in a thin plate using free vibration data," *Mechanical Systems and Signal Processing*, vol. 29, pp. 284 – 295, 2012.

[15] F. Lorenzoni, F. Casarin, M. Caldon, K. Islami, and C. Modena, "Uncertainty quantification in structural health monitoring: Applications on cultural heritage buildings," *Mechanical Systems and Signal Processing*, vol. 66-67, pp. 268 – 281, 2016.

[16] S. S. Kessler, S. Spearing, M. J. Atalla, C. E. Cesnik, and C. Soutis, "Damage detection in composite materials using frequency response methods," *Composites Part B: Engineering*, vol. 33, no. 1, pp. 87 – 95, 2002.

[17] A. D'Alessandro, M. Rallini, F. Ubertini, A. L. Materazzi, and J. M. Kenny, "Investigations on scalable fabrication procedures for self-sensing carbon nanotube cement-matrix composites for shm applications," *Cement and Concrete Composites*, vol. 65, pp. 200 – 213, 2016.

[18] P. Diaz Montiel, L. Escalona, and S. Venkataraman, "Exploration of surrogate models for inverse identification of delamination cracks in composites using electrical resistance tomography," in *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, p. 0199, 2017.

[19] J. E. Warner, J. D. Hochhalter, W. P. Leser, P. E. Leser, and J. A. Newman, "A computationally-efficient inverse approach to probabilistic strain-based damage diagnosis," 2016.

[20] C. Sbarufatti, A. Manes, and M. Giglio, "Performance optimization of a diagnostic system based upon a simulated strain field for fatigue damage characterization," *Mechanical Systems and Signal Processing*, vol. 40, pp. 667–690, 11 2013.

[21] X. Yang, X. Guo, H. Ouyang, and D. Li, "A kriging model based finite element model updating method for damage detection," *Applied Sciences*, vol. 7, p. 1039, 10 2017.

[22] T. Mukhopadhyay, S. Naskar, S. Dey, and S. Adhikari, "On quantifying the effect of noise in surrogate based stochastic free vibration analysis of laminated composite shallow shells," vol. 140, 4 2016.

[23] G. Yan, "A bayesian approach for impact load identification of stiffened composite panel," *Inverse Problems in Science and Engineering*, vol. 22, no. 6, pp. 940–965, 2014.

[24] T. Peng, A. Saxena, K. Goebel, Y. Xiang, S. Sankararaman, and Y. Liu, "A novel bayesian imaging method for probabilistic delamination detection of composite materials," *Smart Material Structures*, vol. 22, pp. 5019–, 12 2013.

[25] R. Ghiasi, M. R. Ghasemi, and M. Noori, "Comparative studies of metamodeling and ai-based techniques in damage detection of structures," *Advances in Engineering Software*, vol. 125, pp. 101 – 112, 2018.

[26] F. Cheng, J. Yu, and H. Xiong, "Facial expression recognition in jaffe dataset based on gaussian process classification," *IEEE Transactions on Neural Networks*, vol. 21, pp. 1685–1690, Oct 2010.

[27] M. Seeger, "Gaussian processes for machine learning," *International journal of neural systems*, vol. 14, no. 02, pp. 69–106, 2004.

[28] C. M. Bishop, *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

[29] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA, 2006.

[30] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[31] D. Gamerman, "Sampling from the posterior distribution in generalized linear mixed models," *Statistics and Computing*, vol. 7, no. 1, pp. 57–68, 1997.

[32] A. Smith, *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2013.

[33] P. Diaconis and L. Saloff-Coste, "Walks on generating sets of groups," *Inventiones mathematicae*, vol. 134, no. 2, pp. 251–299, 1998.

[34] C. Robert and G. Casella, *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[35] H. Haario, E. Saksman, and J. Tamminen, "Adaptive proposal distribution for random walk metropolis algorithm," *Computational Statistics*, vol. 14, no. 3, pp. 375–396, 1999.

[36] T.-W. Shyr and Y.-H. Pan, "Impact resistance and damage characteristics of composite laminates," *Composite structures*, vol. 62, no. 2, pp. 193–203, 2003.

[37] E. Panettieri, D. Fanteria, and F. Danzi, "Delaminations growth in compression after impact test simulations: Influence of cohesive elements parameters on numerical results," *Composite Structures*, vol. 137, 11 2015.