MAREK SŁOŃSKI

BAYESIAN MACHINE LEARNING IN ANALYSIS OF SELECTED IDENTIFICATION PROBLEMS IN MECHANICS OF MATERIALS AND STRUCTURES
Contents

Notation 5

1 Introduction 7

2 Bayesian machine learning 11
   2.1 Machine learning .............................................. 11
   2.2 The basics of Bayesian inference ............................ 13
   2.3 Bayesian inference for machine learning ................. 15
      2.3.1 Example of Bayesian inference for a linear regression model 15
      2.3.2 Approximate methods for Bayesian inference .......... 27

3 Concrete properties prediction 31
   3.1 Introduction .................................................... 31
   3.2 Problem formulation ......................................... 32
   3.3 Nonlinear models for regression ............................ 33
      3.3.1 The Gaussian process ................................. 33
      3.3.2 Bayesian neural networks ............................ 35
   3.4 Prediction of plain concrete fatigue failure .......... 40
      3.4.1 Introduction ........................................... 40
      3.4.2 Data description and analysis ....................... 42
      3.4.3 Computer experiments ............................... 43
      3.4.4 Discussion of the results ........................... 45
   3.5 Prediction of HPC compressive strength ............... 49
      3.5.1 Introduction ........................................... 49
      3.5.2 Data description and analysis ....................... 50
      3.5.3 Computer experiments ............................... 53
      3.5.4 Presentation of the results ......................... 56
   3.6 Remarks ....................................................... 60
## 4 Structural damage identification

4.1 Problem description ........................................... 63
  4.1.1 Damage identification ................................... 63
  4.1.2 Damage detection ...................................... 64
  4.1.3 Damage localization .................................... 65
4.2 Benchmark steel frame ...................................... 66
  4.2.1 Frame description ...................................... 66
  4.2.2 Simulation results .................................... 66
4.3 Damage-sensitive features extraction and visualization ......... 69
  4.3.1 Autoregressive model for features extraction .............. 69
  4.3.2 Dimensionality reduction and visualization ............... 69
4.4 Damage detection with a Gaussian mixture model ............... 70
  4.4.1 A Gaussian mixture model and the EM algorithm .......... 70
  4.4.2 Computer experiments .................................. 74
  4.4.3 Presentation and discussion of the damage detection results ........................................ 78
4.5 Damage localization using classification models ............... 79
  4.5.1 Classification models .................................. 79
  4.5.2 Computer experiments for three cases of damage ......... 82
  4.5.3 Computer experiments for the six cases of damage ....... 85
4.6 Remarks ....................................................... 87

## 5 Sequential identification of elastic properties

5.1 Introduction .................................................. 89
5.2 Sequential parametric identification .......................... 91
  5.2.1 State space models .................................... 92
  5.2.2 Bayesian state estimation .............................. 93
  5.2.3 The Kalman filter ..................................... 94
  5.2.4 The particle filter .................................... 96
  5.2.5 Sequential parametric identification algorithm ......... 97
5.3 Estimation of Young’s modulus using Bayesian filters ......... 100
  5.3.1 Problem description ................................... 100
  5.3.2 Computer experiments .................................. 101
  5.3.3 Presentation and discussion of the estimation results .... 102
5.4 Estimation of elastic parameters of plates with the particle filter ........................................ 103
  5.4.1 Problem description ................................... 103
  5.4.2 Lamb waves for elastic plates .......................... 106
  5.4.3 Computer experiments for Young’s modulus identification ........................................ 108
  5.4.4 Computer experiments for Young’s modulus and Poisson’s ratio identification .................. 112
5.5 Closing remarks ............................................. 114
## Final conclusions

6.1 General remarks ........................................ 117

6.2 Summary of results ....................................... 118

6.2.1 Prediction of concrete properties .................. 118

6.2.2 Structural damage detection and localization .... 119

6.2.3 Sequential estimation of elastic constants ......... 119

6.3 Original contribution .................................... 119

6.4 Practical applications ................................... 120

6.5 Future work ............................................. 120

### Bibliography

121

### List of Figures

135

### List of Tables

139

### Summary

141

### Summary in German

142

### Summary in Polish

143
Notation

Below is the list of basic acronyms and symbols used in this monograph.

Acronyms

APE – Average Percentage Error
ANN – Artificial Neural Network
BNN – Bayesian Neural Network
SNN – Standard Neural Network
S-BNN – Semi-Bayesian Neural Network
T-BNN – True-Bayesian Neural Network
EM – Expectation Maximization
FEM – Finite Element Method
GMM – Gaussian Mixture Model
GP – Gaussian Process
MAP – Maximum a Posteriori
ML – Maximum Likelihood
MML – Maximum Marginal Likelihood
PCA – Principal Component Analysis
RMSE – Root Mean Square Error
RVM – Relevance Vector Machine
SVM – Support Vector Machine

Symbols

\( t \) – vector of target outputs,
\( w \) – vector of model parameters,
\( x \) – vector of model inputs,
\( y \) – vector of model outputs,
\( E \) – Young’s modulus,
\( H \) – number of NN hidden neurons,
\( L \) – number of learning patterns,
\( P \) – total number of patterns,
\( T \) – number of testing patterns,
\( V \) – number of validation patterns,
\( r \) – Pearson’s correlation coefficient
\( \nu \) – Poisson’s ratio
Chapter 1

Introduction

Many scientific and engineering problems in mechanics of materials and structures can be formulated and solved as identification problems. Identification problems belong to the class of inverse problems [16, 63, 70, 122]. Solving inverse problems means trying to find the causes (inputs) knowing the model and the effects (outputs) or trying to find the model knowing the causes and the corresponding effects [88, 136]. In nondestructive evaluation of structures, we are trying to find the changes in the model (for example possible changes in a structure resulting from a local damage) knowing the causes (loads) and the effects in the form of measurements (for example displacements of selected points of the structure) [13, 68].

In this context, many identification problems can be viewed as problems that can be solved by using machine learning methods [29]. Machine learning is a branch of artificial intelligence with a fast-growing impact on science, technology and everyday life [71, 103]. Machine learning methods solve identification problems by automatically learning (estimating) a computer program parameters (model parameters) from available data (for example experimental data) instead of programming it by hand [19]. Another description says that the field of machine learning is concerned with the automatic discovery of patterns in data through the use of computer algorithms and with the use of these patterns to take actions such as classifying the data into different categories or prediction of numerical values for a given set of inputs. Moreover, machine learning aims to learn a useful representation of data which allows us to obtain optimal predictions and make optimal decisions [11, 97].

For example, in structural health monitoring, we can use machine learning methods and a large number of data from a monitored structure working in nominal (normal) conditions to learn a model (also known as a normality model). Subsequently, this model can be used for detection of anomalous behavior of the structure (also known as novelty detection), possibly caused by a local, invisible damage or a fault [42, 74].
One of the most popular and useful types of machine learning methods is based on various kinds of artificial neural networks (ANNs) [10, 41]. ANN-based methods have proved to be very useful tools for solving identification problems in the context of mechanics of materials and structures [56, 61, 86, 87, 141, 142, 143]. For example, feed-forward neural networks are used for identification of concrete properties based on a known mix design and experimental results of concrete properties like compressive strength [115, 149].

From the mathematical point of view, the common feature of all identification problems is their ill-conditioning or lack of a unique solution [6]. This feature is mainly caused by uncertainty, which is the effect of modeling and measurement errors and the finite number of data. Various regularization techniques have been developed to overcome the ill-conditioning problem – for example the Tikhonov regularization [139].

One of the most promising approaches to regularization is based on statistical inference [47, 127]. This approach uses Bayes’ theorem to infer posterior probability distribution of unknown random variables (for example model parameters) by combining certain prior knowledge in the form of prior probability distribution and the observed variables in the form of a likelihood function [18]. Moreover, this approach is naturally suited for uncertainty quantification by allowing computing predictive probability distribution of output variables by using probability rules and numerical integration techniques, such as stochastic sampling methods or deterministic variational methods [11]. The Bayesian approach is also sequential in nature, i.e. it allows solving identification problems recursively at every acquisition of measurements [26].

These statistical techniques based on machine learning and Bayesian inference are very general and extremely useful for solving identification problems [81, 82, 83]. This is why they are becoming more and more popular also in the context of mechanics of materials and structures [8, 93].

Despite the growing popularity and the number of applications of machine learning and Bayesian inference for solving identification problems in mechanics of materials and structures, there are still problems waiting to be solved with the use of this approach (especially in Poland). Thus, the main goal of this monograph is to present selected applications of Bayesian machine learning in this context and to show the advantages and disadvantages of the Bayesian inference-based approach.

The first identification problem concerns prediction of concrete mechanical properties based on some input information, such as the amounts of the concrete mix ingredients. The unknown relations between the output variables (the concrete properties) and input variables are modeled by applying nonlinear regression models such as Gaussian processes and Bayesian neural networks.

The second identification problem pertains to structural damage identification.
In particular, we focus on two related problems of structural damage detection and damage localization. To solve these two problems, we apply the vibration-based approach and machine learning methods for classification.

In the third identification problem, we investigate the application of Bayesian filtering to sequentially estimate elastic properties of materials and structures based on some observed variables, like displacements in the case of a laboratory-scale portal aluminum frame, and some characteristics, like dispersion curves in the case of thin aluminum plates.

The monograph is organized as follows:

– Chapter 1 first presents the introductory remarks and then goes on to discuss the motivation and subject of the monograph. The objectives of the work are introduced in the next section, followed by the scope of the book. Finally, the chapter closes with some remarks concerning the notation used in this work.

– Chapter 2 gives the basics of Bayesian inference in the machine learning context. This chapter starts with some introductory remarks about machine learning. Then Bayesian inference basics are presented. Next, the exact Bayesian inference is demonstrated for a simple linear regression model. Finally, some introductory information is given about approximate methods for Bayesian inference.

– Chapter 3 focuses on the solution of concrete properties prediction problems on the basis of experimental evidence. These problems are formulated as regression problems. In this work, we concentrate on two prediction problems. The first one deals with prediction of plain concrete fatigue failure whereas the other - with prediction of high-performance concrete compressive strength. Searching for the solution of the above problems, we apply two nonlinear regression models, namely Bayesian neural networks and Gaussian processes.

– Chapter 4 demonstrates the application of classification models for solution of two vibration-based structural damage identification problems in a benchmark steel frame. In the first problem, we use Gaussian mixture model for solution of structural damage detection. In the second problem, we apply a set of Gaussian mixture models and two classification models, namely generalized linear models and relevance vector machine, for solution of structural damage localization.

– In Chapter 5, we present how the Bayesian state estimation could be applied for solution of parametric identification problems using Bayesian filters. In this work, we concentrate on solving two identification problems of sequential estimation of elastic properties of materials. In the first problem, we apply the
Kalman filter and a particle filter for sequential estimation of Young’s modulus of a laboratory aluminum frame based on comparison of the measured and the predicted displacements of selected nodes of the frame. In the second problem, we use a particle filter to estimate elastic properties of aluminum plates by comparing the measured and the predicted dispersion curves obtained from Lamb waves.

– Finally, Chapter 6 summarizes and concludes the monograph as well as presents some directions for further study and prospective applications of Bayesian machine learning methods in the context of mechanics of materials and structures.

Acknowledgments

This monograph summarizes the research the author carried out during years 2003-2013, yet it is largely based on published papers, listed in the Bibliography. This stage of progress would not have been achieved without the constant support of Prof. Zenon Waszczyszyn from Cracow University of Technology.

Parts of this work have been financially supported by Polish Ministry of Science and Higher Education within the projects: “Application of Bayesian neural networks in experimental mechanics of structures and materials”, No N506 1814 33, and “Applications of Bayesian machine learning methods in identification problems of experimental mechanics of materials and structures”, No N N506 2509 38.

Most of the calculations have been carried out with the scripts written in MATLAB by the Author. They were based on various open source toolboxes (mainly NETLAB [72] and MCMCStuff [137]).
Chapter 2

Bayesian machine learning

In this chapter, we give a short overview of machine learning and Bayesian inference. We start by presenting some introductory remarks about the field of machine learning and some examples of its current applications in the context of identification problems. Then we present the basics of Bayesian inference by giving some introductory remarks on the Bayesian view of probability as well as introducing two fundamental probability rules and Bayes’ theorem. Next we give a simple example of application of Bayesian inference for a linear regression model. We show how Bayesian inference can be useful for uncertainty quantification in the context of model parameters estimation, making predictions and model selection. We end the chapter with some remarks on approximate methods for Bayesian inference in machine learning.

2.1 Machine learning

In very short words, machine learning (ML) is about inferring a function from a collection of input-output pairs and then applying of the inferred function to the prediction of the output for new inputs [11, 103]. It is thus closely related to the field of statistical pattern recognition [27, 33, 144]. In this context, machine learning is a set of data-driven methods that can automatically detect regularities in data and then use them to predict future outcomes that are of interest in a particular situation [7, 71, 97].

Historically, the first paper on machine learning was published by A. Samuel in 1959 about a computer which managed to learn how to play checkers [106]. Samuel defined machine learning as a "field of study that gives computers the ability to learn without being explicitly programmed". A more formal definition was provided by T. Mitchell: "A computer program is said to learn from experience E with respect to
some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$" [69].

Currently, there are an extremely large number of practical applications of machine learning in science, engineering and technology [31, 133]. For example, machine learning methods are used for detection and classification of objects, like people’s faces, handwritten digits or road signs in images [129]. Machine learning methods are also used for prediction tasks, like house prices, or time series forecasting, like the stock market indexes [1].

Machine learning is a large field, and – broadly speaking – the main three subfields of ML are:

- unsupervised learning,
- supervised learning,
- reinforcement learning.

In unsupervised learning, learning is based on only the inputs. The most common unsupervised learning tasks are clustering, density estimation and dimensionality reduction of the input space. For example, in the structural damage detection context, we can use the density estimation approach to find the probability distribution which can represent a structure in the normal (healthy) condition. Similarly, we can apply dimensionality reduction techniques, like principal components analysis (PCA), to reduce the size of the input space without losing much of the information contained in the original space.

In supervised learning, learning is based on input-output pairs and the goal is to learn a function that maps from input to output. The most common supervised learning tasks are regression and classification. For example, the task of prediction of concrete properties based on the mix design can be viewed as a regression problem in which the input is a vector consisting of amounts of cement, water, coarse aggregates etc. and the output is a vector of quality properties of concrete, like compressive strength. Similarly, the task of localizing damage in a structure can be formulated as a classification problem in which the input is a vector of damage-sensitive features and the output is a vector representing possible localizations of damage. In all previous problems we have assumed that the datasets contain data items which are identical and independently distributed. In some problems this assumption is not valid. For example, in the analysis of time series we assume that the data points are correlated. The most common approach to estimation of parameters of machine learning models is based on batch approach. The other approach is based on on-line or sequential approach. Finally, in reinforcement learning, learning is based on a series of reinforcements (rewards and penalties). In practice, these distinctions are sometimes not so crisp [103].
2.2 The basics of Bayesian inference

One of the main trends in the field of machine learning is now concentrated on using statistical inference for solving learning tasks. The statistical approach to machine learning can be viewed as a form of uncertain reasoning from observations. In this approach, the learning task is formulated as a process of probabilistic inferences. In this context, the Bayesian view of learning is extremely powerful, providing a general solution to the problem of noise, overfitting and optimal prediction [103].

2.2 The basics of Bayesian inference

There are two main views of probability: the classical or frequentist view and the more general, Bayesian view. In the frequentist interpretation of probability, probability is defined as the fraction of times that an event occurs out of the total number of trials, in the limit that the total number of trials goes to infinity

\[ P = \lim_{N \to \infty} \frac{N_1}{N}, \quad (2.1) \]

where \( N_1 \) is the number of trials that the event occurs out of the total number of \( N \) trials.

However, there are examples of events that cannot be repeated numerous times to define their probabilities, but nevertheless, we are generally able to formulate some idea on the matter. For example, in the context of how quickly we think the polar ice is melting [11]. In this context, Bayesian interpretation of probability provides a quantification of uncertainty.

The basic elements of the probabilistic approach to machine learning are the two elementary rules of probability (known as sum rule and product rule) as well as Bayes’ rule. This approach is used throughout this monograph. A detailed derivation of these rules can be found, for example, in the book by Bishop [11].

We start introducing these rules by defining the probability of a random event. We refer here to the classical interpretation of probability. In order to present the rules of probability, let us consider two discrete random variables \( X \) and \( Y \). We assume that \( X \) can take any of the values \( x_i \), where \( i = 1, \ldots, M \), and \( Y \) can take the values \( y_j \), where \( j = 1, \ldots, N \). The joint probability that \( X \) will take the value \( x_i \) and \( Y \) will take the value \( y_j \) is written as

\[ p(X = x_i, Y = y_j). \quad (2.2) \]

The probability that \( X \) will take the value \( x_i \) can be evaluated by applying the sum rule of probability to the joint probability \( p(X = x_i, Y = y_j) \)

\[ p(X = x_i) = \sum_{j=1}^{N} p(X = x_i, Y = y_j). \quad (2.3) \]
Note that $p(X = x_i)$ is often called the marginal probability, as it is computed by marginalizing the other variables (in this case $Y$).

Similarly, the probability that $Y$ will take the value $y_j$ may be presented by

$$p(Y = y_j) = \sum_{i=1}^{M} p(X = x_i, Y = y_j). \quad (2.4)$$

The case of the probability of $Y = y_j$ conditioned on $X = x_i$ (verbalized as “the probability of $Y$ given $X$”) is called conditional probability and is written as

$$p(Y = y_j | X = x_i). \quad (2.5)$$

In the same way we define the conditional probability $p(X = x_i | Y = y_j)$ of $X$ given $Y$.

The joint probability $p(X = x_i, Y = y_j)$ can be expressed by the product of the conditional probability $p(Y = y_j | X = x_i)$ and the marginal probability $p(X = x_i)$ as

$$p(X = x_i, Y = y_j) = p(Y = y_j | X = x_i)p(X = x_i). \quad (2.6)$$

This relationship is called the product rule of probability.

From the product rule (2.6), together with the symmetry property $p(X,Y) = p(Y,X)$ for the joint probability, we obtain Bayes’ rule, which relates the conditional probabilities $p(Y | X)$ and $p(X | Y)$ in the following form:

$$p(Y | X) = \frac{p(X | Y)p(Y)}{p(X)}. \quad (2.7)$$

Bayes’ theorem can be expressed by means of the following verbalization

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}, \quad (2.8)$$

where the words correspond to terms commonly applied in the Bayesian data analysis [36]. The evidence term in the denominator plays the role of the normalization constant and can be written by means of the sum rule (2.3) and the product rule (2.6)

$$p(X) = \sum_{j=1}^{N} p(X | Y = y_j)p(Y = y_j). \quad (2.9)$$

It is important to note, that Bayes’ theorem is the key element of Bayesian inference. For example, Bayesian inference for a random variable $p(Y = y_j)$, starts from a rough (‘a priori’) probability of the random variable $p(Y = y_j)$. Then, Bayes’ rule (2.7) enables us to compute the ‘a posteriori’ probability $p(Y = y_j | X = x_i)$ having the additional knowledge introduced by the likelihood $p(X = x_i | Y = y_j)$ for the observed event $X = x_i$. 
2.3 Bayesian inference for machine learning

In this section we introduce Bayesian inference in the context of machine learning models. We begin by providing some introductory remarks on Bayesian machine learning. Next, we present the application of Bayesian inference for machine learning via solving a simple regression problem. Finally, we give a short overview of more advanced topics concerning approximate inference methods.

2.3.1 Example of Bayesian inference for a linear regression model

In this section, we introduce Bayesian inference for a regression problem following the description given by Tipping in [135], Bishop in [11] and Waszczyszyn and Słoński in [142].

For regression problems, it is generally assumed that the target variable \( t \) is given by a deterministic function \( y(x; w) \) with an additive random component \( \epsilon \)

\[
\begin{align*}
t = y(x; w) + \epsilon,
\end{align*}
\]

where \( w \) denotes a vector of parameters and \( x \) denotes a vector of input variables. The main goal of regression analysis is to find a regression function \( y(x; w) \) using a dataset of inputs \( X = \{x_1, \ldots, x_N\} \) and the corresponding observations \( t = \{t_1, \ldots, t_N\} \).

In order to illustrate various approaches to estimation of parameters of the linear regression model, a simple numerical example of a sinusoidal curve fitting, discussed in [11], is presented together with the description of the methods under consideration. The mathematical function is the sin curve \( f(x) = \sin 2\pi x \) for \( x \in [0.0, 1.0] \), marked in Fig. 2.1 by a dashed line. Let us consider the case where we observe \( N = 30 \) noisy patterns. These patterns are generated from the known mathematical function \( f(x) \).

Fig. 2.1 shows the plot of the example mathematical function \( f(x) \) and the patterns used for learning, validation and testing.

Linear model for regression

We start by considering a class of functions called linear regression. These models are linear functions in parameters \( w \) and nonlinear functions of the input vector \( x \). In general, the linear model is defined as a linear combination of fixed, nonlinear basis functions of the input variables, in the following form:

\[
\begin{align*}
y(x; w) = \sum_{j=1}^{M-1} w_j \phi_j(x) + w_0,
\end{align*}
\]
Figure 2.1: Plot of the mathematical curve \( f(x) = \sin 2\pi x \) applied to generating \( N = 30 \) patterns used for learning, validation and testing shown using different markers.

where \( \phi_j(x) \) are called basis functions. Equation (2.11) is often written by using an additional dummy ‘basis function’ \( \phi_0(x) = 1 \)

\[
y(x; w = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x). \tag{2.12}
\]

There are many possible choices as regards the basis functions – for example, in the case of a single input variable \( x \), we can use a polynomial basis function in the form of powers of \( x \) so that \( \phi_j(x) = x^j \). The popular choice is the so-called ‘Gaussian’ basis function \( \phi_j(x) = \exp(-r^2/s^2) \), where \( r = ||x - \mu_j|| \). The two parameters \( \mu_j \) and \( s \) are the basis function center and width, respectively.

In our simple numerical example we apply a linear model with fixed polynomial basis functions \( \phi_j(x) = x^j , j = 0, 1, 2, ..., 9 \).
2.3 Bayesian inference for machine learning

Standard least squares approach

One of the classical (deterministic) methods for learning the values of the parameter vector $w$ uses the sum-of-squares error function defined by

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2. \quad (2.13)$$

Minimizing this error function with respect to $w$ gives an estimate $w_{\text{LS}}$, where the subscript LS is added in order to mark that solution (2.14) fully corresponds to that obtained by the application of the least square method. The estimate is then used to make a prediction for a new value of $x_{N+1}$ by evaluating $y(x_{N+1}; w_{\text{LS}})$.

In the case of a linear regression model, the minimizer of (2.13) is obtained in a closed-form via linear algebra

$$w_{\text{LS}} = (\Phi^T \Phi)^{-1} \Phi^T t \equiv \Phi^\dagger t, \quad (2.14)$$

where the design matrix $\Phi$ is composed of $N$ rows vectors $\phi^T_n(x_n)$ corresponding to the number of patterns $n = 1, \ldots, N$, takes the following form:

$$\Phi_{(N \times M)} = \begin{bmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_{M-1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_{M-1}(x_N) \end{bmatrix}, \quad (2.15)$$

and its elements are given by $\Phi_{nj} = \phi_j(x_n)$. The quantity $\Phi^\dagger$ is known as the Moore-Penrose pseudo-inverse of matrix $\Phi$.

Penalized least squares approach

A well-known problem with complex and flexible models is that they can over-fit the training data, leading to poor generalization. One classical (deterministic) technique for avoiding over-fitting is adding a penalty term to the error function (2.13), for example by penalizing large values of the parameters. This approach is known as penalized least squares (PLS). A common regularizer is given by the sum of the squares of the weights, so that

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2 + \frac{\lambda}{2} \|w\|^2, \quad (2.16)$$

In the case of linear regression, the above leads to the following solution

$$w_{\text{PLS}} = [(\Phi^T \Phi) + \lambda I]^{-1} \Phi^T t, \quad (2.17)$$
where $\lambda$ is a regularization coefficient typically optimized by using a separate validation dataset, also called a hold-out dataset [11].

In our simple numerical example, we apply the penalized least squares approach to find the optimal solution for polynomial parameters $w_{PLS}$. The regularization coefficient $\lambda_{opt}$ is found using the validation dataset. Fig. 2.2 shows the plots of the 9th order polynomials which were trained with different values of regularization coefficients $\lambda$. The polynomial corresponding to the optimal value of regularization coefficient $\lambda_{opt} = 1e^{-4}$ is shown as a solid line.

![Figure 2.2: Plots of the 9th order polynomials, trained with different values of $\lambda$; polynomial with the optimal value of $\lambda_{opt} = 1e^{-4}$ is shown as a solid line](image)

Fig. 2.3 presents plots of the root-mean-square errors (RMSE) for training and validation of the polynomial. The training and validation curves are computed for different values of regularization coefficients $\lambda$, selected in advance. These curves show the impact of the regularization coefficient on the learning and validation errors. The validation curve is used for selecting the optimal value of the regularization coefficient $\lambda_{opt}$. This value corresponds to the minimal value of the validation error (see Fig. 2.3).

**Model selection**

Model selection is concerned with selecting the optimal class of models in our problem. For example, in the context of polynomial regression it means selecting
the optimal order of the polynomial. The standard approach is based on selecting a polynomial order corresponding to the minimum of validation error. In our simple numerical example, we use the validation dataset to find the optimal order of the polynomial. Fig. 2.4 shows the plot of the validation curve for selecting the optimal order of the polynomial. The optimal order of the polynomial corresponding to the minimum of validation error is $M_{opt} = 3$. Fig. 2.5 shows the plot of polynomials for different orders $M$ ranging from 0 to 9.

**Probabilistic approach**

In the probabilistic approach to regression, we start with the definition of a probabilistic model of noise term $\epsilon_n$. Here we assume the additive noise model to be a Gaussian distribution defined by two parameters: mean $\mu$ and variance $\sigma^2$, i.e. $p(\epsilon|\mu, \sigma^2) = N(\epsilon|\mu, \sigma^2)$.

Moreover, we assume that the noise has zero mean and is independent and identically distributed (i.i.d.). As a result, the Gaussian noise model can be written as

$$N(\epsilon|\mu, \sigma^2) = N(\epsilon|0, \sigma^2) = (2\pi\sigma^2)^{-1/2} \exp \left[ -\frac{\epsilon^2}{2\sigma^2} \right]. \quad (2.18)$$

Based on this noise model and the assumed relation between the target value and the model output value represented in (2.10), the conditional distribution of the target
Figure 2.4: Plot of the validation curve for selecting the optimal order of polynomial

Figure 2.5: Plot of polynomials for different orders $M$ (from 0 to 9)
variable, given the input variable and the parameters $w$, is also a Gaussian distribution in the following form:

$$p(t|x, w, \sigma^2) = (2\pi\sigma^2)^{-1/2} \exp\left[ -\frac{(t - y(x; w))^2}{2\sigma^2} \right].$$ (2.19)

The joint probability of the data set $t$ is given by the product over all data points of the distribution (2.19) evaluated at the observed data points (since the data points are independent)

$$L \equiv p(t|x, w, \sigma^2) = \prod_{n=1}^{N} p(t_n|x_n, w, \sigma^2) = \prod_{n=1}^{N} (2\pi\sigma^2)^{-1/2} \exp\left[ -\frac{(t_n - y(x_n; w))^2}{2\sigma^2} \right].$$ (2.20)

This joint probability distribution is called the likelihood function (when treated as a function of the parameters vector $w$).

**Maximum likelihood method**

Having defined the likelihood function (2.20), we are now interested in finding the optimal parameters of the regression function by maximizing (2.20) with respect to vector $w$. Equivalently, we can obtain the same results by minimizing the following equation

$$-\ln L = -\ln p(t|x, w, \sigma^2) = \frac{1}{2\sigma^2} \sum_{n=1}^{N} (t_n - y(x_n; w))^2 - \frac{N}{2} \ln(\frac{1}{\sigma^2}) + \frac{N}{2} \ln(2\pi).$$ (2.21)

This is the negative logarithm of the likelihood function and it is commonly used for mathematical convenience and numerical stability.

Setting the gradient of the negative log of the likelihood function to zero gives the following minimizer:

$$w_{ML} = (\Phi^T\Phi)^{-1}\Phi^T t.$$ (2.22)

This point estimate is called – in classical statistics – a ‘maximum likelihood’ estimate (ML) and is denoted by $w_{ML}$. We notice that the standard, ‘least-square’ point estimate (LS) for $w_{LS}$ is equivalent to the ‘maximum likelihood’ estimate (ML) $w_{ML}$. We can also minimize (2.20) with respect to $\sigma^2$ and obtain the following estimate of the noise level

$$\sigma^2_{ML} = \frac{1}{N} \sum_{n=1}^{N} (t_n - y(x_n; w_{ML}))^2.$$ (2.23)
Like the least squares approach, the maximum likelihood approach will also be vulnerable to overfitting. To overcome this problem, instead of the earlier regularization weight term $\frac{1}{2} ||w||^2$, we now turn to the general approach, which assumes that the model parameters are random variables and uses Bayesian inference for finding the posterior distribution over parameters vector.

**Bayesian approach**

In Bayesian approach to regression, we start by defining a prior distribution over model parameters $p(w)$. The prior distribution is assumed here to be a spherical Gaussian distribution (with zero mean and precision (inverse variance) hyperparameter $\alpha$), given by

$$p(w | \alpha) = \prod_{m=1}^{M} \left( \frac{\alpha}{2\pi} \right)^{1/2} \exp \left\{ -\frac{\alpha}{2} w_m^2 \right\}.$$  

(2.24)

This prior distribution describes our a priori preferences for smoother models to minimize the possibility of overfitting. Such models, in general, have smaller values of parameters.

After defining the prior distribution and the likelihood function (2.20), we apply Bayes’ theorem to infer the posterior distribution over parameters

$$p(w | t, \alpha, \beta) = \frac{p(t | w, \beta)p(w | \alpha)}{p(t | \alpha, \beta)}.$$  

(2.25)

where $\beta$ is defined as inverse variance of the Gaussian noise model: $\beta = 1/\sigma^2$. Parameters such as $\alpha$ and $\beta$ are called hyperparameters since they control parameters of probability distributions over model parameters.

The normalizing factor in (2.25) is given by the integral of the numerator in Bayes’ theorem over the parameter space

$$p(t | \alpha, \beta) = \int p(t | w, \beta)p(w | \alpha)dw.$$  

(2.26)

This normalizing factor is also known as marginal likelihood or evidence.

It is important to note that Bayes’ rule allows us to combine two sources of information about the estimated relationship and instead of the point estimate, we have obtained the updated distribution (in the light of the training data) over all possible values of parameters.

Going back to our linear regression example, the posterior distribution is a multivariate Gaussian distribution $p(w | t, \alpha, \beta) = \mathcal{N}(w | m_N, S_N)$, where the mean vector and the covariance matrix are given by

$$m_N = \beta S_N \Phi^T t,$$  

(2.27)
\[ S_N^{-1} = \alpha I + \beta \Phi^T \Phi. \] (2.28)

The Gaussian posterior is the result of assuming the Gaussian prior and the linear regression model with the Gaussian noise model.

Fig. 2.6 shows the plot of sequential Bayesian learning for a simple example of inferring model parameters of a linear polynomial \( y(x; w) = w_1 x + w_0 \). The plots of the likelihood function are presented in the first column. The plots of the prior/posterior distributions are shown in the second column. In the third column are presented the plots of training data and linear models with parameters sampled from prior/posterior distributions.

The first row shows, the prior distribution and a set of linear models with parameters sampled from the prior. The likelihood function, the posterior distribution and a set of linear models with parameters sampled from the posterior are shown after using the first training pattern in the second row. The third row presents the likelihood function, the posterior distribution and a set of linear models with parameters sampled from the posterior after using the second training pattern. The yellow diamond-shaped plot symbol represents the true values of the polynomial parameters for which the training patterns were generated.

**Maximum a posteriori approach**

The maximum a posteriori (MAP) approach can be used to find the point estimate of model parameters \( w_{MAP} \), which is the most probable value under the posterior distribution. Since the numerator in Bayes’ rule is independent of \( w \), this is equivalent to minimizing the negative logarithm of the numerator in Bayes’ rule (2.25). Retaining only those terms which depend on \( w \) gives the following:

\[
    E_{MAP}(w) = - \log p(t|w, \beta) - \log p(w|\alpha) = \frac{\beta}{2} \sum_{n=1}^{N} (t_n - y(x_n; w))^2 + \frac{\alpha}{2} \sum_{m=1}^{M} w_m^2,
\] (2.29)

It is equivalent to the penalized least squares estimate \( w_{PLS} \) with \( \lambda \) in (2.16) given by \( \lambda = \alpha / \beta \) [135].

**Making predictions**

After learning the model parameters, we can make a prediction for the value of \( t_{N+1} \) given a new input vector \( x_{N+1} \). The key distinction between the classical approach and the Bayesian one is in using marginalization. In the non-Bayesian approach with regularization, the point prediction is given by \( y(x_{N+1}; w_{PLS}) \). In the MAP settings, instead of a point prediction, we have a predictive distribution \( p(t_{N+1}|w_{MAP}, \beta) \), taking into account the MAP point estimate for weights \( w_{MAP} \).
In the Bayesian approach, we obtain a predictive distribution by integrating out parameters $w$, i.e., averaging the model probability for $t_{N+1}$ over all possible values of $w$. This distribution incorporates uncertainty over parameters in the light of the training data. The predictive distribution, given a new input vector $x_{N+1}$, is then defined as

$$p(t_{N+1}|t, \alpha, \beta) = \int p(t_{N+1}|w, \beta)p(w|t, \alpha, \beta)dw.$$  (2.30)

It depends on values of hyperparameters $\alpha$ and $\beta$, which, in general, are unknown in advance. Fig. 2.7 shows the plot of predictive distributions for linear models in the case of an increasing number of learning patterns.
Model selection

Bayesian inference can also be applied for the problem of model selection. In the context of the linear regression model, model selection can mean, for example, selection of the polynomial order or the type of basis function. Model selection is based on the evaluation of the log of marginal likelihood, also called evidence, which appears in the denominator of Bayes’ rule (2.25)

\[
\ln p(t|\alpha, \beta) = -E_{MAP}(m_N) - \frac{1}{2} \ln |A| + \frac{M}{2} \ln \alpha + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi), \quad (2.31)
\]

where \(m_N\) denotes the mean vector of the posterior distribution (2.27) and \(A\) denotes the matrix of second derivatives of the error function \(E_{MAP}(w)\) (2.29) with respect to model parameters \(w\).

In our simple numerical example, we use the log of the marginal likelihood criterion to find the optimal order of the polynomial. Fig. 2.8 shows the plot of the log marginal likelihood curve for the optimal order selection of the polynomial. The optimal order of the polynomial corresponding to the maximum of the log marginal likelihood is \(M_{opt} = 3\).
Fully Bayesian approach

We have hitherto treated hyperparameters $\alpha$ and $\beta$ as we have known them. However, in the fully Bayesian framework, we should also take into account uncertainty over the hyperparameters and integrate them out. We first define the prior distributions over hyperparameters $p(\alpha)$ and $p(\beta)$, which are called hyperpriors. Having defined these priors, we obtain the full posterior distribution over weights in the following form:

$$p(w, \alpha, \beta | t) = \frac{p(t | w, \beta)p(w | \alpha)p(\alpha)p(\beta)}{p(t)},$$

where the denominator (normalizing term) in (2.32) is

$$p(t) = \int p(t | w, \beta)p(w | \alpha)p(\alpha)p(\beta)dw \, d\alpha \, d\beta. \quad (2.33)$$

Nevertheless, this integral can not be computed analytically, and we must approximate the integrations using one of the previously developed methods.

Finally, we can make a fully Bayesian prediction based on the posterior distribution (2.32), taking into account all the possible values of model parameters and hyperparameters in the following form:

$$p(t_{N+1} | t) = \int p(t_{N+1} | w, \beta)p(w, \alpha, \beta | t)dw \, d\alpha \, d\beta. \quad (2.34)$$
However, the integral cannot be evaluated analytically, so we have to apply approximate methods.

2.3.2 Approximate methods for Bayesian inference

Approximate methods for Bayesian inference can be divided into two groups: deterministic (type 2 maximum likelihood, Laplace’s method, variational techniques) and stochastic (Monte Carlo methods). The detailed description of these methods can be found in Bishop [11] or MacKay [64].

Evidence approximation

In this section we describe in short the deterministic approach based on type 2 maximum likelihood, which is also called evidence approximation (EA) for hyperparameters \( \alpha \) and \( \beta \). This approach starts with rewriting the full posterior distribution (2.32), using the product rule of probability, as:

\[
p(w, \alpha, \beta | t) = p(w | t, \alpha, \beta)p(\alpha, \beta | t).
\]

(2.35)

The first term is the posterior distribution defined in (2.25). The second term is the posterior distribution for hyperparameters

\[
p(\alpha, \beta | t) = \frac{p(t | \alpha, \beta)p(\alpha)p(\beta)}{p(t)}.
\]

(2.36)

This posterior distribution can be approximated by a \( \delta \)-function at its mode. This approximation is valid if we assume that this distribution is sharply peaked around their most probable values \( \alpha_{MP} \) and \( \beta_{MP} \). Then the predictive distribution in (2.34) is given by

\[
p(t_{N+1} | t) \approx p(t_{N+1} | t, \alpha_{MP}, \beta_{MP}) = \int p(t_{N+1} | w, \beta_{MP})p(w | t, \alpha_{MP}, \beta_{MP})dw.
\]

(2.37)

In order to find these values, we need to compute values of the hyperparameters which maximize the posterior (2.36). If we have no idea which values would be suitable for \( \alpha \) and \( \beta \), then we can use relatively flat prior distributions \( p(\alpha) \) and \( p(\beta) \). In this case, the most probable values of the hyperparameters are obtained by maximizing \( p(t | \alpha, \beta) \) term in (2.25). It is called evidence (also known as marginal likelihood function) and is given in (2.26) by

\[
p(t | \alpha, \beta) = \int p(t | w, \beta)p(w | \alpha)dw.
\]

(2.38)
The maximization of the evidence function can only be done by using another approximation. An approach called evidence approximation uses the posterior distribution over weights approximated by a spherical Gaussian distribution around a mode of the posterior distribution. Then we can find values of the hyperparameters which maximize the evidence by differentiating the log of the evidence given by

$$
\ln p(t|\alpha, \beta) \approx -E(w_{MAP}) - \frac{1}{2} \ln |A| + \frac{M}{2} \ln \alpha + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) \tag{2.39}
$$

with respect to $\alpha$ and $\beta$ and setting the derivatives to zero. Matrix $A$ is the matrix of second derivatives of the negative log posterior distribution and is given by

$$
A = -\nabla \nabla \ln p(w|t, \alpha, \beta) = \alpha I + \beta H, \tag{2.40}
$$

where $H$ is the Hessian matrix with values of the second derivatives of the error function (2.13), evaluated at $w = w_{MP}$. This procedure leads to the following formulae for finding the optimal values of the hyperparameters given by

$$
\alpha_{\text{new}} = \frac{\gamma}{2E_W}, \tag{2.41}
$$

$$
\beta_{\text{new}} = \frac{N - \gamma}{2E_D}, \tag{2.42}
$$

where $\gamma$ is the number of well-determined parameters:

$$
\gamma = \sum_{i=1}^{M} \frac{\lambda_i}{\lambda_i + \alpha}, \tag{2.43}
$$

where $\lambda_i$ is the eigenvalue of the Hessian $H$ matrix, evaluated at $w = w_{MP}$. Similarly, the error terms $E_W$ and $E_D$ are evaluated, setting weights to the most probable values $w_{MP}$. The details of the evidence approximation approach can be found in Bishop [11].

In our simple numerical example, we use the evidence approximation approach to find the optimal values of hyperparameters $\alpha_{\text{opt}}$ and $\beta_{\text{opt}}$. Fig. 2.9 shows plots of the results of finding the optimal hyperparameters $\alpha$ and $\beta$. Fig. 2.10 shows plots of the curve for the computations of optimal $\gamma$ and the ratio of $\alpha$ and $\beta$ corresponding to the regularization coefficient $\lambda$.

**Stochastic sampling methods**

The non-deterministic approach to machine learning is based on stochastic sampling and Monte Carlo methods. In the Monte Carlo approach, we approximate integrals, such as (2.30), by the finite sum in the following form:

$$
p(t_{N+1}|x_{N+1}, t) \approx \frac{1}{m} \sum_{i=1}^{m} p(t_{N+1}|x_{N+1}, w_i), \tag{2.44}
$$
Figure 2.9: Plots of the results of finding the optimal hyperparameters $\alpha$ and $\beta$ using the evidence approximation approach.

Figure 2.10: Plots of the curve for the computations of optimal $\gamma$ (left) and the ratio of $\alpha$ and $\beta$ corresponding to the regularization coefficient $\lambda$ (right).

where $w_i$ are samples of model parameters generated from the posterior distribution $p(w|t)$. The details of the stochastic sampling methods in the context of machine learning can be found in Bishop [11].
Chapter 3

Concrete properties prediction

In this chapter, we consider two identification problems from experimental mechanics of concrete. Such identification problems commonly arise during the concrete mix design phase. The problems are formulated in this work as nonlinear regression problems. We show how two kinds of Bayesian nonlinear regression models, namely the Gaussian process and the Bayesian feed-forward neural network, can be successfully applied for prediction of mechanical quality properties, such as fatigue failure of plain concrete, or compressive strength of high-performance concrete (HPC).

The chapter starts with a short overview of the Gaussian process and Bayesian neural network models. The second section focuses on application of the regression models to the problem of plain concrete fatigue failure prediction. The next section presents the solutions of two benchmark problems related to identification of compressive strength of high-performance concrete. Finally, the chapter ends with some remarks.

3.1 Introduction

Mix design methods for plain concrete and high performance concrete have been an active field of research in the last decades [23, 77]. Standard methods for HPC mix design are based mainly on the previous experience and simple empirical relationships. However, they are rarely capable of sufficiently accurate modeling of the nonlinear relationships between concrete mix ingredients and the final concrete properties. On the other hand, various neural networks models have been applied to model such relationships with partial success, see for example [44, 50, 84, 150]. Jakubek [44] applied fuzzy weighted neural network for compressive prediction. Kasperkiewicz et al. [50] used the fuzzy-ARTMAP neural network for modeling
compressive strength of HPC mixes using only six components. In another study, Yeh [150] demonstrated that a standard feed-forward layered neural network (FLNN) with eight input components, trained on experimental data using gradient descent with momentum, can be an accurate model for HPC strength. Similarly, Oh et al. [84], applied a FLNN model for proportioning concrete mixes.

The results presented in the above papers, have proved that standard neural networks (SNNs) are quite useful. But in general, the complexity of data and/or task may render the development of such models by hand impractical. This is because complex models may easily overfit the noisy data and fail to infer the true process which generated the data. So, there is a strong need for regularization and complexity control methods. Standard approach is to select the number of model parameters manually – by trial and error – and/or by automatic regularization techniques. In this context, Bayesian inference offers a “principled” and practical approach, based on well established theoretical and statistical foundations [11].

Among methodologies for building neural networks models, those founded on the Bayesian approach are especially powerful because: a) they are based on the probability rules; b) they offer the posterior probability distribution of the model parameters conditioned on the available data [10]; c) the predictive distribution can be obtained, once we have the posterior distribution, which gives a description of the prediction uncertainties.

In recent years, there has been a growing interest in probabilistic methods for solving nonlinear regression problems. Today, probabilistic regression techniques, especially those associated with Bayesian inference, are gaining more and more popularity as a convenient tool for finding robust solutions of prediction problems, which arise also in the context of concrete quality properties identification. One of the Bayesian regression models which have been successfully applied for concrete properties prediction are Bayesian neural networks and Gaussian processes [113, 114, 115]. Vehtari and Lampinen [58] built an FLNN model for prediction of plain concrete properties (e.g., compressive strength, density, slump) on 27 given input variables. They successfully applied the Bayesian approach to the learning of the neural model and to the final prediction.

### 3.2 Problem formulation

In this chapter, we consider the problem of concrete properties prediction viewed as a nonlinear regression task. For regression problems, we generally assume that the target values are some noisy output of an unknown process described using functional relationship \( y(x) \) that we want to estimate, so that

\[
  t_n = y(x_n, w) + \epsilon_n, \tag{3.1}
\]
where: $t$ – output (target) variable, $y(x, w)$ – regression function (model), $x$ – vector of input variables, $w$ – vector of adjustable parameters, $\epsilon$ – additive noise process in which values $\epsilon_n$ are assumed to be independent and identically distributed.

We also assume that we have certain experimental evidence in the form of a dataset of $N$ examples (patterns). The dataset consists of $N$ input vectors $\{x_n\}_{n=1}^{N}$ alongside the corresponding $N$ target values $t = \{t_n\}_{n=1}^{N}$.

### 3.3 Nonlinear models for regression

In the previous chapter, we introduced Bayesian inference in the context of the linear model for regression. Linear regression model has some limitations, yet we can overcome these limitations by allowing the basis functions to be adaptive. This approach leads, for example, to a feed-forward layered neural network. Another approach leads to the Gaussian process, which can be derived from the Bayesian viewpoint of linear regression. This model is also closely related to kernel methods, which are obtained by reformulation of linear models in terms of a dual representation and kernel functions [11].

These two models can also be introduced as parametric and nonparametric models for regression. In the parametric approach, we represent the underlying relationship by some function which is parameterized by a finite number of parameters. A feed-forward neural network model is an example of a parametric regression model. A Gaussian process model is a nonparametric model. The nonparametric regression model is defined without any explicit parameterization. It is defined using a kernel function and a set of hyperparameters. Another well-known nonparametric regression method is the spline smoothing method [140].

#### 3.3.1 The Gaussian process

Gaussian process is an example of a class of models known as kernel methods [109, 110]. They are obtained by reformulation of a linear regression model presented in the previous chapter in the form of a dual representation. In this approach, the linear regression model is trained by minimizing a regularized error function (2.16), described in terms of the Gram matrix $K = \Phi\Phi^T$, where $\Phi$ is the design matrix. The design matrix is an $N \times M$ matrix with elements $\Phi_{nm} = \phi_m(x_n)$. The Gram matrix is an $N \times N$ symmetric matrix with elements

$$K_{nm} = \phi(x_n)^T \phi(x_m) = k(x_n, x_m),$$

(3.2)

where $k(x, x') = \phi(x)^T \phi(x')$ is a kernel function.

The prediction for a new input vector $x$ is obtained from

$$y(x) = k(x)^T (K + \lambda I_N)^{-1} t,$$

(3.3)
34 Concrete properties prediction

where \( t = (t_1, \ldots, t_N)^T \) is a vector of training target values. Vector \( k(x) \) is defined by elements \( k_n(x) = k(x_n, x) \).

From the Bayesian inference viewpoint of the linear regression model, the dual representation approach leads to the Gaussian process, where the kernel function is interpreted as a covariance function of the Gaussian process. The covariance function can be any function that will generate a non-negative definite covariance matrix for any ordered set of (input) vectors \((x_1, \ldots, x_n)\) [95]. It is usual to choose the covariance function to be stationary, i.e. such that the condition

\[
k(x, x') = k(x - x')
\]

holds. This means that the location of points \( x \) and \( x' \) does not affect their covariance, just the vector joining them. In this work, we use the squared exponential (SE) covariance function

\[
k(x_n, x_m) = \theta_0 \exp \left( -\frac{1}{2} \sum_{i=1}^{d} \eta_i (x_{ni} - x_{mi})^2 \right) + \theta_2,
\]

which is the exponential of the weighted squared distance between points in \( \mathbb{R}^d \). The SE covariance function has some free parameters, which are called hyperparameters to emphasize that they are parameters of a nonparametric model. Term \( \theta_2 \) controls the vertical offset of the GP model, while \( \theta_0 \) controls the vertical scale of the process. The \( \eta_i \) hyperparameters allow a different distance measure for each dimension.

After defining the covariance function we can make predictions for a new input vector, but it is often necessary to learn the hyperparameters before making a reliable prediction. The simplest approach is similar to the evidence approximation discussed above. For the Gaussian process, we find the most probable hyperparameters of the covariance function by maximizing the log likelihood function given by

\[
\ln p(t|\theta) = -\frac{1}{2} \ln |C_N| - \frac{1}{2} t^T C_N^{-1} t - \frac{N}{2} \ln(2\pi),
\]

where \( C_N \) is the \( N \times N \) covariance matrix with elements given by a sum of two terms: the covariance function \( k(x_n, x_m) \) and the Gaussian noise component represented by the precision hyperparameter \( \beta \)

\[
C(x_n, x_m) = k(x_n, x_m) + \beta^{-1} \delta_{nm}.
\]

Vector \( k \) has elements \( k(x_n, x_{N+1}) \) and scalar \( c \) is \( k(x_{N+1}, x_{N+1}) + \beta^{-1} \). Maximization of the log likelihood function in (3.6) is done by using gradient-based optimization algorithms, such as conjugate gradients. More details on the Gaussian process regression models can be found in Bishop [11] and in Rasmussen and Williams [95].
After learning the hyperparameters of the covariance function we can make prediction for the new input vector $\mathbf{x}$. Here we concentrate on the key result allowing direct application of the Gaussian process for regression. It is the predictive distribution of the target variable $t_{N+1}$ for the new input vector $\mathbf{x}_{N+1}$. It requires evaluation of conditional distribution $p(t_{N+1}|t_N)$, where $t_N$ is a vector of training target values. This conditional distribution for the Gaussian process is a Gaussian distribution with mean and variance given by

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t} \quad (3.8)$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}. \quad (3.9)$$

From equations (3.8) and (3.9) we can see that the Gaussian process regression model is completely defined by covariance function $k(\mathbf{x}_n, \mathbf{x}_m)$. This function allows us to define the situation where for the nearby points $\mathbf{x}_n$ and $\mathbf{x}_m$ in the input space the corresponding values $y(\mathbf{x}_n)$ and $y(\mathbf{x}_m)$ will be more strongly correlated than for dissimilar points [11].

### 3.3.2 Bayesian neural networks

In this section, we introduce Bayesian inference for a feed-forward layered neural network (FLNN) model in the context of regression problems. We start by briefly describing the architecture and the standard learning methods for a FLNN model, known also as a multilayer perceptron (MLP). More details on that model can be found in several textbooks, for example in Bishop and Haykin [10, 11, 41].

A feed-forward layered neural network with one hidden layer of $H$ adaptive units (neurons) can be described in a functional form by the following equation

$$y(\mathbf{x}; \mathbf{w}) = F_{out} \left( \sum_{h=1}^{H} w_h^{(2)} F_h \left( \sum_{j=1}^{D} w_{hj}^{(1)} x_j + w_{0j}^{(1)} \right) + w_0^{(2)} \right), \quad (3.10)$$

where $D$ is the number of input variables, and the set of all weight and bias parameters have been grouped together into vector $\mathbf{w}$. The neural model parameters vector $\mathbf{w}$ consists of $w_{hj}^{(1)}$ which is the first layer weights from the $j$th input to the $h$th hidden unit, $w_h^{(2)}$ which is the second layer weights from the $h$th hidden unit to the scalar output and $w_{0j}^{(1)}$ and $w_0^{(2)}$, which are the bias parameters for the hidden and output units, respectively. This function can be represented in the form of a diagram as shown in Fig. 3.1. The nonlinear function $F_h(\cdot)$ is the activation function for the hidden units. In this work, we use ‘tanh’ activation function, shown in Fig. 3.1, which is defined as

$$F_h(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)}, \quad (3.11)$$
The nonlinear function $F_{out}(\cdot)$ is the activation function for the output units. For regression problems, this function is a linear function of the form $F_{out}(a) = a$.

Standard deterministic methods for learning feed-forward layered neural networks are based on minimization of the sum-of-squares error function $E(w)$

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

with respect to the weight vector $w$ by using gradient-based optimization algorithms, like conjugate gradient method or Levenberg-Marquardt algorithm (LMA) [10]. Minimizing this error function with respect to $w$ gives the least-square (LS) estimate $w_{LS}$, which is used to make a prediction for a new value of $x$ by evaluating $y(x; w_{LS})$. This approach is also equivalent to the probabilistic maximum likelihood (ML) approach presented in Chap. 2 in the context of the linear regression model assuming independent, identically distributed and normally distributed measurement errors.

However, feed-forward layered neural network models easily over-fit the noisy training data and the common approach (to counteract this) is to add a regularizer term to the error function (3.12) for penalizing large values of the network weights, see Eq. (2.16) in Chap. 2. Minimizing this error function with respect to $w$ gives PLS estimate $w_{PLS}$. This approach is also equivalent to the probabilistic maximum a posteriori (MAP) approach presented also in Chap. 2 in the context of the linear regression model assuming an isotropic Gaussian prior probability distribution.
Bayesian inference for feed-forward layered neural networks

The Bayesian approach to neural networks entails integration of the parameters of a neural model instead of searching for a single vector of the parameters. In this approach all parameters are treated as random variables. First, the prior distribution $p(w|\alpha)$ over weights is defined. It is generally assumed that the prior distribution is a spherical Gaussian distribution $N(0, \alpha^{-1}I)$, with the zero mean vector and the inverse variance (precision) matrix defined by hyperparameters $\alpha_k$. Parameter $k$ is used to identify four different groups of network weights ($k = w^{(1)}_h, w^{(1)}_0, w^{(2)}_h, w^{(2)}_0$). For example, a Gaussian prior distribution for the hidden layer weights $p(w^{(1)}_h)$ is defined as follows:

$$p(w^{(1)}_h|\alpha_{w^{(1)}_h}) = \frac{1}{Z_{W(\alpha_{w^{(1)}_h})}} \exp \left( -\frac{\alpha_{w^{(1)}_h}}{2} \| w^{(1)}_h \|^2 \right), \quad (3.13)$$

where $Z_{W(\alpha_{w^{(1)}_h})} = \left( \frac{2\pi}{\alpha_{w^{(1)}_h}} \right)^{W_{h^{(1)}}/2}$ and $W_{h^{(1)}}$ is the number of the hidden layer weights.

A hierarchical Gaussian prior distribution for the hidden layer weights, called Automatic Relevance Determination (ARD), can be applied [64] as well. The ARD prior distribution is an automatic method for determining the relevance of the inputs. In ARD all weights connected to the same input $j$ have the same variance hyperparameters $\alpha_j$. In general, these hyperparameters are important because they control the complexity of the neural network model. The irrelevant inputs should have smaller weights in the connections to the hidden units than more important weights [80].

After defining the prior distribution, the next step is to define the likelihood function by assuming, first, the noise model. In this work, the Gaussian noise model $N(\epsilon_n|0, \beta)$ is adopted, where $\beta = 1/\sigma^2$. Hyperparameter $\beta$ is the inverse of noise variance $\sigma^2$ and is called precision. As follows from (3.1), the conditional probability of observing data value $t$ for the given input vector $x$ is

$$p(t|w, \beta) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{(y(x, w) - t)^2}{2\sigma^2} \right\}, \quad (3.14)$$

where $y(x; w)$ represents a network function as the mean of the conditional distribution, and parameter $\sigma^2$ controls the variance of the noise.

For data points independently drawn from the distribution defined by (3.14), the likelihood function of target data vector $t$, given the network parameters $w$, is defined
Concrete properties prediction

as

\[
p(t|w, \beta) = \prod_{n=1}^{N} p(t_n|w, \beta) = \prod_{n=1}^{N} \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2 \right\}.
\]

(3.15)

After observing the target data vector \( t \), Bayes’ theorem is used to update the posterior probability distribution over weights \( p(w|t, \alpha, \beta) \)

\[
p(w|t, \alpha, \beta) = \frac{p(t|w, \beta)p(w|\alpha)}{p(t|\alpha, \beta)},
\]

(3.16)

where \( p(t|w, \beta) \) is the likelihood function defined in (3.15), and \( p(w|\alpha) \) is the prior distribution. This posterior probability distribution was already defined in Chap. 2 in Eq. (2.25) in the context of the linear regression model. Taking the negative logarithm of the nominator in (3.16), we can also find the maximum a posteriori (MAP) estimate of \( w \), which is equivalent to minimizing error function with a regularization term in (2.16) [10].

The probability distribution in denominator \( p(t|\alpha, \beta) \), is the normalizing factor defined as

\[
p(t|\alpha, \beta) = \int p(t|w, \beta)p(w|\alpha)dw.
\]

(3.17)

This equation defines the marginal likelihood function (also called evidence function). It is used for determination of the hyperparameters and also for model selection, for example, selection of the optimal number of neurons in the hidden layer [10].

**Predictive distribution**

What is of main interest for regression problems is making the prediction of \( t_{N+1} \) for a new value of \( x_{N+1} \). In the Bayesian approach, instead of point prediction, the predictive distribution over target variable \( t_{N+1} \) is computed applying the sum rule of probability and marginalizing out the weight vector \( w \). This leads to the following equation:

\[
p(t_{N+1}|x_{N+1}, t, \alpha, \beta) = \int p(t_{N+1}|x_{N+1}, w, \beta)p(w|t, \alpha, \beta)dw,
\]

(3.18)

assuming that hyperparameters \( \alpha \) and \( \beta \) are known.

In the fully Bayesian framework, the uncertainty over the hyperparameters should also be taken into account by defining prior distributions \( p(\alpha) \) and \( p(\beta) \), which are called hyperpriors. Then the full posterior distribution is defined by (2.32) presented in Chap. 2. Finally, the fully Bayesian prediction is computed evaluating the predictive distribution defined by (2.34) in Chap. 2, using the posterior distribution \( p(w, \alpha, \beta|t) \).
Because the Bayesian prediction is based on integrations in the parameter and hyperparameter space over all parameters and hyperparameters, in general it is analytically intractable and the approximation techniques have to be used.

**Semi-Bayesian neural network**

The semi-Bayesian approach to neural network learning is based on an iterative procedure, in statistics called as type 2 maximum likelihood and in machine learning – evidence approximation [11, 64]. This procedure means maximization of the marginal likelihood in (3.17) for determining optimal hyperparameters $\alpha$ and $\beta$ instead of marginalizing them out.

In the case of a single hyperparameter $\alpha$, this procedure leads to the following iterative formula for finding the optimal value of the hyperparameter

$$\alpha_{\text{new}} = \frac{\gamma}{2E_W},$$  \hspace{1cm} (3.19)

where the term $E_W$ corresponds to the second term in (2.16) and $\gamma$ is the number of well-determined parameters computed from

$$\gamma = \sum_{i=1}^{W} \frac{\lambda_i}{\lambda_i + \alpha},$$  \hspace{1cm} (3.20)

where $\lambda_i$ is the $i$-th eigenvalue of the Hessian matrix containing the second derivatives of the sum-of-squares error function (2.16), evaluated at $w = w_{MAP}$, and $W$ is the total number of neural network parameters (weights).

In the case of hyperparameter $\beta$, the re-estimation procedure leads to the following iterative formula:

$$\beta_{\text{new}} = \frac{N - \gamma}{2E_D},$$  \hspace{1cm} (3.21)

where the term $E_D$ corresponds to the first term in (2.16) and $N$ is the total number of training patterns. The details of the evidence approximation approach can be found in Bishop [10].

**True-Bayesian neural network**

The true-Bayesian approach to neural network learning is based on stochastic sampling and Monte Carlo methods. In the Monte Carlo approach for Bayesian neural networks (BNN), we approximate integrals such as (2.34) by the finite sum in the form

$$p(t_{N+1}|x_{N+1}, t) \approx \frac{1}{m} \sum_{i=1}^{m} p(t_{N+1}|x_{N+1}, w_i),$$  \hspace{1cm} (3.22)
where \( \mathbf{w}_i \) are samples of weight vectors generated from the posterior distribution \( p(\mathbf{w}|t) \). The most effective sampling method for BNN is the hybrid Monte Carlo algorithm, which was introduced to BNN learning and prediction by Neal [79, 80].

This method combines the Metropolis-Hastings algorithm with the dynamical approach to stochastic sampling based on Hamiltonian dynamics [11]. This allows us to incorporate gradient information from the posterior distribution, which means that exploration of the sample space is more effective than an exclusively stochastic approach such as the Metropolis-Hastings algorithm [72].

Model selection methods

There are two main approaches to model comparison and selection in the context of selection of the optimal number of hidden neurons \( H_{opt} \) in a neural network model: the standard one based on a hold-out dataset and the Bayesian approach using only a training dataset.

The standard approach to model selection uses independent data, called a validation set, for comparison of neural models with a range of values representing the number of hidden units \( H^V \). We choose the number \( H_{ML}^V \), which minimizes validation error \( E_V(H,V) \). This process is illustrated in Fig. 3.2 by the learning and validation error curves for different numbers of hidden neurons.

The Bayesian approach to model selection is based on maximization of the marginal likelihood function in (3.17), also known as evidence function. For an FLNN model with \( H \) neurons in the hidden layer, the log of the marginal likelihood function is defined as

\[
\ln p(t|\alpha, \beta, H) = -E(\mathbf{w}_{MAP}) - \frac{1}{2} \ln |\mathbf{A}| + \frac{W}{2} \ln \alpha + \frac{N}{2} \ln \beta + \ln H! + 2 \ln H - \frac{N}{2} \ln(2\pi),
\]

where \( E(\mathbf{w}_{MAP}) \) is the regularized error function value evaluated for the optimal weights vector \( \mathbf{w}_{MAP} \), \( \mathbf{A} \) is the Hessian matrix, \( W \) is the total number of parameters (weights) and \( N \) is the number of training patterns. We choose the optimal model with \( H_{ML}^{ML} \) for which we have the maximum of log marginal likelihood function as illustrated in Fig. 3.2.

3.4 Prediction of plain concrete fatigue failure

3.4.1 Introduction

The problem of concrete fatigue failure has been considered by many researchers. For example, Antrim presented his results in [4]. The prediction of plain concrete
fatigue failure or fatigue life is an important factor in designing a concrete mix. Concrete fatigue is a progressive and localized structural damage that occurs when concrete is subjected to repeated loading and unloading. If the loads are above a certain threshold, microscopic cracks begin to form at stress concentrators, such as the surface, persistent slip bands (PSBs), and grain interfaces. Eventually, a crack reaches a critical size, and the structure suddenly fractures. In general, the nominal maximum stress values are less than the ultimate tensile stress limit and may be below the yield stress limit of the material.

The focus of this section is an example application of Gaussian processes to the problem of fatigue failure prediction of a plain concrete specimen under repeated loads. The example application is based on the experimental evidence from various papers collected by Furtak in [34]. Predictions of a Gaussian process are compared with the empirical formula proposed by Furtak, cf. [34]. The results presented in this chapter were previously published by the Author in three papers, see [112, 113, 117].

The American Society for Testing and Materials (ASTM International) defines fatigue life as the number of stress cycles of a specified character that a specimen sustains before the failure of a specified nature occurs. In the case of concrete, fatigue life or fatigue failure is defined as a number of loading cycles $N$ causing fatigue damage of the concrete specimen [60].

There are various approaches to fatigue life prediction. For example, one can use an analytical curve (formula) or apply a simple curve with a few parameters which are fitted to experimental data. The use of fracture mechanics is also possible [62]. Furtak derives an empirical formula in [34], as the following implicit relation between
variables:

$$\log N = \frac{1}{A} \left[ \log(1.16 \cdot C_f / \chi) + \log(1 + B \cdot R \cdot \log N) \right], \quad (3.24)$$

where: $\chi = f_{cN} / f_c$ and $R = \sigma_{\text{min}} / \sigma_{\text{max}}$ and the parameters according to paper [34], have the following values:

$$A = 0.008 - 0.118 \cdot \log(\sigma_I / f_c),$$

$$B = 0.118 \cdot (\sigma_{II} / \sigma_I - 1),$$

$$C_f = 1 + 0.07 \cdot (1 - R) \cdot \log f,$$

$\sigma_I$ and $\sigma_{II}$ are critical strengths.

In this work, the problem of predicting plain concrete fatigue failure $N$ is formulated as a regression problem. We assume that fatigue failure is a sum of an underlying deterministic function $y(x)$ and a random variable $\epsilon$. The input vector $x$ consists here of four variables, namely the plain concrete static uniaxial compressive strength ($f_c$), the ratio of minimal and maximal stress level in a compressive cycle of loading ($R = \sigma_{\text{min}} / \sigma_{\text{max}}$), the ratio of compressive fatigue strength and static strength of plain concrete, also called maximal compressive stress level ($\chi = f_{cN} / f_c$), and the frequency of the loading cycle ($f$). The target variable is the scalar output $y = \log N$.

A number of various feed-forward neural network models were used to the problem of predicting concrete fatigue failure in similar previous works on the matter:: back propagation (BPNN) [48], radial basis function (RBFNN) [49, 94], adaptive neuro-fuzzy inference system (ANFIS) [111] and fuzzy weights NN (FWNN) [89].

### 3.4.2 Data description and analysis

In his paper, Furtak collected and described a wide experimental dataset. This dataset corresponds to more than 400 tests performed on cubic or cylindrical specimens in 14 laboratories in 1934-80 (see Table 3.1) for details. The concrete specimens were subjected to cycles of compressive loadings at fixed frequencies. The numbers of cycles $N$ which caused the specimens fatigue damage were measured. In this work, we used experimental results from only 8 laboratories (group I), for which exclusively crisp (scalar) values were reported ($P = 216$ results), also mentioned in [44] and [48]. We didn’t use the experimental results from group II because the results were given in the form of an interval defined by minimal and maximal values of compressive strength $f_c$.

For example, Fig. 3.3 presents the results of two various fatigue tests performed on concrete samples. Table 3.2 shows the statistical parameters, namely minimal and maximal values, mean values and standard deviations for inputs and output variables.
Table 3.1: Experimental evidence collected in [34]

<table>
<thead>
<tr>
<th>No.</th>
<th>$R=\frac{s_{min}}{s_{max}}$</th>
<th>$f$ [Hz]</th>
<th>$f_c$ [MN/m$^2$]</th>
<th>Dimensions of specimens</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.025</td>
<td>16.7</td>
<td>28.0</td>
<td>$\phi 7.6 \times 15.2$</td>
</tr>
<tr>
<td>2</td>
<td>0.15, 0.38, 0.60, 0.88</td>
<td>41.0</td>
<td></td>
<td>$\phi 5.1 \times 10.2$</td>
</tr>
<tr>
<td>3</td>
<td>0.44</td>
<td>0.025</td>
<td>28.0</td>
<td>$10.2 \times 10.2 \times 30.5$</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>5.0</td>
<td>$[20.0, 30.0]$</td>
<td>$7.0 \times 7.0 \times 21.0$</td>
</tr>
<tr>
<td>5</td>
<td>0.14, 0.75</td>
<td>7.5</td>
<td>$[14.8, 32.7]$</td>
<td>$13.0 \times 13.0 \times 40.0$</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>20.0</td>
<td>$[20.0, 45.0]$</td>
<td>$10.2 \times 10.2 \times 50.8$</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>20.0</td>
<td>$[33.1, 44.8]$</td>
<td>$10.2 \times 10.2 \times 50.8$</td>
</tr>
<tr>
<td>8</td>
<td>0.044, 0.75</td>
<td>7.5</td>
<td>$[14.8, 32.7]$</td>
<td>$13.0 \times 13.0 \times 40.0$</td>
</tr>
<tr>
<td>9</td>
<td>0.05</td>
<td>16.7</td>
<td>25.5, 42.7</td>
<td>$\phi 7.6 \times 15.2$</td>
</tr>
<tr>
<td>10</td>
<td>0.05</td>
<td>1.167</td>
<td>24.8, 33.1</td>
<td>$15.2 \times 15.2 \times 162.6$</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>$[5.0, 16.7]$</td>
<td>$[20.0, 30.0]$</td>
<td>Different</td>
</tr>
<tr>
<td>12</td>
<td>0.074, 0.253</td>
<td>10.0</td>
<td>45.2</td>
<td>$\phi 5.0 \times 10.0$</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0.25</td>
<td>20.7</td>
<td>$10.2 \times 13.0 \times 82.7$</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>6.67, 15.0</td>
<td>26.2</td>
<td>$15.0 \times 15.0 \times 15.0$</td>
</tr>
</tbody>
</table>

### 3.4.3 Computer experiments

In this section, we describe the computer experiments which were done using experimental evidence presented in the previous section and two nonlinear regression models (a feed-forward neural network and a Gaussian process) together with learning algorithms introduced in Section 3.3.

A Bayesian neural network (BNN) with a single hidden layer of ten hyperbolic tangent units (neurons) and a linear output unit was used to model the relationship between the inputs and the output. The network weights were initially randomized from a Gaussian distribution. The neural network with automatic relevance determination (ARD) was trained using hybrid Monte Carlo (HMC) implemented in the Flexible Bayesian Modeling (FBM) software by Radford Neal [79, 80]. The length of the chains and the burn-in length were decided using visual inspection.

The BNN was compared with several models, starting from the standard MLP model. An early-stopped committee of MLP networks (ESC MLP) and a Gaussian process (GP) model were also applied. An early-stopped committee is an ad hoc method but it is fast and has proved to be quite a robust method when used as a committee of early-stopped MLPs [58]. The simplest form of the committee involves taking the output of the committee to be the average of the outputs of $L$ networks.
Table 3.2: Statistical parameters of input and output variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Std.Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_c ) [MPa]</td>
<td>20.70</td>
<td>45.20</td>
<td>34.68</td>
<td>8.84</td>
</tr>
<tr>
<td>( R ) [-]</td>
<td>0.00</td>
<td>0.88</td>
<td>0.14</td>
<td>0.18</td>
</tr>
<tr>
<td>( f ) [Hz]</td>
<td>0.025</td>
<td>150.0</td>
<td>21.30</td>
<td>39.38</td>
</tr>
<tr>
<td>( \chi ) [-]</td>
<td>0.49</td>
<td>0.94</td>
<td>0.74</td>
<td>0.11</td>
</tr>
<tr>
<td>( \log N ) [-]</td>
<td>1.86</td>
<td>7.34</td>
<td>4.56</td>
<td>1.41</td>
</tr>
</tbody>
</table>

Figure 3.3: Results of fatigue tests on concrete samples presented by: Antrim and McLaughlin (1959) (left) and Gray et al. (1961) (right), taken from [34]

The committee prediction is in the following form [10]:

\[
f_{\text{com}}(\mathbf{x}) = \frac{1}{L} \sum_{i=1}^{L} f_i(\mathbf{x}).
\]  

(3.25)

The MLP network and the ESC MLP model were implemented in MATLAB with the Netlab toolbox [72]. The ESC MLP model contained 10 networks which were created with a different division of the training data into learning (estimating) and stopped (validation) sets for each member. One third of the training examples (rounded down if necessary) were used for stopping (validation) and the rest for learning (estimating the weights). The network weights were initially randomized from a Gaussian distribution and they were trained by the scaled conjugate gradient (SCG) optimization method using 100 iterations (epochs). The Gaussian process model was also implemented and trained using the hybrid Monte Carlo method and the Flexible Bayesian Modeling (FBM) program.

In the analysis, both the inputs and output variables were first standardized to zero
mean and unit standard deviation by transformation

\[ \tilde{x}_i^n = \frac{x_i^n - \bar{x}_i}{s_i}, \]  

(3.26)

where \( \bar{x}_i \) is an average value and \( s_i \) is the standard deviation

\[ \bar{x}_i = \frac{1}{N} \sum_{n=1}^{N} x_i^n, \quad s_i = \sqrt{\frac{1}{N-1} \sum_{n=1}^{N} (x_i^n - \bar{x}_i)^2}. \]  

(3.27)

With data normalization, network weights can all be expected to have similar values if the inputs are equally important, and can be initialized randomly.

The experiments were performed using 216 examples (patterns). The generalization capability of the models in predicting concrete fatigue durability was estimated by the cross-validation method. To reduce the effects of data partitioning on the generalization performance, the models were evaluated for ten different (random) partitions of the data (195 or 194 for training and 21 or 22 for testing).

### 3.4.4 Discussion of the results

This section presents the results of the numerical experiments described in the previous section. The presented results have been computed and plotted for the first fold split of data into training and testing sets. In Table 3.3, in the case of Bayesian neural networks and the Gaussian process, the ARD column contains information about application of the hierarchical Gaussian prior distribution for the weights, called Automatic Relevance Determination, in these models. The next column contains information about the type of noise model, where ‘N’ indicates that the Gaussian noise model was assumed and \( t_\nu \) means that the Student’s \( t \)-distribution with unknown degrees of freedom \( \nu \) noise model was applied.

Table 3.3 shows the following estimated mean prediction errors and their standard deviations:

- root mean square error (RMSE)

\[ \text{RMSE} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (t_n - f(x_n))^2}, \]  

(3.28)

- average percentage error (APE)

\[ \text{APE} = \frac{1}{N} \sum_{n=1}^{N} \left| 1 - \frac{f(x_n)}{t_n} \right| \cdot 100\%, \]  

(3.29)
where $t_n$ is the target value, $f(x_n)$ is the output value of model $f$ for the input vector $x_n$ and $(\bar{t})$ and $(\bar{y})$ are the mean values. Both errors are values averaged over 10-fold cross-validation with standard deviation of the mean.

The values of the Pearson’s correlation coefficient are also presented:

$$
 r = \frac{\sum_{n=1}^{N} (t_n - \bar{t})(y_n - \bar{y})}{\sum_{n=1}^{N} (t_n - \bar{t}) \sum_{n=1}^{N} (y_n - \bar{y})}.
$$

(3.30)

From the results of the experiments we can see that the true-Bayesian neural networks and Gaussian process have better generalization performance than the early-stopped committee MLP ESC models and standard neural networks. The standard neural networks and the early-stopped committee MLP ESC models were not able to sufficiently control their complexity (the number of active, non-zero, parameters) during learning phase comparing with T-BNNs and GP models.

Table 3.3: Generalization performance comparison of various models with 5 and 10 hidden units in predicting concrete fatigue durability. The shown testing errors are the root-mean-squared error (RMSE) and the average percentage error (APE) averaged over 10-fold cross-validation with standard deviation of the mean

<table>
<thead>
<tr>
<th>Model</th>
<th>ARD</th>
<th>Noise model</th>
<th>RMSE±std</th>
<th>APE±std [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Furtak’s form.</td>
<td>-</td>
<td>-</td>
<td>0.885 ± 0.11</td>
<td>18.6 ± 3.0</td>
</tr>
<tr>
<td>SNN 4-5-1</td>
<td>-</td>
<td>-</td>
<td>0.717 ± 0.11</td>
<td>13.7 ± 2.8</td>
</tr>
<tr>
<td>SNN 4-10-1</td>
<td>-</td>
<td>-</td>
<td>0.727 ± 0.17</td>
<td>14.2 ± 3.8</td>
</tr>
<tr>
<td>ESC 4-5-1</td>
<td>-</td>
<td>-</td>
<td>0.703 ± 0.10</td>
<td>13.5 ± 2.6</td>
</tr>
<tr>
<td>ESC 4-10-1</td>
<td>-</td>
<td>-</td>
<td>0.731 ± 0.10</td>
<td>14.1 ± 2.6</td>
</tr>
<tr>
<td>T-BNN 4-5-1</td>
<td>yes</td>
<td>N</td>
<td>0.694 ± 0.09</td>
<td>13.3 ± 2.3</td>
</tr>
<tr>
<td>T-BNN 4-10-1</td>
<td>yes</td>
<td>N</td>
<td>0.693 ± 0.09</td>
<td>13.3 ± 2.4</td>
</tr>
<tr>
<td>T-BNN 4-5-1</td>
<td>no</td>
<td>$t_\nu$</td>
<td>0.692 ± 0.09</td>
<td>13.0 ± 2.5</td>
</tr>
<tr>
<td>T-BNN 4-10-1</td>
<td>no</td>
<td>$t_\nu$</td>
<td>0.689 ± 0.09</td>
<td>13.0 ± 2.6</td>
</tr>
<tr>
<td>GP</td>
<td>no</td>
<td>$t_\nu$</td>
<td>0.674 ± 0.09</td>
<td>13.0 ± 2.6</td>
</tr>
</tbody>
</table>

Fig. 3.4 shows the relations $f_{cN}/f_c - \log N$ for the empirical formula by Furtak [34] and the true-Bayesian neural networks for the tests performed by Gray et al. (1961) and for the tests carried out by Antrim and McLaughlin (1959) (for the first fold split of data). The dash-dot lines in Fig. 3.4 show the relations modeled by the true-Bayesian neural network and the dashed lines are the corresponding 1σ error bars which were computed on the basis of the RMS error for all 216 patterns (for the first fold split of data).

In Fig. 3.5, the relations $f_{cN}/f_c - \log N$ for the empirical formula by Furtak [34] and the Gaussian processes are shown for tests performed by Gray et al. (1961) and
3.4 Prediction of plain concrete fatigue failure

Figure 3.4: Results of prediction of fatigue tests by using true-Bayesian neural networks on concrete samples presented by: Antrim and McLaughlin (1959) (left) and Gray et al. (1961) (right), taken from [34] for tests carried out by Antrim and McLaughlin (1959) (for the first fold split of data). The dash-dot lines in Fig. 3.5 show the relations modeled by the true-Bayesian neural network and the dashed lines are the corresponding $1\sigma$ error bars which were computed on the basis of the RMS error for all 216 patterns (for the first fold split of data).

Figure 3.5: Results of prediction of fatigue tests by using Gaussian processes on concrete samples presented by: Antrim and McLaughlin (1959) (left) and Gray et al. (1961) (right), taken from [34]

Fig. 3.6 shows the measured concrete fatigue failure values vs. predicted their counterparts predicted by the Bayesian neural network and the Gaussian process (for the first fold split of data) with $1\sigma$ error bars (for 22 testing patterns only). It is also visible from the plots that the T-BNN and GP based predictions of the number
of fatigue cycles give lower values of log \( N \) and they locally better approximate the relation than the estimation done with the use of the empirical formula.

Figure 3.6: The measured concrete fatigue failure values contrasted with their counterparts predicted by the Bayesian neural network (left) and the Gaussian process (right) with 1\( \sigma \) error bars (for testing patterns only)

Fig. 3.7 shows variation in mean square error on the standardized testing set for the networks sampled in the last 100 iterations. It also presents the mean squared error on the testing set using the averaged predictions from all networks sampled up to the given iteration (within the last 100). There is clearly a benefit to be gained from averaging the predictions of a number of networks rather than making predictions from a single network trained with classical methods.

In the Bayesian framework it is also possible to assess the relative importance of inputs by observing the values of hyperparameters \( \alpha_i \) that control the prior for each of the four input variables. Table 3.4 gives mean and standard deviation for the hyperparameters associated with each input variable across each of the ten data partitions. A small value of \( \alpha_i \) corresponds to a large variance prior which allows weights of large magnitude. Such a variable is very important for predicting the output. On the other hand variable \( R \) has a larger value than the other three hyperparameters and can be identified as the least relevant variable in predicting concrete fatigue durability.
3.5 Prediction of HPC compressive strength

3.5.1 Introduction

Several definitions for high-performance concrete have been proposed so far. For example, Forster in [32] defined HPC as a concrete made with appropriate materials combined according to a selected mix design, properly mixed, transported, placed, consolidated and cured so that the resulting concrete will give excellent performance in the structure in which it is placed, in the environment to which it is exposed and with the loads to which it will be subject in the course of its design life.

In general, high-performance concrete is a mixture of several components. There are five basic components: cement \((C)\), water \((W)\), superplasticizer \((SP)\), fine aggregate \((FAG)\), coarse aggregate \((CAG)\) and some additives, like fly ash \((FA)\) or blast furnace slag \((FS)\).

The 28-day HPC compressive strength can be computed using empirical formulas, like the de Larrard’s empirical formula [23]

\[
f'_c = \frac{K_k R_c}{\left(\frac{1+3.1W/C}{1.4-0.4\exp(-11S/C)}\right)^{2}},
\]

(3.31)

where \(K_k\) is the coefficient which takes into account the effects of aggregate charac-

![Figure 3.7: Mean square error on testing set for networks sampled in last 100 iterations (one of the runs)](image)

Table 3.4: Mean and standard deviation for the ARD hyperparameters values

<table>
<thead>
<tr>
<th></th>
<th>(f_c)</th>
<th>(R)</th>
<th>(f)</th>
<th>(\chi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\bar{\alpha})</td>
<td>12.85</td>
<td>22.14</td>
<td>1.44</td>
<td>2.45</td>
</tr>
<tr>
<td>(s_\alpha)</td>
<td>24.78</td>
<td>36.54</td>
<td>1.73</td>
<td>2.04</td>
</tr>
</tbody>
</table>

3.5 Prediction of HPC compressive strength
teristics on concrete strength (the commonly assumed value is $K_k = 4.9$) and $R_c$ is the actual 28-day compressive strength of cement in MPa. The 28-day HPC compressive strength $f'_c$ plotted as a function of $W/C$ and $S/C$ based on Larrard’s formula, is presented in Fig. 3.8.

![Figure 3.8: The 28-day HPC compressive strength $f'_c$ as a function of $W/C$ and $S/C$, based on de Larrard’s formula computed for $K_k = 4.9$ and $R_c = 45$ MPa](image)

### 3.5.2 Data description and analysis

The models described in the third Section of this chapter were also tested using another regression benchmark dataset. In this benchmark, the goal is to predict compressive strength of high-performance concrete (HPC) on the basis of the HPC mix design. This benchmark has previously been analyzed in various papers, for example in [21, 22, 115].

This dataset was collected and described by Yeh [151] and contains the total number of $N = 1030$ examples of experimental results. Each example consists of an input vector $x_n$ and the corresponding target value $t_n$, where $n = 1, \ldots, N$. The input variables that are considered in this benchmark are:

1. the amount of cement ($C$), in kg/m$^3$
2. the amount of blast furnace slag ($FS$), in kg/m$^3$
3. the amount of fly ash ($FA$), in kg/m$^3$
3. Prediction of HPC compressive strength

4. the amount of water \((W)\), in kg/m\(^3\)

5. the amount of superplasticizer \((SP)\), in kg/m\(^3\)

6. the amount of coarse aggregate \((CAG)\), in kg/m\(^3\)

7. the amount of fine aggregate \((FAG)\), in kg/m\(^3\)

8. the age of concrete \((AGE)\), in days.

The output variable is the compressive strength of high-performance concrete \(f'_c\) in MPa after a given number of days. This dataset is available for download from the website of the University of California, Irvine Machine Learning Repository.

Before turning to the Bayesian regression methods, some intuition about the dataset is built by computing the summary statistics such as range, median, mean and standard deviation for each input and output variable. They are presented in Table 3.5. Additionally, a five-number summary is computed and presented in the box-and-whisker diagram in Fig. 3.9.

Table 3.5: Statistical parameters of input and output variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Range</th>
<th>Median</th>
<th>Mean</th>
<th>St.Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C) (kg/m(^3))</td>
<td>102-540</td>
<td>272</td>
<td>281</td>
<td>104</td>
</tr>
<tr>
<td>(FS) (kg/m(^3))</td>
<td>0-359</td>
<td>22</td>
<td>74</td>
<td>86</td>
</tr>
<tr>
<td>(FA) (kg/m(^3))</td>
<td>0-200</td>
<td>0</td>
<td>54</td>
<td>64</td>
</tr>
<tr>
<td>(W) (kg/m(^3))</td>
<td>121-247</td>
<td>185</td>
<td>182</td>
<td>21</td>
</tr>
<tr>
<td>(SP) (kg/m(^3))</td>
<td>0-32</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>(CAG) (kg/m(^3))</td>
<td>801-1145</td>
<td>968</td>
<td>973</td>
<td>78</td>
</tr>
<tr>
<td>(FAG) (kg/m(^3))</td>
<td>594-993</td>
<td>779</td>
<td>774</td>
<td>80</td>
</tr>
<tr>
<td>(AGE) (day)</td>
<td>1-365</td>
<td>28</td>
<td>46</td>
<td>63</td>
</tr>
<tr>
<td>(f'_c) (MPa)</td>
<td>2.3-82.6</td>
<td>34.4</td>
<td>35.8</td>
<td>16.7</td>
</tr>
</tbody>
</table>

Water/cement ratio 0.27-1.88 0.68 0.75 0.31
Water/binder ratio 0.24-0.90 0.47 0.47 0.13
Slag/binder ratio 0.00-0.61 0.04 0.17 0.20
Fly ash/binder ratio 0.00-0.55 0.00 0.14 0.17
(Slag+fly ash)/binder r. 0.00-0.74 0.32 0.31 0.21

The correlation coefficients between input variables are computed and presented in Table 3.6. Some variables are slightly linearly correlated: the amount of water \(W\) and the amount of superplasticizer \(SP\) have \(r = -0.66\), which is physically obvious. Moreover, in order to investigate nonlinear relationships between input variables (at power \(p\)) and the output variable, the correlation coefficients are computed in Table 3.7.
Fig. 3.10 illustrates the relationships by plotting the output variable against each input variable. Table 3.7 and Fig. 3.10 show that there is a close relationship between compressive strength $f'_{c}$ and the amount of cement $C$, with the linear coefficient of correlation $r = 0.50$. Fig. 3.11 shows the HPC dataset visualization using scatter plots of water to binder ratio ($W/(C + FS + FA)$) vs. compressive strength ($f'_{c}$) after various number of days ($AGE$) and for mixes without or with superplasticizer ($SP$), respectively.

Table 3.6: Correlation coefficients between input variables

<table>
<thead>
<tr>
<th></th>
<th>$C$</th>
<th>$F$</th>
<th>$FS$</th>
<th>$W$</th>
<th>$SP$</th>
<th>$FAG$</th>
<th>$CAG$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AGE$</td>
<td>0.08</td>
<td>-0.04</td>
<td>-0.15</td>
<td>0.28</td>
<td>-0.19</td>
<td>-0.00</td>
<td>-0.16</td>
</tr>
<tr>
<td>$C$</td>
<td>-</td>
<td>-0.28</td>
<td>-0.40</td>
<td>-0.08</td>
<td>0.09</td>
<td>-0.11</td>
<td>-0.22</td>
</tr>
<tr>
<td>$FA$</td>
<td>-</td>
<td>-</td>
<td>-0.32</td>
<td>0.11</td>
<td>0.04</td>
<td>-0.28</td>
<td>-0.28</td>
</tr>
<tr>
<td>$FS$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-0.26</td>
<td>0.38</td>
<td>-0.01</td>
<td>0.08</td>
</tr>
<tr>
<td>$W$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-0.66</td>
<td>-0.18</td>
<td>-0.45</td>
</tr>
<tr>
<td>$SP$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-0.27</td>
<td>0.22</td>
</tr>
<tr>
<td>$FAG$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-0.18</td>
</tr>
</tbody>
</table>
3.5.3 Computer experiments

Computer experiments related to training Gaussian process models and neural network models were carried out using $L=687$ training patterns. The generalization capabilities of trained models were finally assessed using $T=343$ testing patterns. The computations for standard neural networks (SNNs) and semi-Bayesian neural networks (S-BNNs) were done using the Netlab toolbox for MATLAB [72]. The computations for the true-Bayesian neural network (T-BNN) were done with the MCMCstuff toolbox for MATLAB [137].

Gaussian process

In this work, we have applied the Gaussian process model described in Sec. 3.3.1. It was defined using the squared exponential (SE) covariance function presented in 3.5. The hyperparameters of the covariance function were computed by using the
Figure 3.11: The HPC dataset visualization using scatter plots of water to binder ratio \( W/(C+FS+FA) \) vs. compressive strength \( f'_{c} \) after various numbers of days \( AGE \) and for mixes without or with superplasticizer \( SP \), respectively

maximum likelihood approach and the conjugate gradient algorithm.

### Neural networks

The feed-forward neural networks (FLNN) with a single hidden layer of neurons was used to model the relationship between the inputs and the output. We have applied a standard neural network (SNN) with gradient-based learning methods and two types of Bayesian neural networks described in Sec. 3.3.2. Both types of Bayesian neural networks were defined using an isotropic Gaussian prior distribution for weights with hyperparameters \( \alpha \) and a Gaussian noise model with hyperparameter \( \beta \). The initial network weights were sampled from a zero mean spherical Gaussian distribution with variance equal to one.

In the case of the semi-Bayesian neural network (S-BNN), training and prediction were done by using the evidence maximization approach, implemented in the Netlab toolbox for MATLAB. In the case of the true-Bayesian neural network (T-BNN),
3.5 Prediction of HPC compressive strength

Table 3.7: Correlation coefficients between each input variable at power \( p \) and output variable \( f'_c \)

<table>
<thead>
<tr>
<th>power ( p )</th>
<th>( C )</th>
<th>( FA )</th>
<th>( FS )</th>
<th>( W )</th>
<th>( SP )</th>
<th>( FAG )</th>
<th>( CAG )</th>
<th>( AGE )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.50</td>
<td>0.13</td>
<td>-0.11</td>
<td>-0.29</td>
<td>0.37</td>
<td>-0.16</td>
<td>-0.17</td>
<td>0.33</td>
</tr>
<tr>
<td>2</td>
<td>0.49</td>
<td>0.08</td>
<td>-0.13</td>
<td>-0.27</td>
<td>0.32</td>
<td>-0.16</td>
<td>-0.16</td>
<td>0.16</td>
</tr>
<tr>
<td>3</td>
<td>0.47</td>
<td>0.04</td>
<td>-0.14</td>
<td>-0.25</td>
<td>0.24</td>
<td>-0.15</td>
<td>-0.16</td>
<td>0.11</td>
</tr>
<tr>
<td>4</td>
<td>0.44</td>
<td>0.02</td>
<td>-0.14</td>
<td>-0.22</td>
<td>0.20</td>
<td>-0.15</td>
<td>-0.15</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Training and prediction were carried out by using the hybrid Monte Carlo (HMC) algorithm, implemented in the MCMCstuff toolbox for MATLAB.

Model selection for neural networks

An important problem related to layered neural networks is determining the optimal number of neurons in the hidden layers, which can sufficiently explain the observed data. This problem can be solved by applying model selection methods. Three approaches to this task have been tested in this work: the conventional approach based on a validation set, the semi-Bayesian approach based on maximization of marginal likelihood function and the full Bayesian approach based on Monte Carlo sampling.

The three approaches to model selection and learning of the FLNN model described in Sec. 3.3.2 were compared using this benchmark dataset. The optimal number of hidden neurons \( H_{opt} \) was chosen in the range \( H_{min} \leq H_{opt} \leq H_{max} \) with three approaches:

1. In the first approach the minimum of the validation error criterion \( H_{opt} = H_{opt}^V \) was used. The simultaneous selection of \( H_{opt} \) and regularization coefficient \( \lambda_{opt} \) using the penalized least squares (PLS) method (400 epochs of scaled conjugate gradient) was done with \( L = 344 \) standardized (zero mean and unit variance) training patterns and \( V = 343 \) standardized validating examples. The final training of the optimal model was done using \( L = 344 + 343 = 687 \) training patterns.

2. In the second approach we used the maximum marginal likelihood (MML) criterion \( H_{opt} = H_{opt}^{ML} \). The simultaneous selection of \( H_{opt}^{ML} \) and optimization of hyperparameters \( \alpha_{opt} \) and \( \beta_{opt} \) using the maximum a posteriori (MAP) approach for weights estimation and evidence procedure for setting the hyperparameters (8 re-estimation steps with 400 epochs of scaled conjugate gradient) was done using \( L = 344 + 343 \) standardized training patterns. The final training of the optimal model was not needed.
3. In the last approach the maximal number of hidden neurons \( H_{\text{opt}} = H_{\text{max}} \) was assumed in advance, because – as pointed out by Neal – over-complex models should be chosen where there is no prior knowledge, in order to ensure sufficient universal approximation capability of the model [80]. The hybrid Monte Carlo (HMC) sampling method was used for marginalization over model parameters and hyperparameters. Prediction was based on \( L = 687 \) not standardized training patterns.

3.5.4 Presentation of the results

In the case of neural networks, computer experiments defined in the previous subsection were carried out for a number of hidden neurons in the range from 4 to 16. The selection of that range allowed detailed comparison of models with a small number of hidden neurons (4-7), more complex models (8-12 hidden neurons) and, finally, probably the over-complex models (13-16 hidden neurons).

Comparison of models

Table 3.8 shows the comparison of predictive performance of models for learning and testing datasets. The first column gives the name of a given model. Columns 3 and 4 present the estimated root-mean-squared errors (RMSE) and average percentage errors (APE) for learning patterns. Column 5 gives the Pearson’s correlation coefficient \((r)\) for a learning set. In columns 6-8 are shown the corresponding errors and the correlation coefficient for testing patterns. The results were computed for the following models using learning algorithms: a standard neural network (SNN) using the PLS learning method and validation (VAL) for model selection; a semi-Bayesian neural network (S-BNN) using MAP learning, the evidence procedure for hyperparameters and MML for model selection; a true-Bayesian neural network (T-BNN) with the hybrid Monte Carlo (HMC) sampling method; a Gaussian process (GP) with the maximum likelihood-based learning of hyperparameters. For comparison, the results for a linear regression model (LR) with the maximum likelihood (ML) learning method are given in the first row of the table. It can be seen from the table that the best results are obtained for the true-Bayesian neural network (T-BNN) with 16 hidden neurons. However, these results are only slightly better in comparison with the semi-Bayesian neural network (S-BNN) with 13 hidden neurons and Gaussian process (GP).

Selection of the number of hidden neurons for neural networks

The results of the model selection using the validation set approach and the marginal likelihood (ML) approach are presented below. Fig. 3.12 shows validation
3.5 Prediction of HPC compressive strength

Table 3.8: Performance comparison of models for learning (L) and testing (T)

<table>
<thead>
<tr>
<th>Model</th>
<th>H</th>
<th>RMSE(L)</th>
<th>APE(L)</th>
<th>r(L)</th>
<th>RMSE(T)</th>
<th>APE(T)</th>
<th>r(T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td></td>
<td>10.3</td>
<td>31%</td>
<td>0.79</td>
<td>10.5</td>
<td>31%</td>
<td>0.78</td>
</tr>
<tr>
<td>SNN</td>
<td>10</td>
<td>4.4</td>
<td>12%</td>
<td>0.97</td>
<td>6.1</td>
<td>16%</td>
<td>0.93</td>
</tr>
<tr>
<td>GP</td>
<td></td>
<td>3.7</td>
<td>9%</td>
<td>0.98</td>
<td>5.4</td>
<td>13%</td>
<td>0.95</td>
</tr>
<tr>
<td>S-BNN</td>
<td>13</td>
<td>3.7</td>
<td>10%</td>
<td>0.98</td>
<td>5.2</td>
<td>12%</td>
<td>0.95</td>
</tr>
<tr>
<td>T-BNN</td>
<td>16</td>
<td>3.7</td>
<td>9%</td>
<td>0.98</td>
<td>5.4</td>
<td>11%</td>
<td>0.95</td>
</tr>
</tbody>
</table>

curves for selection of the number of hidden units in the standard neural network. During the learning phase, for each network model, the optimal regularization coefficient \( \lambda_{opt} \) was also determined. It is clearly discernible in the plots that in this approach the optimal number of hidden neurons is \( H^V_{opt} = 10 \). The corresponding optimal regularization coefficient is \( \lambda_{opt} = 0.1 \).

Fig. 3.12: Validation curves for selection of the number of hidden units in the standard neural network for different regularization coefficients

Fig. 3.13 presents the marginal likelihood function curve for selection of the number of hidden units in the semi-Bayesian neural network. In the maximum marginal likelihood function (MML) approach, the optimal neural network model has \( H^{MML}_{opt} = 13 \) hidden neurons. The optimal hyperparameters for that optimal model were also found by using the evidence maximization procedure. Fig. 3.14 shows the results of the iterative estimation of hyperparameters \( \alpha \) and \( \beta \). The optimal hyperparameters are \( \alpha_{opt} = 1.46 \) and \( \beta_{opt} = 16.9 \), giving the corresponding regular-
Figure 3.13: Marginal likelihood curve for selection of the number of hidden units in the semi-Bayesian neural network

Regularization coefficient $\lambda = \alpha / \beta = 0.09$, which is close to the value of the regularization coefficient from the previous approach ($\lambda_{\text{opt}} = 0.10$).

Fig. 3.15 presents the corresponding results of the iterative estimation of hyperparameters $\lambda = \alpha / \beta$ and $\gamma$. It can be seen from the plot that parameter $\gamma$, which is related to the number of network weights well determined by the learning patterns, is equal to 123 parameters, where the total number of network parameters in the network with $H^M_{\text{opt}} = 13$ is equal to 131 parameters. So it can be said that approximately 8 parameters are not absolutely necessary. In the case of a similar network model with 12 hidden units, the network model has only 121 parameters and this is why the optimal number of hidden neurons is 13 not 12.

Figure 3.14: Estimation of hyperparameters $\alpha$ (left) and $\beta$ (right)
3.5 Prediction of HPC compressive strength

Figure 3.15: Estimation of hyperparameters $\lambda = \alpha/\beta$ (left) and $\gamma$ (right)

Fig. 3.16 shows a scatter plot of training and testing datasets and the Gaussian predictive probability distribution (mean value and $2\sigma$ error bars) for compressive strength prediction with a semi-Bayesian neural network. Fig. 3.17 shows the Gaussian predictive probability distribution for compressive strength prediction with a semi-Bayesian neural network, corresponding to the vertical blue line in Fig. 3.16. Fig. 3.18 presents the comparison of predictive performance for the true-Bayesian neural network and the Gaussian process in the form of two plots showing measured versus predicted compressive strengths.
In this chapter we considered two problems of concrete properties prediction on the basis of experimental evidence. These problems were formulated as regression problems. The first problem pertained to prediction of fatigue failure of plain concrete. The second problem was related to prediction of compressive strength of high-performance concrete (HPC).

For solving these two problems, we applied two kinds of nonlinear regression models, namely feed-forward layered neural networks (FLNN) and Gaussian processes (GP) together with Bayesian inference. Bayesian inference was used for inferring posterior distributions of parameters, computing predictive distributions and for model selection in the case of neural networks. These two kinds of models have been compared in the aspects of prediction accuracy and computational efficiency. The comparison has also involved different approaches to model selection for neural networks: using a validation set, applying the maximum marginal likelihood criterion and using full Bayesian inference with the hybrid Monte Carlo method.

In the case of compressive strength prediction of HPC, computer experiments have shown that the strength can be predicted with satisfactory accuracy, using only eight input components defining the amounts of concrete mix ingredients and the age of the concrete sample. The results of computer experiments have shown that the best approach with respect to prediction accuracy and computational efficiency was the one applying Gaussian processes. However, it is important to note that the true-Bayesian neural network (T-BNN) with hybrid Monte Carlo based inference has given similar prediction accuracy but at the price of involving a lot more computations. A useful
alternative to these two models can be a semi-Bayesian neural network (S-BNN). This model, based on the evidence maximization approach, has given slightly worse predictions but was much faster than T-BNN.

The results have also confirmed the feasibility of using Bayesian neural networks and Gaussian processes to model the uncertainty in the predictions. The uncertainty in the predictions is caused mainly by variability in the datasets and a relatively small number of training patterns. Smaller uncertainties in predictions should be obtained through the use of larger datasets.

The experiments confirmed that the selection procedure for the number of hidden neurons based on the maximum marginal likelihood criterion allows an effective neural model design using only the training data. The evidence framework is also useful because it automatically selects regularization coefficients. Moreover, the full Bayesian inference applied to neural model learning and prediction is capable of giving the best predictive accuracy without the need of manual selection of the final model architecture and hyperparameters. The required computations are expensive, but in complex real-world problems with a large number of input and output variables there are few better alternatives, especially in the context of a steady increase in computational capabilities of computers (see for example [59] for an overview of recent methods for training neural networks).
Chapter 4

Structural damage identification

This chapter is devoted to solving two identification problems connected with damage identification in structures, such as buildings. In this work, we concentrate solely on vibration-based damage identification methods and only on the two initial steps of damage identification: damage detection and damage localization. Both steps can be formulated and solved using Bayesian machine learning methods for classification problems.

In this context, the damage detection problem is formulated and solved as a one-class classification problem whereas the damage localization problem is formulated and solved as a multi-class classification problem. We illustrate the application of classification methods in damage detection and localization with the use of a benchmark problem defined for a laboratory-scale steel frame.

4.1 Problem description

4.1.1 Damage identification

Vibration-based damage identification methods are supported by the premise that structural damage causes changes in the measured vibration signals. They use dynamic data from a monitored structure to detect and localize abnormal vibration patterns which may correspond to the condition of damage in the structure [25]. In this context, various algorithms and techniques developed using machine learning methods are used in the field of structural health monitoring (SHM), see for example [145, 146]. In particular, feed-forward layered neural networks (FLNNs) have proved to be useful for the purpose of vibration-based structural damage detection and localization. Nazarko, in his PhD thesis [78], presented an application of FLNNs for elastic waves-based damage identification in structural members, like aluminum strips. He found that feed-forward neural networks can be very useful in damage

There are two types of vibration-based methods. The most popular methods are called model-based. They use global dynamic analysis of vibration data and FE model updating for detecting changes in dynamic parameters of the monitored structure [28]. Other methods are non-model or model-free and they are based on time-series analysis of vibration data for damage detection and localization. Time-series methods form an important category within the broader vibration-based family of methods [3, 30]. They use random excitation and/or response signals (time series), statistical model building, and statistical decision making for inferring the health state of a structure [53]. For example, the acceleration signals from sensors are modeled with time series models and the coefficients are used as damage-sensitive features [30].

In general, a damage identification problem can be divided into four main steps [104]:

1. damage detection – in which abnormal behavior of the structure is detected
2. damage localization – in which the localization of the damage is done
3. damage quantification (level assessment) – in which the severity of the damage is assessed
4. damage prognosis – in which possible damage evolution scenarios are considered

4.1.2 Damage detection

In this work, we consider the problem of structural damage detection as a novelty (anomaly) detection task. Novelty detection (also known as anomaly detection, outlier detection or one-class classification) is a process of identification of novel (abnormal) patterns in a dataset [9, 96]. This process is based on computing the probability of observed patterns that do not belong to the density distribution characterizing a normal dataset. The crucial point in this approach is the assumption that
normal data are generated from an underlying data distribution, which – in turn –
may be estimated from the collected dataset [128].

A novelty detection framework consists of several steps. The first step is collec-
tion of a large number of patterns from normal (operational) conditions of the moni-
tored structure. The next step, is feature extraction. The aim of this step is to derive
the damage-sensitive features that describe normality. This is a very important step,
which may require some expert knowledge of the system under study [128]. After
the feature extraction step has been completed, we make use of dimensionality re-
duction techniques together with data visualization methods for better understanding
of the normal patterns scatter, especially in the vicinity of the boundaries. In the next
step, the probability density function $p(x)$ of normal feature vectors is assumed and
estimated from the normal dataset. A novelty threshold on the value of $p(x)$ is then
set using a validation dataset of abnormal patterns and $F_1$ score. In the final step, this
novelty threshold allows us to classify a new feature vector as either “abnormal” or
”normal”.

There are various approaches to describing (to modeling) the dataset of normal
patterns. The standard approach uses a density-based approach, where the form of the
density distribution is assumed in advance and the parameters of the distribution are
estimated using the maximum likelihood method and the normal dataset. For multi-
modal forms of data distribution, mixture models can be used such as the Gaussian
mixture model (GMM) or the expectation maximization (EM) algorithm for GMM
parameters estimation. GMM was previously applied for structural damage detection
as well. For example, Martin used a GMM-based method for anomaly detection in
the Space Shuttle Main Engine (SSME) [66]. Nair showed that GMM together with
the Mahalanobis distance can be useful in solving damage detection problems [73].
Słoński also presented a GMM-based approach for a benchmark steel frame [119].

### 4.1.3 Damage localization

The problem of structural damage localization is considered here as a multi-class
classification problem. In this approach, each possible damage scenario is considered
as a single class. The first step is collecting a large number of pseudo-experimental
data using simulation tools for each defined damage scenario. In the second step we
use the collected data to develop a classifier which will be able to correctly classify
most of the simulated damage scenarios. Then, in the final step, the trained classifier
is used to predict damage localization in a structure for a new input vector represent-
ing the condition of the structure. Słoński presented both a RVM-based approach for
a benchmark steel frame [116] and a GMM-based approach [118].
4.2 Benchmark steel frame

To illustrate the concept of novelty detection as the basis of a practical system for damage detection, we consider a specific application of novelty detection in structural health monitoring (SHM) concerned with determination whether there is damage in the 3D frame benchmark structure [45]. The presented algorithm is validated using data generated from the FE model of the ASCE benchmark steel frame. This benchmark is a standardized simulation tool for development and comparison of algorithms for SHM, and the analysis is limited to the linear stationary signals only.

4.2.1 Frame description

A scale-model of a modular 4-storey, 2-bay by 2-bay, steel-frame structure has been designed and built by the Earthquake Engineering Research Laboratory at the University of British Columbia (UBC), see the diagram of the structure in Fig. 4.1 (left). It is approximately 3.6 m tall with the total width of 2.5 m. Each floor is 0.9 m high and each bay is 1.25 m wide. The members are hot rolled grade 300W steel with the nominal yield stress 300 MPa. The columns are all oriented to be bending more strongly towards the x-direction (i.e., about the y-axis). The floor beams are oriented to be bending more strongly in the vertical direction, i.e. about the y-axis (x-axis) for those oriented with the longitudinal axis parallel to the x-axis (y-axis). The braces have no bending stiffness, so their orientation is irrelevant. There is one floor slab per bay per floor: four 800 kg slabs at the first level, four 600 kg slabs at each of the second and third levels and on the fourth floor, either four 400 kg slabs or three 400 kg and one 550 kg to create some asymmetry.

4.2.2 Simulation results

Two finite element models based on this structure were developed to generate the simulated response data. The first is a 12DOF shear-building model that constrains all motion except two horizontal translations and one rotation per floor. The second is a 120DOF model that only requires floor nodes to have the same horizontal translation and in-plane rotation. The columns and floor beams are modeled as Euler-Bernoulli beams in both finite element models. The braces are bars with no bending stiffness [46]. A diagram of the analytical model is shown in Fig. 4.1 (right). It should be noted that the x-direction (i.e., bending about the y-axis) is the strong direction due to the orientation of the columns. Furthermore, to be consistent with the axes used in later experimental tests, the compass directions associated with the axes are South for the positive y (weak) direction, and West for the positive x (strong) direction. In the present work, we have limited our study to the 120DOF
4.2 Benchmark steel frame

Figure 4.1: ASCE benchmark steel frame structure: photo of the structure in the laboratory (left); diagram showing accelerometers locations (right); source: [45]

model. The data generation scripts written in MATLAB are available on the WWW at http://mase.wustl.edu/asce.shm.

The simulated structure’s responses are measured using 16 virtual accelerometers. Locations of sensors and directions of measured accelerogram signals are shown in Fig. 4.1 (right). The frame can be excited in different ways but in this work we use only acceleration responses which were obtained by simulating an electrodynamic shaker placed at the center of one of the four bays on the top level of the structure. Fig. 4.2 presents an example of time series of simulated exciting force in y-direction (the top plot) together with the corresponding accelerograms time series in the undamaged structure in y-direction measured by two sensors placed on the first and the fourth floor of the structure (the bottom plot).

The main part of the benchmark problem consists of six cases of various structure damage as well as the additional case of undamaged structure (D0). Structural damage of the benchmark structure is simulated mainly by reducing the stiffness in the braces to zero.

The damage cases are defined as follows:

- D1: removal of all braces on the first floor,
- D2: removal of all braces on the first and third floors,
- D3: removal of one brace on the first floor,
Figure 4.2: Example of simulated time series for the benchmark structure. The top plot shows exciting force time series generated by electrodynamic shaker placed on the top level of the structure. The bottom plot shows accelerograms in y direction for two sensors placed on the first and fourth floor in the undamaged structure.

- D4: removal of one brace on the first and third floors,

- D5: case 4 + unscrewing the left end of element 18 (marked by x on the diagram of the frame in Fig. 4.1),

- D6: area of one brace on the first floor reduced to 2/3.

The first and the second damage cases are treated as severe damage scenarios and the rest of the damage cases are examples of minor damage. For more information about the benchmark frame and the related projects using this frame, see references [12, 138].
4.3 Damage-sensitive features extraction and visualization

4.3.1 Autoregressive model for features extraction

In this work, we use an autoregressive (AR) model of order $p$. The acceleration signal $x_{\text{acc},i}(t)$ from sensor $i$th is modeled by

$$x_{\text{acc},i}(t) = \sum_{k=1}^{p} \alpha_{ik}x_{\text{acc},i}(t - k) + \epsilon(t),$$

(4.1)

where $\alpha_{ik}$ is $k$th AR coefficient and $\epsilon(t)$ is the residual term.

The AR coefficients contain information about the dynamic characteristics of the structure (modal natural frequencies and damping ratios). Thus, changes to the structure stiffness matrix resulting from a permanent damage will change the AR coefficients. It appears that it is sufficient to use only the first AR coefficient to spot the changes in structural stiffness resulting from damage [74]. Using time series modeling of the structure acceleration response and the autoregressive (AR) coefficients $\alpha_{i1}$ as a features vector $\mathbf{x} = \{\alpha_{11}, \alpha_{21}, ..., \alpha_{I1}\}$, it is possible to build a one-class classifier capable of detecting damage in the structure. The coefficients computed for the undamaged structure form a statistical model of normality. After training, this model is subsequently applied for damage detection. Only the first coefficient of the AR model for each sensor is used in this work.

4.3.2 Dimensionality reduction and visualization

To visualize and reduce the dimensionality of the feature vectors, we apply a linear transformation, known as principal components analysis (PCA) or as the Karhunen-Loève transform [11]. For all seven damage cases, we subtract the dataset mean and project the dataset onto the principal component subspaces of dimensionality $M = 3$ and $M = 2$ obtained from PCA, respectively. Fig. 4.3 (the plot on the left) shows the eigenvalues arranged in the decreasing order. The cumulative sum of the eigenvalues, presented as a fraction of the entire sum, is shown on the right-hand plot in Fig. 4.3. They show that there is one dominating principal component with the corresponding largest eigenvalue $\lambda_1$ and that the last four eigenvalues are near zero.

Fig. 4.4 and Fig. 4.5 show the projected feature vectors for all the damage cases using the first three and the first two principal components. In the case of the first two principal components, it can be seen that the principal components projection separates well the most severe damage case D2 (removal of all braces on the first and third floors) and partially well the damage case D1 (removal of all braces on the first floor). However, the applied projection does not separate well either the remaining damage cases (D3-D6) or the undamaged case D0.
4.4 Damage detection with a Gaussian mixture model

In this work, we examine the application of a Gaussian mixture model (GMM) for solving the damage detection problem on the basis of pseudo-experimental acceleration time series obtained from the finite element model of the ASCE SHM Task Group benchmark structure [45]. GMM is a popular model for clustering data and continuous multimodal probability density estimation tasks. A set of GMMs (a committee of GMMs) can be used for multi-class classification problems.

In this section, the dataset for the undamaged frame (normal data) is modeled using a Gaussian mixture model with two components. This GMM-based approach has been selected because feature vectors form two clusters, see Fig. 4.4 and Fig. 4.5. The model is trained with the expectation maximization (EM) algorithm, which is an efficient iterative algorithm in the context of models with hidden variables, such as a GMM.

4.4.1 A Gaussian mixture model and the EM algorithm

Model equation

A Gaussian mixture model is defined as a linear superposition of $K$ Gaussian probability densities. It provides a richer class of density models than a single Gaussian [11]. The Gaussian mixture distribution can be written in the following form:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x | \mu_k, \Sigma_k),$$  \hspace{1cm} (4.2)
4.4 Damage detection with a Gaussian mixture model

Figure 4.4: Visualization of the feature vectors obtained by projecting data onto principal components. Seven cases of damage are shown: undamaged (D0) and damage (D1, D2, ..., D6). The plot shows the projected feature vectors in three-dimensional space

where \( p(x) \) denotes probability density over multidimensional vectors \( x \). Each Gaussian component of mixture \( \mathcal{N}(x|\mu_k, \Sigma_k) \) has its own mean vector \( \mu_k \) and covariance matrix \( \Sigma_k \). The parameters \( \pi_k \) are called mixing coefficients satisfying two requirements:

\[
0 \leq \pi_k \leq 1 \quad (4.3)
\]

as well as

\[
\sum_{k=1}^{K} \pi_k = 1. \quad (4.4)
\]

Therefore these parameters can be interpreted as probabilities. A Gaussian mixture model having 3 components is shown in Fig. 4.6.

A GMM can also be formulated using discrete latent variables. In this approach, we use a \( K \)-dimensional binary random variable \( z \) which has a 1-of-\( K \) representation and in which a particular element \( z_k \) is equal to 1 and all other elements are equal to 0. Therefore the values of \( z_k \) have to satisfy two conditions: \( z_k \in \{0, 1\} \) and \( \sum_{k} z_k = 1 \).
Figure 4.5: Visualization of feature vectors obtained by projecting data onto principal components. Seven cases of damaged are shown: undamaged (D0) and damaged (D1, D2, ..., D6). The plot shows the projected feature vectors in two-dimensional space.

**Expectation maximization algorithm for learning a GMM**

One way to set the values of the Gaussian mixture distribution is to use the maximum likelihood approach, maximizing the log of the likelihood function. It can be done with iterative optimization techniques, like a conjugate gradient method, or alternatively, using a powerful framework called expectation maximization (EM) [24]. The EM algorithm is a general technique for finding maximum likelihood solutions for probabilistic models with latent variables [11].

A quantity that plays an important role in the EM algorithm is the conditional probability of \( z \) given \( x \). We use \( \gamma(z_k) \) to denote \( (z_k = 1|x) \), whose value can be found using Bayes’ theorem

\[
\gamma(z_k) \equiv p(z_k = 1|x) = \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n|\mu_j, \Sigma_j)}.
\] (4.5)

We can view \( \pi_k \) as the prior probability of \( z_k = 1 \), and the quantity \( \gamma(z_k) \) as the corresponding posterior probability once we have observed \( x \). \( \gamma(z_k) \) can also be viewed as the responsibility that component \( k \) takes for ‘explaining’ the observation.
4.4 Damage detection with a Gaussian mixture model

Figure 4.6: Example of Gaussian mixture distribution \( p(x) \) in one dimension showing three Gaussians (each scaled by a coefficient) and their sum [11].

The starting point for the EM algorithm is the likelihood function. In the context of a GMM, the log of the likelihood function is given by

\[
\ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\}. \tag{4.6}
\]

The expectation-maximization algorithm for a GMM is defined by the following steps [11]:

1. Initialize parameters of the GMM (means \( \mu_k \), covariances \( \Sigma_k \) and mixing coefficients \( \pi_k \)).

2. E step. Evaluate the responsibilities using the most recent values of the GMM parameters

\[
\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}. \tag{4.7}
\]

3. M step. Re-estimate the parameters with the up-to-date values of responsibilities

\[
\mu_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n, \tag{4.8}
\]

\[
\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{\text{new}})(x_n - \mu_k^{\text{new}})^T, \tag{4.9}
\]

\[
\pi_k^{\text{new}} = \frac{N_k}{N}, \quad \text{where} \quad N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{4.10}
\]
4. Evaluate the log likelihood (4.6) and monitor the convergence of either the GMM parameters or the log likelihood. If the stopping criterion is not fulfilled, return to step 2.

For better convergence of the EM algorithm, the optimization process is often initialized with the parameter estimates of the GMM obtained from the \( K \)-means algorithm. Useful implementation of the EM algorithm for the GMM can be found in Netlab toolbox for MATLAB [72].

In Fig. 4.7 we illustrate the application of the EM algorithm to learning of the GMM applied to a dataset generated in two-dimensional space from a mixture of two Gaussians. The upper-left plot presents the dataset and the initial configuration of the GMM. The next two plots show the E-step and the M-step for the first cycle, respectively. Finally, the final configuration of the GMM after 9 cycles is presented in the lower-right plot.

### 4.4.2 Computer experiments

In computer experiments 580 patterns (16-dimensional feature vectors) have been generated for the undamaged structure and 490 – for each case of damage. Thus the total number of patterns corresponding to all six cases of damage is 2940. At first, all the patterns are transformed by standardization (zero mean and unit variance). Fig. 4.8 shows a plot of the datasets used for learning, validating and testing of a GMM in the damage detection step, presented in two-dimensional space defined by the first two features \( \alpha_1 \) and \( \alpha_2 \).

#### Learning the normality model

The first step in the damage detection process is using a dataset collected from the undamaged structure to learn a model that represents the structure behavior in normal (nominal) conditions. Fig. 4.9 shows the normal dataset in two-dimensional space defined by the first two features \( \alpha_1 \) and \( \alpha_2 \). This dataset is used for learning the GMM using the EM algorithm.

Fig. 4.10 shows the final configuration of the GMM after learning with the use of the EM algorithm. Fig. 4.10 presents the models in two-dimensional space defined by the first two features \( \alpha_1 \) and \( \alpha_2 \) using ellipses (left), and a 3D plot of the probability density defined by the GMM (right).

#### Setting the novelty threshold

The next step after defining and learning the model of a normal dataset, is to set the novelty threshold. In this work, the novelty threshold has been estimated by
4.4 Damage detection with a Gaussian mixture model

Figure 4.7: Example of the EM algorithm for learning a Gaussian mixture model in 2D. The upper-left plot shows the dataset and initial configuration of the GMM. The next two plots show E-step and M-step for the first cycle, respectively. Finally, the lower-right plot shows the final configuration of the GMM after 9 cycles, based on the example from [72]

applying 10-fold cross-validation and taking into account the patterns defined for all six damage scenarios (two severe and four minor cases).

Estimation of novelty threshold is based on the $F_1$ score

$$F_1 = \frac{2 \cdot prec \cdot rec}{prec + rec}.$$  \hspace{1cm} (4.11)

This measure is computed using precision ($prec$) and recall ($rec$) parameters. These parameters are defined as

$$prec = \frac{tp}{tp + fp},$$ \hspace{1cm} (4.12)

$$rec = \frac{tp}{tp + fn},$$ \hspace{1cm} (4.13)
Figure 4.8: Datasets for learning, validating and testing a GMM in the damage detection step, shown in two-dimensional space defined by the first two features $\alpha_1$ and $\alpha_2$.

Figure 4.9: The normal dataset shown in two-dimensional space defined by the first two features $\alpha_1$ and $\alpha_2$. This dataset is used for learning the GMM using the EM algorithm.

where

- $tp$ is the number of true positives,
- $fp$ is the number of false positives,
4.4 Damage detection with a Gaussian mixture model

Figure 4.10: Final configuration of the GMM after learning with the EM algorithm

- \( f_n \) is the number of false negatives.

Fig. 4.11 shows the validation dataset used for estimation of the novelty threshold.

![Validation dataset](image)

Figure 4.11: Validation dataset used for estimation of the novelty threshold

Fig. 4.12 presents the results of the novelty threshold estimation.

**Testing the Gaussian mixture model**

Finally, the presented algorithm is verified by using the testing patterns for all seven cases of damage. Fig. 4.8 presents the dataset applied for testing a GMM for damage detection in the benchmark frame.
4.4.3 Presentation and discussion of the damage detection results

The results of the structural damage detection with 16-dimensional feature vectors are presented in the first row of Table 4.1. In the case of severe damage scenarios, the presented algorithm gives almost perfect result, detecting correctly 98% of the patterns. But this result is obtained at the expense of a rather large number of misclassifications of minor damage cases and the undamaged structure.

For comparison, feature vectors from 2D and 3D spaces, computed by the PCA, have also been considered and the corresponding two GMMs were trained and applied to novelty detection. The results are shown in the last two rows of Table 4.1. It is interesting to note that the coefficient of success in the cases of severe damage using the three-dimensional feature vector is the same as with the full 16D feature vector.

Table 4.1: Results presented as coefficients of success in % in detection of scenarios of severe and minor damage in the structure. Results of testing the proposed algorithm for damage detection for various numbers of input variables (16, 3 and 2 (after applying PCA)). The last column presents the coefficients of success in damage detection for the undamaged structure.

<table>
<thead>
<tr>
<th>Number of inputs</th>
<th>severe damage [%]</th>
<th>minor damage [%]</th>
<th>undamaged [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 (all)</td>
<td>98</td>
<td>67</td>
<td>60</td>
</tr>
<tr>
<td>3 (PCA)</td>
<td>98</td>
<td>60</td>
<td>53</td>
</tr>
<tr>
<td>2 (PCA)</td>
<td>95</td>
<td>54</td>
<td>43</td>
</tr>
</tbody>
</table>
4.5 Damage localization using classification models

A classifier for damage localization has been built on the base of $N$ input-output data pairs $\{x_n, t^*_n\}_{n=1}^N$, where the input vector is defined as the feature vector $x$ and the target vector $t = \{t_1, t_2, ..., t_k, ..., t_K\}$ uses 1-of-$K$ coding scheme. So if $t_k = 1$, the actual damage pattern is $k$. Having built the classifier, we have used it to localize possible damage for a new feature vector $x^{N+1}$ describing the current structural stiffness. So the classifier predicts a corresponding target vector $t^{N+1}$, which consists of probabilities of the considered damage patterns to be the actual case of damage.

We apply and compare three various classifiers, namely a set of Gaussian mixture models (also known as a committee of Gaussian mixture models), a generalized linear model (GLM) and a relevance vector machine (RVM). The presented algorithm for damage localization is validated using damage-sensitive features obtained for time-series acceleration data generated from the numerical simulator of the ASCE benchmark steel frame. In computer experiments, we consider two damage localization tasks. The first task is defined for three cases of damage whereas the second task was defined for six cases of damage.

4.5.1 Classification models

Committee of Gaussian mixture models

A committee of Gaussian mixture models can be used for structural damage localization. In this case, each mixture is responsible for modeling one case of damage, defined a priori. This multi-mixture approach has been selected because feature vectors may form two or more clusters for each case of damage. It is illustrated in Fig. 4.13, where two-dimensional feature vectors form two clusters for three cases of damage.

As in the case of learning of a single GMM, learning of a set of GMMs is based on the EM algorithm.

Generalized linear model

A generalized linear model (GLM) for binary classification (two classes, $K = 2$) is defined by

$$p(C_1 | x) = y(x, w) = \sigma(w^T \phi(x)),$$

(4.14)

where $x$ is a vector of input variables (here: damage-sensitive), $w$ is a vector of model parameters, $\sigma(\cdot)$ is a logistic sigmoid function, $y(x, w)$ is a discrimination function and $p(C_1 | x)$ is the posterior probability of class $C_1$. 
Similarly, a generalized linear model (GLM) for $K$-class classification problems ($K > 2$, using 1-of-$K$ coding of classes) is defined by

$$p(C_k|x) = y_k(x, w_k) = \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)}, \quad a_k = w_k^T \phi(x),$$

(4.15)

where $y_k(x, w_k)$ is a discrimination function for $k$th class and $p(C_k|x)$ is the posterior probability of class $C_k$.

Parameters of the GLM are estimated using iterative reweighted least squares (IRLS) method which is based on the Newton-Raphson iterative optimization scheme [11].

**Relevance vector machine**

The relevance vector machine (RVM) is a Bayesian sparse kernel technique for regression and classification, proposed by Tipping [134] and described in detail by Bishop [11]. It shares many useful characteristics with the popular support vector machine (SVM) yet avoids the principal limitations of the SVM as it allows solving multiclass problems as well. It also gives a much sparser solution and, as a result, the RVM performs faster on test data. Finally, it provides posterior probabilistic outputs.

In the case of a binary classification problem, the RVM is defined by the same equation as the GLM:

$$p(C_1|x) = y(x, w) = \sigma(w^T \phi(x)).$$

(4.16)
Similarly, in the case of multi-class classification problems with $K$ classes, the equations for the RVM are the same as the equations for the GLM:

$$p(C_k|x) = y_k(x, w_k) = \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)}, \quad a_k = w_k^T \phi(x), \quad (4.17)$$

where $y_k(x, w_k)$ is a discrimination function for $k$th class and $p(C_k|x)$ is the posterior probability of class $C_k$.

Estimation of the RVM parameters is based on Bayesian inference. The starting point is the log likelihood function. In the case of binary classification it is defined by

$$p(t|w) = \prod_{n=1}^{N} y_{tn} \{1 - y_{tn}\}^{(1-t_n)}, \quad (4.18)$$

where $t$ denotes a vector of target values, $w$ denotes a vector of model parameters and $y_{tn}$ denotes a predicted class for a given input vector $x_n$ corresponding to the target class $t_n$.

The next step is to assume a prior probability distribution for model parameters $w$. For the RVM, the prior is a zero-mean isotropic Gaussian with a separate hyperparameter $\alpha_i$ for each of the parameters $w_i$ and has the form:

$$p(w|\alpha) = \prod_{i=1}^{M} N(w_i|0, \alpha_i^{-1}), \quad (4.19)$$

where $\alpha_i$ is a precision hyperparameter of the corresponding parameter $w_i$. It is an example of an automatic relevance determination (ARD)-based prior that is also used in the context of Bayesian neural networks.

Next, we compute the posterior distribution, which in this case is a Gaussian approximation in the form of

$$p(w|t, \alpha) = q(w|m, \Sigma). \quad (4.20)$$

The mean vector $m$ is computed from

$$m = A^{-1} \Phi^T (t - y), \quad (4.21)$$

where $\Phi$ is the design matrix with elements $\Phi_{ni} = \phi_i(x_n)$. The covariance matrix $\Sigma$ is computed from

$$\Sigma^{-1} = \Phi^T B \Phi + A, \quad (4.22)$$

where $B$ is a diagonal matrix with elements $b_n = y_{tn}(1 - y_{tn})$ and $A = \text{diag}(\alpha_i)$.

To compute the posterior distribution, we have to estimate the hyperparameters $\alpha_i$. In the context of the RVM, it is done by using the re-estimation procedure, which
is based on maximization of the log marginal likelihood function. The marginal likelihood function is defined in general as

$$p(t|\alpha) = \int p(t|w)p(w|\alpha)dw.$$  \hspace{1cm} (4.23)

Now taking into account the Gaussian approximation, the marginal likelihood function has the form:

$$p(t|\alpha) \simeq p(t|m)p(m|\alpha)(2\pi)^{M/2}|\Sigma|^{1/2}.$$ \hspace{1cm} (4.24)

Finally, setting the derivative of the marginal likelihood function with respect to $\alpha_i$ at zero, we obtain the re-estimation equations

$$\alpha_i^{\text{new}} = \frac{\gamma_i}{m_i^2},$$ \hspace{1cm} (4.25)

where $\gamma_i = 1 - \alpha_i\Sigma_{ii}$ [11].

4.5.2 Computer experiments for three cases of damage

In this section, we present computer experiments for damage localization in the benchmark steel frame described in Sec. 4.2. We take into account all the six cases of damage defined for the benchmark frame but reduced to only three cases by merging all four cases of minor damage into one case.

The computer experiments are based on learning and testing datasets generated by using simulation results presented in Sec. 4.2. 470 numerical results have been prepared for each case of damage, and the total number of patterns is $N=6\times470=2820$. We present the experiments and results both for the full sixteen-dimensional feature space and for the reduced, two-dimensional feature space resulting from the principal components analysis (PCA) transformation. The reduced feature vectors for three cases of damage are shown in Fig. 4.13.

We apply here three classification models introduced in the previous section. The first step in application of classification models to damage localization is the learning step. The number of learning patterns was $L=2580$. In the second step, the learned models were tested using a testing dataset. The number of testing patterns was $T=240$. In our computer experiments, we used the Netlab toolbox for MATLAB [72]

Learning classification models

First, we present the results of learning of a committee of three Gaussian mixture models. Fig. 4.14 shows the learning results for damage localization in the form of confusion matrices. In a confusion matrix, the rows represent the true classes (cases
4.5 Damage localization using classification models

of damage) and the columns – the predicted classes. The number above the matrix represents the percentage of correct classifications. The confusion matrix on the left was computed for the 16D feature space and the percentage of correct classifications was 99%. Fig. 4.14 (right) presents the learning results for damage localization in the case of 2D feature space. The percentage of correct classifications was 93%.

![Confusion matrices for learning of a committee of three GMMs in the case of 16D feature space (left) and in the case of 2D feature space (right)](image)

As regards the generalized linear model (GLM) and the relevance vector machine (RVM), Fig. 4.15 presents two confusion matrices for learning the results of these two models in 16D feature space. For GLM the percentage of correct classifications (success ratio) was 100% and for RVM it was 94% (see also Table 4.2). The best results of learning of the classification models in the case of 16D feature space were obtained for the generalized linear model and the committee of three Gaussian mixture models.

![Confusion matrices for learning of the GLM and RVM models in the case of 16D feature space; left: for GLM; right: for RVM](image)
Testing classification models

Here we present the results of testing the committee of three Gaussian mixture models. Fig. 4.16 presents the confusion matrices for testing. The confusion matrix on the left, was computed for the 16D feature space. The percentage of correct classifications was 98%. Fig. 4.16 (right) presents the testing results for damage localization in the case of 2D feature space. The percentage of correct classifications is 92%.

![Confusion matrices for testing the committee of three GMMs](image)

Figure 4.16: Confusion matrices for testing the committee of three GMMs; left: in the 16D feature space; right: in the 2D feature space

As for the of generalized linear model (GLM) and the relevance vector machine (RVM), Fig. 4.17 presents two confusion matrices for testing results of these two models in the 16D feature space. For both the GLM and for the RVM the percentage of correct classifications (success ratio) was 96% (see also Table 4.2). The best results of testing the classification models in the 16D feature space were obtained for the committee of three Gaussian mixture models. The results for the committees of three GMMs in both the 3D and 2D feature spaces, are presented in Table 4.2 (the last two rows).

Table 4.2: Learning (L) and testing (T) success ratios (in %) of damage localization for three cases of damage and for different numbers of input variables

<table>
<thead>
<tr>
<th>model</th>
<th># of inputs</th>
<th>learning [%]</th>
<th>testing [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMMs</td>
<td>16D</td>
<td>99</td>
<td>98</td>
</tr>
<tr>
<td>GLM</td>
<td>16D</td>
<td><strong>100</strong></td>
<td>96</td>
</tr>
<tr>
<td>RVM</td>
<td>16D</td>
<td>94</td>
<td>96</td>
</tr>
<tr>
<td>GMMs</td>
<td>3D</td>
<td>95</td>
<td>94</td>
</tr>
<tr>
<td>GMMs</td>
<td>2D</td>
<td>93</td>
<td>92</td>
</tr>
</tbody>
</table>
4.5 Damage localization using classification models

4.5.3 Computer experiments for the six cases of damage

In this section, we present computer experiments for damage localization in the benchmark steel frame, taking into account all the six cases of damage defined for the benchmark frame. 470 numerical results were prepared for each case of damage and the total number of patterns was $N=6 \times 470 = 2820$. We present the results both for the full sixteen-dimensional feature space and for the reduced, two-dimensional feature space resulting from the principal components analysis (PCA) transformation. The number of learning patterns was $L=2580$ and the number of testing patterns was $T=240$. We apply here solely the committee of Gaussian mixture models introduced above. In our computer experiments we use the Netlab toolbox for MATLAB [72].

Learning classification models

Here we present the learning results of a committee of Gaussian mixture models, consisting of six GMMs. The number of parameters of the final model in the case of the full sixteen-dimensional feature space was $6 \times 152 = 912$ whereas in the case of the reduced two-dimensional feature space it was only $6 \times 5 = 30$. Each component of the committee has been trained independently with the EM algorithm described in Sec. 4.4.1.

Fig. 4.18 shows confusion matrices for learning. The confusion matrix on the left, has been computed for the 16D feature space. The percentage of correct classifications is 67.6%. On the right are the learning results for damage localization in the case of the 2D feature space. The percentage of correct classifications was only 51.0%.

Figure 4.17: Confusion matrices for testing GLM and RVM models in 16D feature space; left: for GLM; right: for RVM
Structural damage identification

Figure 4.18: Confusion matrices for learning results of the committee of six GMMs; left: confusion matrix for the 16D feature space; right: confusion matrix for the 2D feature space

Testing classification models

Finally, we present the results of testing the committee of six Gaussian mixture models. Fig. 4.19 shows the testing results in the form of confusion matrices. The confusion matrix on the left was computed for the 16D feature space. The percentage of correct classifications was 65.8%. On the right are the testing results for damage localization in the case of 2D feature space. In this case, the percentage of correct classifications was only 50.0%. The results for these experiments are also presented in Table 4.3. The results for 3D and 2D feature vectors are presented in Table 4.3 in the last two rows. In this case, the results are worse because the models cannot classify accurately the cases of minor damage. The low rate of correct classifications may to some extent be the result of the insufficient number of sensors used for solving the structural damage localization problem in the analyzed structure.

Table 4.3: Learning (L) and testing (T) success ratios (in %) of damage localization for six cases of damage and for different numbers of input variables

<table>
<thead>
<tr>
<th>model</th>
<th># of inputs</th>
<th>learning [%]</th>
<th>testing [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMMs</td>
<td>16D</td>
<td>68</td>
<td>66</td>
</tr>
<tr>
<td>GMMs</td>
<td>3D</td>
<td>52</td>
<td>50</td>
</tr>
<tr>
<td>GMMs</td>
<td>2D</td>
<td>51</td>
<td>50</td>
</tr>
</tbody>
</table>
In this chapter we have demonstrated that classification models can be useful in solution of vibration-based structural damage identification problems. The two damage identification problems considered in this monograph have been solved for a benchmark laboratory-scale steel frame. In these problems, the acceleration time series data have been obtained by using 16 accelerometers. Each time series are represented by the first order auto-regressive (AR) model and coefficients of these 16 models defined the damage-sensitive feature vector.

In the first problem, we have used a Gaussian mixture model for solution of the structural damage detection problem. The optimal parameters of the GMM model have been estimated by employing the expectation-maximization (EM) algorithm and by using the feature vectors obtained from the undamaged structure. The application of the proposed algorithm for damage detection in the ASCE benchmark structure with the use of simulated data demonstrates that it is simple to implement and very effective in identification of severe damage scenarios, yet rather ineffective in detecting cases of minor damage.

In the second problem, we have applied a set of Gaussian mixture models for solving the structural damage localization problem. In this approach, each mixture is responsible for modeling one case of damage defined a priori. Moreover, in the second problem, we have applied and compared two classification models, namely
the generalized linear models and the relevance vector machine. As in the case of damage detection, the presented GMM-based algorithm for damage localization is very useful in locating severe damage but quite useless for locating spots of minor damage. However, after merging four cases of minor damage into one, the results got much better.
Chapter 5

Sequential identification of elastic properties

In this chapter we consider a sequential approach to identification problems. These problems often arise through sequential measurements of time series or are important features of other than time series sequential data sets, for example the sequence of characters in a sentence. The sequential data sets can be generated by stationary or non-stationary sequential distributions. Here we consider only the stationary case.

In this work we concentrate on two identification problems in the context of sequential estimation of elastic constants. The most general framework for dealing with such sequential identification problems is based on state space models. To solve these problems, we apply Bayesian approach using Kalman filters and particle filters also known as sequential Monte Carlo.

5.1 Introduction

So far in this book we have considered problems for which we have assumed sets of data points to be independent and identically distributed. This allowed us to use the likelihood function in the form of the product over all data points of the probability distribution evaluated at each data point [11].

However, we can observe numerous phenomena for which the data points indicate correlations between these data points that are close in the sequence. Such sequential data points can be easily put in order if they are related to a monotonically increasing parameter. For dynamic systems the physical time $t$ is frequently used, but in other cases any other pseudo-time parameters $\tau$ of monotonically increasing values can be used to define the sequential data $y(\tau)$. Moreover, we can apply the sequential
Sequential identification of elastic properties

approach to identification problems instead of the batch approach. This leads to on-
line Bayesian inference for model parameters.

The characteristic feature of state space models is the introduction of latent vari-
ables for states evolution. There are two most important examples of state space
models: hidden Markov models (HMMs) and linear dynamical system (LDSs)
[11, 40, 100]. In the hidden Markov model the latent variables are discrete. In the
linear dynamical system the latent variables are Gaussian. The well-known Kalman
filter (KF) is an example of a linear dynamical system. There are also other variants
of continuous state space models, for example nonlinear dynamical systems, in which
the latent variables are non-Gaussian.

In this context, Kalman filters represent a class of algorithms which are widely
used for exact inference in structural dynamical systems and for parametric iden-
tification. For example, Aoki et al. applied the KF for identification of Gurson’s
material constants [5]. The well-known extended Kalman filter (EKF) was applied
by Maier et al. [65] for stochastic estimation in fracture mechanics together with
optical measurements. Similarly, Bolzon et al. [14] presented application of the EKF
to parameter identification of a cohesive crack model.

Furukawa and Pan [35, 91] proposed an energy-based characterization technique
that recursively identifies the elastic constants of anisotropic materials using the
Kalman filter. They compared the proposed technique with a deterministic technique
based on solving a set of linear equations and showed that the filtering-based tech-
nique is not only more robust to measurement noise but also describes uncertainty
in identified constants. Sato and Sato [108] presented application of the FLNN and
the EKF for identification of dynamic characteristics of a structural system. Krok
and Waszczyszyn [55] applied a feed-forward neural network (FLNN) and a decou-
pled extended Kalman filter (DEKF) for prediction of acceleration response spectra
(ARS). Krok also showed application of FLNN and DEKF for analysis of cyclic be-
havior of concrete specimens [54].

Although the EKF has been widely applied, it is only reliable for almost linear
models. For highly-nonlinear models, the particle filter may be a viable alternative.
Recently, Tekieli and Słoński [121, 130] compared the Kalman filter and the particle
filter in the problem of Young’s modulus identification of a laboratory-scale frame.
They showed that the Monte Carlo filter can be a viable alternative to the Kalman
filter, especially in the case of nonlinear problems.

Moreover, various current approaches to on-line structural health monitoring
(SHM) rely heavily on the sequential identification of component or structure states
and/or parameters for damage detection, localization and prognosis [85]. Ching et
al. [20] compared the particle filter and the extended Kalman filter in the problem of
Bayesian state and parameter estimation of uncertain dynamical systems. Nasre:
ah and Manohar [75, 76] proposed a strategy for combining the finite element method
5.2 Sequential parametric identification

In this work we focus on discrete pseudo-time \( \tau \equiv k = 1, 2, \ldots, K \) which can be used to put in order the events in a set of sequential data

\[
\{y_1, y_2, \ldots, y_{k-1}, y_k, y_{k+1}, \ldots, y_K\}
\]

which are correlated to each other. This feature can be expressed by the conditional probability

\[
p(y_1, y_2, \ldots, y_K) = \prod_{k=1}^{K} p(y_k|y_1, \ldots, y_{k-1})
\]

\[
= p(y_1)p(y_2|y_1) \cdots p(y_K|y_1, \ldots, y_{K-1}).
\]

In many observations it is evident that the correlations between the current event and the previous ones are relaxing, which is the basic assumption of Markov models. The theory of Markov models addresses the correlations between the current and several previous \( m \) events. This can be related to Markov models of \( m \) order. Let us illustrate this definition on the examples of the first and second order Markov models, (see also Fig. 5.1):

\[
m = 1 : \quad p(y_1, y_2, \ldots, y_K) = p(y_1) \prod_{k=2}^{K} p(y_k|y_{k-1}),
\]

\[
m = 2 : \quad p(y_1, y_2, \ldots, y_K) = p(y_1)p(y_2|y_1) \prod_{k=3}^{K} p(y_k|y_{k-1}, y_{k-2}).
\]

The idea to preserve only a few time-delay terms is explored in the time series. The 1-st order Markov model is assumed to formulate the Kalman filter. The second assumption concerns the introduction of a state variable \( x_k \) and formulation of a model in the state space.

Sequential identification of elastic constants proposed in this work is based on Bayesian inference for a nonlinear stationary dynamical system defined in the discrete state space form.
5.2.1 State space models

The formulation of a state space model for a dynamical system consists of two nonlinear equations. The first equation is called a transition model and is defined by

\[
x_{k+1} = f(x_k, w_{k+1}), \quad k = 1, ..., K,
\]

where \( x_k \) denotes the set of state variables at time step \( k \). The function \( f(\cdot) \) is a transition function and it defines the evolution of state variables. The system evolution process is corrupted by random noise represented in equation (5.5) by a vector of random variables \( w_k \).

The second equation is called an observation model and is defined by

\[
y_{k+1} = h(x_{k+1}, v_{k+1}), \quad k = 1, ..., K,
\]

where \( y_k \) denotes the set of observable variables at time step \( k \). The output from the dynamical system \( y_k \) is measured at each time \( k \), and the measurements sequence up to time \( k \) is stored in matrix \( Y_{1:k} = \{ y_1, y_2, \ldots, y_k \} \). Function \( h(\cdot) \) is an observation (measurement) function and it defines the measurement process as a function of state variables. Similarly, the measurement process is also corrupted by random noise represented in equation (5.6) by a vector of random variables \( v_k \).

A state space model can be graphically presented in the form of a dynamic Bayesian network, see Fig. 5.2. A Bayesian network is a directed acyclic graph and represents the dependencies among random variables [52, 92]. In this context, the Kalman filter is an example of a dynamic Bayesian network with continuous variables and linear Gaussian conditional distributions. A dynamic Bayesian network can model any distribution in which the joint distribution over the sequence of \( K \) observed variables \( y_k \) and state (hidden) variables \( x_k \) is given by

\[
p(x_1, x_2, \ldots, x_K; y_1, y_2, \ldots, y_K) = p(x_1) \prod_{k=2}^{K} p(x_k | x_{k-1}) \prod_{k=1}^{K} p(y_k | x_k),
\]

where \( p(x_k | x_{k-1}) \) is the transition model (assumed here to be a first-order Markov chain) and \( p(y_k | x_k) \) is the observation (measurement) model.
In this section we describe Bayesian inference in the context of sequential data. There are four basic inference tasks: filtering, prediction, smoothing and the most likely explanation\cite{103, 107}. In addition to these four basic tasks, we also have a learning task in which the goal is to learn model parameters of state space models. The most common algorithms for learning state space models are based on the expectation maximization (EM) framework\cite{24, 101}, presented shortly in Chap. 4 in the context of learning Gaussian mixture models.

In this work we concentrate on the filtering task (also called state estimation) in which we compute the posterior distribution over states given all observations to date \(p(x_{k+1}, y_{1:k+1})\).

### 5.2.2 Bayesian state estimation

The main goal of Bayesian state estimation (filtering) is sequential inference of the posterior distribution \(p(x_{k+1}|Y_{1:k+1})\) starting from the prior distribution \(p(x_k|Y_{1:k})\). The inference is performed recursively in two steps: the prediction step and the update (correction) step. In the first step the prediction of state variables distribution \(p(x_{k+1}|y_k)\) before applying new measurements is done. This distribution is computed using the sum rule of probability and integrating out the state variables as

\[
 p(x_{k+1}|Y_{1:k}) = \int p(x_{k+1}|x_k)p(x_k|Y_{1:k})dx_k. \tag{5.8}
\]

Next new measurements \(y_{k+1}\) are used to update the prior to obtain the posterior distribution \(p(x_{k+1}|Y_{1:k+1})\) applying the Bayes’ rule

\[
 p(x_{k+1}|Y_{1:k+1}) = \frac{p(y_{k+1}|x_{k+1})p(x_{k+1}|Y_{1:k})}{p(y_{k+1}|Y_{1:k})}, \tag{5.9}
\]

where the denominator in (5.9) is computed from

\[
 p(y_{k+1}|Y_{1:k}) = \int p(y_{k+1}|x_{k+1})p(x_{k+1}|Y_{1:k})dx_{k+1}. \tag{5.10}
\]
The update step in Eq. (5.9) can also be written in the recursive form that is more useful for obtaining the particle filter algorithm. Using Bayes’ rule we can rewrite Eq. (5.9) as

\[
p(x_{k+1}|Y_{1:k+1}) = p(x_k|Y_{1:k}) \frac{p(y_{k+1}|x_{k+1})p(x_{k+1})}{p(y_{k+1}|Y_{1:k})}.
\]  

(5.11)

The Bayesian state estimation described above gives the posterior distribution over the states. It does not give however, a way to find the solution efficiently using both Eqs. (5.8) and (5.9). In addition, the exact inference is intractable and an approximate method has to be applied. In this work a particle filter (PF) algorithm, also known as sequential Monte Carlo, is used [38, 51]. This algorithm is based on Monte Carlo sampling and is described in short below. The details of the particle filter and an overview of its applications can be found for example in [26].

5.2.3 The Kalman filter

The model defined by Eq. (5.7) and presented in Fig. 5.2 was used in the formulation of the Kalman filter, which explores the recurrent formulas for computing vectors as mappings \( x_k \to x_{k+1} \) and \( x_{k+1} \to y_{k+1} \). The state and observation vectors \( x_k \) and \( y_k \) are assumed to be random variables. The approach described above is a base for formulating the linear Kalman filter, which is widely explored in the stochastic analysis of discrete linear dynamic systems, see e.g. [39].

The Kalman filter algorithm is based on the modified first order Markov chain model (5.7). This leads to the two following linear equations:

1) transition equation

\[
x_{k+1} = Ax_k + w_{k+1},
\]  

(5.12)

2) observation equation

\[
y_{k+1} = Bx_{k+1} + v_{k+1},
\]  

(5.13)

where \( A \) and \( B \) are matrices.

One-dimensional example of the Kalman filter

Here we show a simple one-dimensional example of the Kalman filter, taken from [103], and we present the corresponding equations for the first step only.

We begin with the prior distribution which is assumed to be a Gaussian with variance \( \sigma_0^2 \):

\[
p(x_0) = N(x_0| \mu_0, \sigma_0^2) = \frac{1}{\sqrt{2\pi \sigma_0^2}} \exp \left( \frac{(x_0 - \mu_0)^2}{2\sigma_0^2} \right).
\]  

(5.14)
The transition model adds a Gaussian perturbation of constant variance \( \sigma_x^2 \) to the current state:

\[
p(x_{t+1}|x_t) = N(x_{t+1}|x_t, \sigma_x^2) = (2\pi \sigma_x^2)^{-1/2} \exp\left[-\frac{(x_{t+1} - x_t)^2}{2\sigma_x^2}\right]. \tag{5.15}
\]

The sensor model assumes Gaussian noise with variance \( \sigma_y^2 \):

\[
p(y_t|x_t) = N(y_t|x_t, \sigma_y^2) = (2\pi \sigma_y^2)^{-1/2} \exp\left[-\frac{(y_t - h(x_t))^2}{2\sigma_y^2}\right]. \tag{5.16}
\]

Next, given the prior \( p(x_0) \), the one-step predicted distribution comes from Eq. (5.8) and by using a trick known as “completing the square” we obtain:

\[
p(x_1) = \int_{-\infty}^{\infty} p(x_1|x_0)p(x_0)dx_0 = (2\pi \sigma_1^2)^{-1/2} \exp\left[-\frac{(x_1 - \mu_0)^2}{2\sigma_1^2}\right], \tag{5.17}
\]

where \( \sigma_1^2 = \sigma_0^2 + \sigma_x^2 \). The one-step predicted distribution is a Gaussian distribution with the same mean \( \mu_0 \) and a variance equal to the sum of the original variance \( \sigma_0^2 \) and the transition variance \( \sigma_x^2 \). Note that the variance is larger than for the prior. It means that we have larger uncertainty in the predicted state.

Finally, we compute the update step using the observation at the first time step, namely \( y_1 \). From Eq. (5.9), this is given by

\[
p(x_1|y_1) = p(y_1|x_1)p(x_1) = (2\pi \sigma_1^2)^{-1/2} \exp\left[-\frac{(x_1 - \mu_1)^2}{2\sigma_1^2}\right], \tag{5.18}
\]

where the mean value \( \mu_1 \) is computed from

\[
\mu_1 = \frac{(\sigma_0^2 + \sigma_x^2)y_{t+1} + \sigma_y^2\mu_0}{\sigma_0^2 + \sigma_x^2 + \sigma_y^2}, \tag{5.19}
\]

and the variance \( \sigma_1^2 \) is now computed from

\[
\sigma_1^2 = \frac{(\sigma_0^2 + \sigma_x^2)\sigma_y^2}{\sigma_0^2 + \sigma_x^2 + \sigma_y^2}. \tag{5.20}
\]

Fig. 5.3 shows one update cycle for certain values of the transition and observation models. From the plot we can see that by using the observation, we obtain probability distribution over the predicted state with smaller uncertainty.
5.2.4 The particle filter

In order to implement Bayesian filtering using the particle filter, we approximate the posterior distribution \( p(x_{k+1} | y_{k+1}) \) using \( N \) particles \( x_{ik+1} \), \( (i = 1, 2, \ldots, N) \), with corresponding importance weights \( w_{ik+1} \), which replaces the posterior distribution with the empirical distribution

\[
P_N(x_{k+1}) = \sum_{i=1}^{N} w_{ik+1} \delta(x_{k+1} - x_{ik+1}),
\]

(5.21)

where \( \delta(\cdot) \) is the Dirac delta function. The weights are computed using sequential importance sampling as

\[
w_{ik+1} = w_k^i \frac{p(y_{k+1} | x_{ik+1}^i)p(x_{ik+1}^i | x_k^i)}{\pi(x_{ik+1}^i | x_k^i, y_{k+1})},
\]

(5.22)

where \( \pi(x_{ik+1} | x_k^i, y_{k+1}) \) is the importance distribution from which samples can be easily generated. In general, choosing the optimal importance distribution is rather difficult so, for simplicity, the common choice is to apply the transition density as the importance density

\[
\pi(x_{ik+1}^i | x_k^i, y_{k+1}) = p(x_{ik+1}^i | x_k^i),
\]

(5.23)
which yields a simple equation for computing weights in the next time step as

\[ w_{k+1}^i = w_k^i \frac{p(y_{k+1}|x_{k+1})}{p(y_{k+1}|x_{k+1})}. \] (5.24)

Note that these weights are normalized and satisfy \( 0 \leq w_k^i \leq 1 \) and \( \sum_{i=1}^{N} w_k^i = 1 \).

The initial weights are uniform with values \( w_k^i = 1/N \) but later on, during recursive computations, they become far from uniform, leading to particles degradation (few particles with large weights). As a result, the empirical distribution becomes very poor approximation of the state variables distribution \( p(x_{k+1}|Y_{1:k+1}) \). To overcome this particular degradation problem, a sequential resampling procedure is applied. The resampling procedure regenerates the set of particles by replicating the particles with high importance weights and removing samples with low weights.

Finally, the basic particle filter algorithm is as follows (see also Fig. 5.4 for a schematic illustration of the operation of the particle filter in 1D). It starts with a population of \( N \) initial-state samples, created by sampling from the prior \( p(x_0) \). Then the prediction-update-resample cycle is repeated for each time step [103]:

1. Each sample is weighted by the likelihood it assigns to the new evidence, \( p(y_{k+1}|x_{k+1}) \).
2. The population is resampled to generate a new population of \( N \) samples. Each new sample gets selected from the current population; the probability that a particular sample is selected is proportional to its weight. The new samples are unweighted.
3. Each sample is propagated forward by sampling the next state value \( x_{k+2} \), given the current value \( x_{k+1} \) for the sample, based on the transition model \( p(x_{k+2}|x_{k+1}) \).

### 5.2.5 Sequential parametric identification algorithm

At the beginning of this section, we formulated the sequential identification problem as a Bayesian state estimation problem. Because the elastic parameters are assumed not to change in time, they are treated here as time-independent state variables and the transition equation has the following simple form:

\[ x_{k+1} = x_k. \] (5.25)

The equation (5.25) is further modified as

\[ x_{k+1} = x_k + w_{k+1}, \] (5.26)
98 Sequential identification of elastic properties

Figure 5.4: Schematic illustration of the prediction-update-resample cycle of particle filter (detailed description in text)

where $w_k$ are random noise variables added for numerical efficiency of particle filter-based identification. In this work, $w$ is assumed to be a set of independent and identically distributed (iid) Gaussian random variables

$$p(w) = \mathcal{N}(w|0, \sigma^2_w),$$

(5.27)

where $\sigma^2_w$ is a covariance matrix.

The states are recursively estimated using measurements $y_k$. They are related to state variables $x_k$ by the nonlinear observation model $h(x_k, v_k)$ as

$$y_k = h(x_k) + v_k,$$

(5.28)

where $v_k$ are random noise variables introduced to account for modeling and measurement uncertainties. Here they are also assumed to be a set of independent and
identically distributed (iid) Gaussian random variables

\[ p(\mathbf{v}) \propto \mathcal{N}(\mathbf{v}|\mathbf{0}, \sigma_v^2), \quad (5.29) \]

where \( \sigma_v^2 \) is a covariance function.

Finally, the proposed approach is graphically presented in Fig. 5.5 in the form of a flowchart.

![Flowchart of the particle filter-based identification of elastic constants](image)

Figure 5.5: Flowchart of the particle filter-based identification of elastic constants
5.3 Estimation of Young’s modulus using Bayesian filters

5.3.1 Problem description

The main goal of this example is to present a novel combination of computational techniques applied for sequential estimation of the Young’s modulus of a laboratory-scale aluminum frame [132]. This proposed approach uses the digital image correlation (DIC) method for frame displacements measurements, the finite element method (FEM) for frame displacements predictions and Bayesian filtering for sequential parametric identification described in Sec. 5.2.

The examined structure is a laboratory two-storey frame, shown in Fig. 5.6 and described in detail in [67]. It is made from an aluminum alloy with the Young’s modulus $E = 70.77$ GPa and the Poisson’s ratio $\nu = 0.3$. The height of the frame is 0.40 m and the width is 0.48 m. The beams and columns have a rectangular cross-section 25x6 mm and the columns are bolted to a steel basement.

Figure 5.6: View of the two-storey laboratory-scale aluminum frame and the experimental stand used for frame displacements measurement

In this work the displacements of the frame are measured by tracking the markers attached to the structure by using computer vision methods [125] (see Fig. 5.6). In particular, the digital image correlation (DIC) method is applied [124]. The DIC method is a popular technique for full-field measurements of a structure displacements [123]. The vision system was developed by M.Tekieli with the aid of the OpenCV Library [15]. More details can be found in [131]. The system is able to track in near-real time the positions of markers with the accuracy of about 0.25mm.

Frame displacements under quasi-static loads are predicted by using the finite element method [153]. The frame is treated as a 2D model using standard 1D two-node frame elements (see Fig. 5.7 for the FEM model schematic illustration and the
positions where load has been applied).

5.3.2 Computer experiments

Having two sources of information about the frame displacements in near-real time, it is possible to solve the material model parameter identification problem using filtering. In our computer experiments, we apply the particle filter as the basic tool for the model updating and compare the results with the standard Kalman filter-based results. We assume that the state vector $x$ for both filters consists of only one state variable $x_k$, which represents the Young’s modulus in time step $k$. The observed variables $y_k$ are the displacements of six points on the frame (with markers). For both filters, the transition model for the evolution of the Young’s modulus is static, i.e. $x_k = x_{k-1}$. The prior distribution for the initial state is a normal density distribution $p(x_0) = N(x_0 | \mu, \sigma_x^2)$, with mean value $\mu = 70$ GPa and standard deviation $\sigma_x = 7$ GPa.

The observation model is represented by a linear Gaussian conditional distribution

$$p(y_k|x_k) = N(y_k|y_k^{FEM}(x_k), \Sigma_y),$$

(5.30)

where $y_k^{FEM}(x_k)$ denotes the displacements of the points on the frame computed by the FEM for the assumed Young’s modulus value, and $\Sigma_y = \sigma_y^2 I$ is the isotropic covariance matrix of measurement errors. All presented results are obtained for particle filters with $N = 400$ particles representing the Young’s modulus posterior distribution $P_N(x_k)$. 

Figure 5.7: Schematic illustration of the aluminum frame FEM model and the positions where load has been applied
5.3.3 Presentation and discussion of the estimation results

The results of two numerical experiments for the Young’s modulus sequential estimation with the application of vertical force are presented in Fig. 5.8 and Fig. 5.9. The plots show the evolution of mean values of the posterior distributions for Young’s modulus estimated using particle filters and the Kalman filter (for the first experiment only). The two horizontal lines shown in Fig. 5.8 and Fig. 5.9 represent the reference value $E=70.8\text{GPa}$, taken from [67], and the mean value computed for the values estimated for each measurement, respectively. In the first experiment, the mean value of Young’s modulus estimated by the particle filter was $E=67.0\text{GPa}$ and the corresponding result for the Kalman filter was also $E=67.0\text{GPa}$. These results are rather close to the reference value (the relative error less than 5%). In the second experiment the mean estimated value of Young’s modulus was $E=68.4\text{GPa}$. The plots also show the evolution of the measurement errors defined as the difference between the DIC-based and the FEM-based displacements of nodes. It is visible that the measurement errors are strongly correlated with the estimated values of Young’s modulus.

![Figure 5.8: Plot of sequential estimation of the Young’s modulus using the Kalman filter and the particle filter for the application of the vertical force (first experiment)](image)

Fig. 5.10 and Fig. 5.11 show the corresponding estimation results obtained for two experiments with the application of horizontal force. In the first experiment, the mean value of Young’s modulus estimated by the particle filter was $E=70.3\text{GPa}$ and in
5.4 Estimation of elastic parameters of plates with the particle filter

5.4.1 Problem description

Lamb waves have been used for three decades for non-destructive identification of elastic constants of plate structures. The reconstruction can be based on minimization...
of the sum of squares of the discrepancy between the experimental and analytical or numerical dispersion curves. Rogers [98] demonstrated identification of elastic properties on the example of several materials (aluminum, steel, glass) and with both thick (6mm) and thin plates (0.8mm) using nonlinear least squares. He also investigated the sensitivity of the nonlinear least squares solution to the measurement region of the dispersion curve and found that identification is more accurate when only certain selected frequencies are used.

Sale et al. [105] presented reconstruction of elastic moduli of plate-like structures based on fundamental symmetric and antisymmetric dispersion curves obtained through a semi-analytical finite element (SAFE) formulation and corresponding numerical or experimental curves. The SAFE method coupled with the inverse procedure was tested by identifying the elastic properties of a 2.54 mm thick aluminum plate. It was found that a smaller residual error could be obtained by applying both fundamental symmetric and antisymmetric dispersion curves.

Recently, Pabisek et al. proposed [90] a semi-analytical method for predictions of thin aluminum plate parameters, like elastic properties and thickness, based on guided-wave testing and a feed-forward neural network. They found that this method, after off-line training of a neural network model, is very fast and accurate in predictions of the plate parameters.
Most of the Lamb wave-based identification methods for elastic constants of plates have been hitherto categorized as deterministic approaches. They provide only numerical values for the elastic properties and often fail to fully characterize reconstruction uncertainty in a systematic manner. In this context Bayesian methods offer a more systematic approach to uncertainty quantification. For example, Gogu et al. [37] adopted the Bayesian framework for identifying elastic constants of an orthotropic composite plate from an open-hole tensile test with full-field displacement measurements. They found that the Bayesian approach is useful since it allows more accurate representation of experimental uncertainty as well as a solid basis for combining measurements and their uncertainties stemming from different sources.

In this work, we propose a novel application of the particle filter for sequential stochastic identification of elastic constants of plate structures using Lamb waves [120]. The proposed procedure is based on the comparison of experimental and analytical or numerical dispersion curves, and the identification results are in the form of a posterior distribution over elastic constants which describes the uncertainty. The proposed procedure is verified on an example of pseudo-experimental dispersion curves computed for an aluminum plate.
5.4.2 Lamb waves for elastic plates

Rayleigh-Lamb equations

Nondestructive identification of elastic constants of thin plates may be based on dispersion curves derived for guided ultrasonic waves propagating in elastic plates. In the case of a homogeneous isotropic and elastic infinite plate, the propagation of ultrasonic waves, assuming plane strain, is described by the Lamb waves theory [57, 99]. The waves propagating in these plates are dispersive and have an infinite number of symmetric and anti-symmetric modes that are characterized by dispersion curves.

The dispersion curves are usually computed by numerical solution of the Rayleigh-Lamb characteristic equations. In the case of symmetric waves, the dispersion curves are computed from the following equation:

$$\frac{\tan(\beta h)}{\tan(\alpha h)} = -\frac{4\alpha \beta k^2}{(k^2 - \beta^2)^2}, \quad (5.31)$$

where $\alpha^2 = \frac{\omega^2}{c_T^2} - k^2$ and $\beta^2 = \frac{\omega^2}{c_T^2} - k^2$. $\omega$ is the frequency, $k$ is the wavenumber, plate thickness is $d = 2h$, $c_L, c_T$ are wave velocities and $\nu$ is the Poisson’s ratio. The velocities of transverse waves $c_L$ and longitudinal waves $c_T$, respectively are given by

$$c_L = \sqrt{\frac{\lambda + 2\mu}{\rho}} = \sqrt{\frac{E}{\rho (1 + \nu)(1 - 2\nu)}}, \quad c_T = \sqrt{\frac{\mu}{\rho}} = \sqrt{\frac{E}{\rho 2(1 + \nu)}}, \quad (5.32)$$

where $\mu$ is the first Lamé constant, $\lambda$ is the second Lamé constant and $\rho$ is the mass density. From Eq. (5.32) we can see that the ratio of velocities $\kappa = \frac{c_L}{c_T}$ depends only on the Poisson’s ratio as

$$\kappa = \frac{2(1 - \nu)}{1 - 2\nu} > \frac{4}{3}. \quad (5.33)$$

In the case of antisymmetric waves, the dispersion curves are computed from a similar equation:

$$\frac{\tan(\beta h)}{\tan(\alpha h)} = -\frac{(k^2 - \beta^2)^2}{4\alpha \beta k^2}, \quad (5.34)$$

A semi-analytical approach for efficient solution of the Rayleigh-Lamb equations is based on the introduction of non-dimensional frequency and wavenumber variables in the form [2]

$$\Omega = \frac{\omega h}{c_T}, \quad \xi = kh, \quad (5.35)$$

and the following two equations

$$x = (\Omega^2 - \xi^2)^{1/2}, \quad y = (\Omega^2 \kappa^2 - \xi^2)^{1/2}. \quad (5.36)$$
Substituting the non-dimensional variables into the frequency equations (5.31) and (5.34) we obtain

\[
\frac{4xy\xi^2}{(\xi^2 - x^2)^2} + \frac{\tan(x)}{\tan(y)} = 0, \quad \text{for symmetric waves,} \tag{5.37}
\]

\[
\frac{4xy\xi^2}{(\xi^2 - x^2)^2} + \frac{\tan(y)}{\tan(x)} = 0, \quad \text{for antisymmetric waves.} \tag{5.38}
\]

After some additional transformations, the Rayleigh-Lamb equations for plates in non-dimensional variables have the form:

\[
(\xi^2 - x^2)^2 \sin(x) \cos(y) + 4xy\xi^2 \cos(x) \sin(y) = 0, \quad \text{for symmetric modes,} \tag{5.39}
\]

\[
(\xi^2 - x^2)^2 \sin(y) \cos(x) + 4xy\xi^2 \cos(y) \sin(x) = 0, \quad \text{for antisymmetric modes.} \tag{5.40}
\]

These equations are solved numerically for the roots which form the basis for obtaining non-dimensional dispersion curves. Fig. 5.12 shows two fundamental non-dimensional dispersion curves \( S_0 \) and \( A_0 \), which were computed using equations (5.39) and (5.40) assuming Poisson’s ratio \( \nu = 0.33 \).

![Plot of wave number vs. \( \Omega \)](image)

**Figure 5.12:** Plot of fundamental non-dimensional dispersion curves \( S_0 \) and \( A_0 \) computed using equations (5.39) and (5.40) for \( \nu = 0.33 \)
Dispersion curves

From the non-dimensional relation between $\Omega$ and $\xi$, using equations in (5.35), we can compute the fundamental dispersion curves for certain specified values of Young’s modulus $E$, half of the plate thickness $d$ and the mass density $\rho$. Each dispersion curve consists of points defined as a pair of frequency value and the corresponding wavenumber. For example Fig. 5.13 presents three dispersion curves obtained assuming the plate thickness $d = 2\text{mm}$, the mass density $\rho = 2700\text{kg/m}^3$ and three values of Young’s modulus $E$ within the range from 60GPa to 70 GPa. It can be seen from this plot that the value of Young’s modulus has rather small influence on the shape of the dispersion curves.

Figure 5.13: Plot of fundamental dispersion curves $A_0$ computed for Poisson’s ratio $\nu = 0.33$, half of plate thickness $d = 2\text{ mm}$, mass density $\rho = 2700\text{kg/m}^3$ and three Young’s moduli $E$

5.4.3 Computer experiments for Young’s modulus identification

To assess the effectiveness of the proposed method, numerical exercises for an aluminum plate are performed. In the numerical experiments, the material properties of the aluminum plate (Young’s modulus, Poisson’s ratio and mass density) and the plate thickness are assumed in advance (see Table 5.1 for the exact values of these parameters).
Table 5.1: Assumed values of the aluminum plate parameters applied in the numerical experiments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>E(GPa)</th>
<th>ν(-)</th>
<th>ρ(kg/m³)</th>
<th>h(mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assumed value</td>
<td>67.5</td>
<td>0.33</td>
<td>2700</td>
<td>4</td>
</tr>
</tbody>
</table>

Having defined the plate parameters, we compute a pseudo-experimental fundamental antisymmetric dispersion curve $A_0$ using the semi-analytical approach described above. During sequential identification, this curve is represented with a small number of parameters that are used as the observed variables $y$.

**One-dimensional transition model**

In the first experiment, we assume that the Young’s modulus is a time-independent state variable and the transition equation for a state-space model reads as follows:

$$x_{k+1} = x_k + w_{k+1}, \quad (5.41)$$

where the state variable $x_k$ is defined as the Young’s modulus in $k$-th step $x_k = E_k$ and $w_k$ denotes an independent and identically distributed Gaussian random variable with zero mean and variance $\sigma_x^2$

$$p(w) = \mathcal{N}(0, \sigma_x^2). \quad (5.42)$$

The value of $\sigma_x$ was set at 0.003 GPa.

**Observation equation**

The vector of observed variables $y_k$ is defined as a set of parameters of the fundamental antisymmetric dispersion curves $A_0$. The observed variables are related to the state variable $x_k$ by the nonlinear observation function $h(x_k)$ in the form:

$$y_k = h(x_k) + v_k, \quad (5.43)$$

where $v_k$ denotes an independent and identically distributed multivariate Gaussian random variable with zero mean vector and covariance matrix $\sigma_y^2$. The covariance matrix is defined as a diagonal isotropic matrix with elements $\sigma_y$ equal to 0.01 l/m.

The nonlinear observation function $h(x_k)$ is used to predict the parameters of the fundamental dispersion curve for a given Young’s modulus.
Prior distribution

Our initial and uncertain knowledge about the state variable (here: Young’s modulus) is represented by a prior distribution \( p(x_0) \). In the experiments we apply a one-dimensional normal prior distribution \( p(x_0) = N(\mu_0, \sigma_0^2) \), with mean value \( \mu_0 = 67.0 \) GPa and standard deviation \( \sigma_0 = 3.35 \) GPa (coefficient of variation (CoV) was 5\%). Fig. 5.14 shows the plot of the prior (as a dashed line).

![Prior and posterior distributions for Young’s modulus](image)

Figure 5.14: Plot of the prior and posterior distributions for Young’s modulus. The Gaussian prior distribution (dashed line) has mean value \( \mu_{\text{prior}} = 67.0 \) GPa and standard deviation \( \sigma_{\text{prior}} = 3.35 \) GPa (coefficient of variation (CoV) is 5\%). The approximate Gaussian posterior distribution (solid line) has mean value \( \mu_{\text{pos}} = 67.5 \) GPa and standard deviation \( \sigma_{\text{pos}} = 0.05 \) GPa (coefficient of variation (CoV) is 0.1\%)

Young’s modulus identification results

The approximate posterior distribution of Young’s modulus given pseudo-experimental dispersion curves \( P_N(x_k|y_k) \) in the \( k \)-th step is computed using the particle filter-based identification procedure described above. In the experiments we apply \( N = 2000 \) particles to obtain the approximate posterior distribution and the number of steps in the sequential identification was set at \( K = 500 \). Fig. 5.15 shows
the sequential nature of the elastic constant identification process by plotting the evolution of the mean value of the posterior distribution and the corresponding plot for the one-standard deviation error bars as a function of the step number. A solid horizontal line – also shown in Fig. 5.15 – represents the reference Young’s modulus value (67.5 GPa) applied in the numerical experiments. It may be observed from the plot that the estimation process approached the reference value quite rapidly (in about 120 iterations).

![Figure 5.15: Plot of evolution of the mean value of posterior distribution for Young’s modulus and the corresponding one-standard deviation error bars (the solid horizontal line represents Young’s modulus value (67.5 GPa) assumed in the numerical experiments)](image)

Table 5.2 presents the statistical parameters of the prior and posterior distributions in the form of mean values, whereas the standard deviations and the coefficients of variation (CoV) are given. It can be seen from the table that the final mean value of the posterior distribution is the same as the reference value. Moreover, the coefficient of variation decreased from 5% for the prior distribution to only 0.1% for the final posterior distribution. Fig. 5.14 shows the final one-dimensional posterior distribution together with the prior distribution.
Table 5.2: Statistical parameters of the prior and posterior distributions for Young’s modulus (mean value, standard deviation and coefficient of variation (CoV))

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Prior</th>
<th>Posterior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean value (GPa)</td>
<td>67.0</td>
<td>67.5</td>
</tr>
<tr>
<td>Standard deviation (GPa)</td>
<td>3.35</td>
<td>0.05</td>
</tr>
<tr>
<td>CoV (%)</td>
<td>5.0</td>
<td>0.1</td>
</tr>
</tbody>
</table>

5.4.4 Computer experiments for Young’s modulus and Poisson’s ratio identification

Two-dimensional transition model

In this example we assume that Young’s modulus and Poisson’s ratio were time-independent state variables and the transition equation for a state-space model is

\[\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{w}_{k+1},\] (5.44)

where the vector of the state variables \(\mathbf{x}_k\) consists of the Young’s modulus and Poisson’s ratio in \(k\)-th step \(\mathbf{x}_k = \{E_k, \nu_k\}\). The vector \(\mathbf{w}_k\) denotes independent and identically distributed Gaussian random variables with zero mean vector and covariance matrix \(\sigma_w^2\)

\[p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | 0, \sigma_w^2),\] (5.45)

The covariance matrix is defined as a diagonal isotropic matrix with elements \(\sigma_x\) equal to 0.003 GPa and 0.0015, respectively.

Observation equation

In the experiments we use a vector of observed variables \(\mathbf{y}_k\), which are defined as parameters of the fundamental antisymmetric dispersion curves \(A_0\). The observed variables are related to the state variable \(\mathbf{x}_k\) by the nonlinear observation function \(h(\mathbf{x}_k)\) in the form:

\[\mathbf{y}_k = h(\mathbf{x}_k) + \mathbf{v}_k,\] (5.46)

where \(\mathbf{v}_k\) denotes an independent and identically distributed multivariate Gaussian random variable with zero mean vector and covariance matrix \(\sigma_v^2\). The covariance matrix is defined as a diagonal isotropic matrix with elements \(\sigma_y\) equal to 0.01 l/m.

The nonlinear observation function \(h(\mathbf{x}_k)\) is used to predict the parameters of the fundamental dispersion curve for a given Young’s modulus and Poisson’s ratio.
Prior distribution

The uncertainty about Young’s modulus and Poisson’s ratio values is represented by a two-dimensional Gaussian prior distribution $p(x_0) = \mathcal{N}(\mu_0, \sigma_0^2)$, with mean values $\mu_0=[67.1 \text{ GPa}, 0.33]$. The standard deviations are about 5% of the corresponding mean values, i.e. $\sigma_0=[3.4 \text{ GPa}, 0.02]$. Fig. 5.16 shows a contour plot of the two-dimensional prior together with the scatter plot of particles sampled from that prior. Table 5.3 presents the statistical parameters of the prior.

Table 5.3: Statistical parameters of the Gaussian prior distribution

<table>
<thead>
<tr>
<th>Parameter</th>
<th>E(GPa)</th>
<th>$\nu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean value</td>
<td>67.1</td>
<td>0.330</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>3.40</td>
<td>0.017</td>
</tr>
<tr>
<td>CoV (%)</td>
<td>5.1</td>
<td>5.1</td>
</tr>
</tbody>
</table>

Young’s modulus and Poisson’s ratio identification results

As in the first example, the approximate posterior distribution of elastic constants given pseudo-experimental dispersion curves $P_N(x_k|y_k)$ in the $k$-th step is computed using the particle filter-based identification procedure described above. In the second experiment, we apply $N = 2000$ particles to approximate the posterior distribution and the number of steps was $K=500$.

Fig. 5.16 shows the contour plot of the posterior distribution together with the scatter plot of particles. Table 5.4 presents the final identification results of elastic constants in the form of statistical parameters. It can be concluded on the basis of the data presented in Table 5.1 and Table 5.4 that the mean values of the posterior distribution are the same as the assumed elastic constants. Moreover, the data in Table 5.3 and Table 5.4 demonstrate that in the case of Young’s modulus the coefficient of variation decreased from 5.1% for the prior distribution to only 0.3% for the posterior distribution. In the case of Poisson’s ratio the coefficient of variation decreased from 5.1% for the prior distribution to 2.0% for the posterior distribution, which indicates that the identified Poisson’s ratio is more uncertain.

Table 5.4: Statistical parameters of the posterior distribution

<table>
<thead>
<tr>
<th>Parameter</th>
<th>E(GPa)</th>
<th>$\nu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean value</td>
<td>67.5</td>
<td>0.33</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0.22</td>
<td>0.01</td>
</tr>
<tr>
<td>CoV (%)</td>
<td>0.3</td>
<td>2</td>
</tr>
</tbody>
</table>
Sequential identification of elastic properties

In this chapter we have presented a sequential approach to parametric identification problems based on Bayesian state estimation formulation. We concentrated on two identification problems of sequential estimation of elastic properties of materials. To solve these problems, we have applied Bayesian filtering using Kalman filters and particle filters also known as sequential Monte Carlo.

The first problem we have solved was concerning a laboratory-scale aluminum frame and the goal to sequentially estimate the Young’s modulus. The solution is based on comparison of the measured and predicted displacements of selected nodes under a quasi-static load. The values of the actual displacements were obtained using optical measurements and a digital image correlation method. The predicted displacements were computed with the use of finite element method. The sequential identification problem has been finally solved by applying two Bayesian filters, namely the Kalman filter and the particle filter. The computer experiments have shown that the sequential approach is able to estimate the Young’s modulus in a few iterations. The elastic constants estimated by the two filters are very similar.

Figure 5.16: Contour and scatter plots of prior and posterior distributions for Young’s modulus and Poisson’s ratio

5.5 Closing remarks

In this chapter we have presented a sequential approach to parametric identification problems based on Bayesian state estimation formulation. We concentrated on two identification problems of sequential estimation of elastic properties of materials. To solve these problems, we have applied Bayesian filtering using Kalman filters and particle filters also known as sequential Monte Carlo.

The first problem we have solved was concerning a laboratory-scale aluminum frame and the goal to sequentially estimate the Young’s modulus. The solution is based on comparison of the measured and predicted displacements of selected nodes under a quasi-static load. The values of the actual displacements were obtained using optical measurements and a digital image correlation method. The predicted displacements were computed with the use of finite element method. The sequential identification problem has been finally solved by applying two Bayesian filters, namely the Kalman filter and the particle filter. The computer experiments have shown that the sequential approach is able to estimate the Young’s modulus in a few iterations. The elastic constants estimated by the two filters are very similar.
In the second problem we have used particle filters to estimate the elastic properties of aluminum thin plates. The identification procedure was based on the comparison of pseudo-experimental and numerical dispersion curves in the context of Bayesian state estimation. Taking into account the assumed experimental errors and considering propagation of errors in the sequential estimation, the uncertainty in the identified value of Poisson’s ratio is 2% and the uncertainty in the estimated value of Young’s modulus is less than 0.5%.
Chapter 6

Final conclusions

6.1 General remarks

This monograph presents applications of Bayesian machine learning in analysis of selected identification problems from mechanics of materials and structures. Six identification problems have been investigated:

1. prediction of fatigue failure of plain concrete specimens (Chapter 3),
2. prediction of compressive strength of high-performance concrete specimens (Chapter 3),
3. structural damage detection in a laboratory steel frame (Chapter 4),
4. structural damage localization in a laboratory steel frame (Chapter 4),
5. sequential estimation of elastic constants of a laboratory frame (Chapter 5),
6. sequential estimation of elastic constants of aluminum thin plates (Chapter 5).

The following measurements have been used as sources of datasets necessary for identification procedures:

- measured fatigue strengths of plain concrete specimens (Chapter 3),
- measured compressive strengths of high-performance concrete specimens (Chapter 3),
- computer-generated dynamic displacements (Chapter 4),
- measured and computer-generated static displacements (Chapter 5),
- computer-generated dynamic displacements (Chapter 5).
The identified parameters have been:

- predicted fatigue strengths of plain concrete specimens (Chapter 3),
- predicted compressive strengths of high-performance concrete specimens (Chapter 3),
- probability of damage occurrence in a structure (Chapter 4),
- probability of damage localization in a structure (Chapter 4),
- sequentially estimated elastic property (Young’s modulus) of a laboratory frame (Chapter 5),
- sequentially estimated elastic properties (Young’s modulus and Poisson’s ratio) of thin plates (Chapter 5).

The applied Bayesian machine learning models and methods have been:

- Bayesian neural networks and Gaussian processes for nonlinear regression (Chapter 3),
- Gaussian mixture model for density estimation (Chapter 4),
- committee of Gaussian mixture models for classification (Chapter 4),
- generalized linear model and relevance vector machine for classification (Chapter 4),
- the Kalman filter and the particle filter for sequential estimation (Chapter 5).

6.2 Summary of results

In this monograph, we analyzed selected identification problems in the context of mechanics of materials and structures. These identification problems were formulated and solved by using various machine learning models and Bayesian inference. The results are presented in short below.

6.2.1 Prediction of concrete properties

In Chapter 3 we considered two problems of concrete properties prediction on the basis of experimental evidence. These problems were formulated as regression problems. In the first problem we analyzed prediction of fatigue failure of plain concrete. The second problem was concerned with prediction of compressive strength of high-performance concrete. We used two nonlinear Bayesian regression models, namely Bayesian neural networks and Gaussian processes.
6.2.2 Structural damage detection and localization

In Chapter 4 we considered a vibration-based structural damage identification problem in a benchmark steel space frame. The problem was divided into two consecutive steps. In the first step, the damage detection problem was formulated as a one-class classification problem and solved by applying a Gaussian mixture model (GMM). This step initially involved modeling of the measured responses of the frame in the form of acceleration time series by the coefficients of auto-regressive models (AR). Next, these AR coefficients were used to build a GMM-based one-class classifier applied for classification of a new AR coefficients vector. In the second step the damage localization problem was formulated as a multi-class classification problem and solved by applying a set Gaussian mixture model (GMMs) and two classification models.

6.2.3 Sequential estimation of elastic constants

In Chapter 5 we applied Bayesian filtering for sequentially solving two problems of elastic constants identification. In the first problem we sequentially estimated the Young’s modulus for a laboratory-scale aluminum frame by applying the Kalman filter and the particle filter. In the second problem we used particle filters to estimate elastic properties of aluminum plates.

6.3 Original contribution

This monograph presents novel approaches to solving selected identification problems in the context of mechanics of materials and structures. The novelty consists in applying various Bayesian machine learning models and methods. We have shown that it is possible to successfully solve regression problems in the context of concrete properties prediction by applying nonlinear regression models with Bayesian inference, such as feed-forward layered neural networks and Gaussian processes. Moreover, in the context of structural damage identification, we have proved that classification models, such as Gaussian mixture models and relevance vector machine, are able to solve damage detection and localization problems by using a benchmark steel frame. Finally, we have shown that the Bayesian state estimation approach can be successfully applied in solving sequential identification problems in the context of determining elastic properties of materials.
6.4 Practical applications

The novel approach to the solutions of identification problems presented in this monograph can be applied in solving various engineering tasks in which we seek to find quantities that are not directly observable. The proposed methods can be successfully applied in the context of noisy measurements and small experimental evidence.

The example identification problems presented in the previous chapters of this work are representative for the following fields of possible applications:

- prediction of concrete quality properties,
- structural health monitoring (SHM),
- sequential estimation of elastic properties of materials.

6.5 Future work

A large part of the work presented in this monograph calls for continuation and still many problems remain open and waiting to be solved. Future research in the field of applications of Bayesian machine learning in the context of mechanics of materials and structures should concentrate on the following areas:

- further investigation of applications of Bayesian neural networks and Gaussian processes in improving the prediction accuracy of models,
- investigation of approximate Bayesian inference methods based on variational Bayes for model selection and predictions,
- development of new applications of Bayesian machine learning in solving inverse problems with uncertainty quantification.
Bibliography


List of Figures

2.1 Plot of the mathematical curve $f(x) = \sin 2\pi x$ applied to generating $N = 30$ patterns used for learning, validation and testing shown using different markers ..................................... 16

2.2 Plots of the 9th order polynomials, trained with different values of $\lambda$; polynomial with the optimal value of $\lambda_{opt} = 1e - 4$ is shown as a solid line ................................................................. 18

2.3 Plots of training and validation curves. Validation curve is used for selecting optimal regularization coefficient $\lambda_{opt}$ for the 9th order polynomial ................................................................. 19

2.4 Plot of the validation curve for selecting the optimal order of polynomial ................................................................. 20

2.5 Plot of polynomials for different orders $M$ (from 0 to 9) ............ 20

2.6 Plot of sequential Bayesian learning for a simple example of inferring model parameters of linear polynomial $y(x; w) = w_1 x + w_0$, based on the example discussed in [11] ................................................................. 24

2.7 Plot of predictive distributions for linear models in the case of an increasing number of learning patterns, based on the example discussed in [11] ................................................................. 25

2.8 Plot of log marginal likelihood curve for optimal polynomial order selection ................................................................. 26

2.9 Plots of the results of finding the optimal hyperparameters $\alpha$ and $\beta$ using the evidence approximation approach ................................................................. 29

2.10 Plots of the curve for the computations of optimal $\gamma$ (left) and the ratio of $\alpha$ and $\beta$ corresponding to the regularization coefficient $\lambda$ (right) ................................................................. 29

3.1 Diagram for the neural network with one hidden layer .................. 36

3.2 Learning and validation error curves and the marginal likelihood function curve for selection of the optimal number of hidden neurons ................................................................. 41

3.3 Results of fatigue tests on concrete samples ............................... 44
3.4 Results of prediction of fatigue tests by using T-BNN on concrete samples ........................................... 47
3.5 Results of prediction of fatigue tests by using GP on concrete samples .................................................. 47
3.7 Mean square error on testing set for networks sampled in last 100 iterations (one of the runs) ......................... 49
3.8 The 28-day HPC compressive strength $f'_c$ as a function of $W/C$ and $S/C$ .............................................. 50
3.9 Box and whisker plot for the HPC dataset [115] ......................................................................................... 52
3.10 Compressive strength $f'_c$ vs. each input variable ..................................................................................... 53
3.11 The HPC dataset visualization using scatter plots .................................................................................... 54
3.12 Validation curves .................................................................................................................................... 57
3.13 Marginal likelihood curve for selection of the number of hidden units in the semi-Bayesian neural network .......................................................... 58
3.14 Estimation of hyperparameters $\alpha$ (left) and $\beta$ (right) ......................................................................... 58
3.15 Estimation of hyperparameters $\lambda = \alpha / \beta$ (left) and $\gamma$ (right) ......................................................... 59
3.16 Scatter plots of training and testing datasets and Gaussian predictive probability distribution for compressive strength prediction with a semi-BNN ............................................................................ 59
3.17 Predictive distribution for compressive strength prediction with a semi-Bayesian neural network, corresponding to the vertical blue line in Fig. 3.16 ................................................................. 60
3.18 Predicted compressive strengths versus measured values ......................................................................... 61

4.1 ASCE benchmark steel frame structure ..................................................................................................... 67
4.2 Example of simulated time series for the benchmark structure ....................................................................... 68
4.3 Plots of eigenvalues from PCA decomposition of dataset ............................................................................. 70
4.4 Visualization of the feature vectors in 3D ...................................................................................................... 71
4.5 Visualization of feature vectors in 2D ........................................................................................................... 72
4.6 Example of Gaussian mixture distribution .................................................................................................. 73
4.7 Example of the EM algorithm for learning a GMM ..................................................................................... 75
4.8 Datasets for learning, validating and testing a GMM in the damage detection step, shown in two-dimensional space defined by the first two features $\alpha_1$ and $\alpha_2$ ........................................................................... 76
4.9 The normal dataset shown in two-dimensional space defined by the first two features $\alpha_1$ and $\alpha_2$. This dataset is used for learning the GMM using the EM algorithm ............................................................................. 76
4.10 Final configuration of the GMM after learning with the EM algorithm ......................................................... 77
4.11 Validation dataset used for estimation of the novelty threshold ................................................................... 77
4.12 Novelty threshold setting results based on the $F_1$ score ........................................................................... 78
4.13 Clusters of feature vectors for three cases of damage .................................................................................. 80
4.14 Confusion matrices for learning of a committee of three GMMs in the case of 16D feature space (left) and in the case of 2D feature space (right) .................................................. 83
4.15 Confusion matrices for learning of the GLM and RVM models in the case of 16D feature space; left: for GLM; right: for RVM .................................................. 83
4.16 Confusion matrices for testing the committee of three GMMs; left: in the 16D feature space; right: in the 2D feature space .................. 84
4.17 Confusion matrices for testing GLM and RVM models in 16D feature space; left: for GLM; right: for RVM ................................. 85
4.18 Confusion matrices for learning results of the committee of six GMMs; left: confusion matrix for the 16D feature space; right: confusion matrix for the 2D feature space ........................................ 86
4.19 Confusion matrices for testing results of the committee of six GMMs; left: confusion matrix for the 16D feature space; right: confusion matrix for the 2D feature space ........................................ 87

5.1 First and second order Markov models .......................... 92
5.2 Dynamic Bayesian network for a sequence of state variables $x_k$ and observable variables $y_k$ (measurements) ..................... 93
5.3 Plot of one update cycle for certain values of transition and observation models ............................................................... 96
5.4 Schematic illustration of the prediction-update-resample cycle of particle filter (detailed description in text) .......................... 98
5.5 Flowchart of the particle filter-based identification of elastic constants ................................................................. 99
5.6 View of the two-storey laboratory-scale aluminum frame and the experimental stand used for frame displacements measurement .... 100
5.7 Schematic illustration of the aluminum frame FEM model and the positions where load has been applied ..................... 101
5.8 Plot of sequential estimation of the Young’s modulus using the Kalman filter and the particle filter for the application of the vertical force (first experiment) .................. 102
5.9 Plot of sequential estimation of Young’s modulus using the particle filter for the application of the vertical force (second experiment) .................. 103
5.10 Plot of sequential estimation of Young’s modulus using the particle filter for the horizontal force (first experiment) .................. 104
5.11 Plot of sequential estimation of Young’s modulus using the particle filter for the horizontal force (second experiment) ........ 105
5.12 Plot of fundamental non-dimensional dispersion curves $S_0$ and $A_0$ computed using equations (5.39) and (5.40) for $\nu = 0.33$ 107
5.13 Plot of fundamental dispersion curves $A_0$ computed for Poisson’s ratio $\nu = 0.33$, half of plate thickness $d = 2$ mm, mass density $\rho = 2700$ kg/m$^3$ and three Young’s moduli $E$ . . . . . . . . . . . . . . . 108

5.14 Plot of the prior and posterior distributions for Young’s modulus. The Gaussian prior distribution (dashed line) has mean value $\mu_{\text{prior}} = 67.0$ GPa and standard deviation $\sigma_{\text{prior}} = 3.35$ GPa (coefficient of variation (CoV) is 5%). The approximate Gaussian posterior distribution (solid line) has mean value $\mu_{\text{pos}} = 67.5$ GPa and standard deviation $\sigma_{\text{pos}} = 0.05$ GPa (coefficient of variation (CoV) is 0.1%) . . . . . . . 110

5.15 Plot of evolution of the mean value of posterior distribution for Young’s modulus and the corresponding one-standard deviation error bars (the solid horizontal line represents Young’s modulus value (67.5 GPa) assumed in the numerical experiments) . . . . . . . . . 111

5.16 Contour and scatter plots of prior and posterior distributions for Young’s modulus and Poisson’s ratio . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 114
List of Tables

3.1 Experimental evidence collected in [34] ........................................ 43
3.2 Statistical parameters of input and output variables ......................... 44
3.3 Generalization performance comparison of various models with 5 and 10 hidden units in predicting concrete fatigue durability. The shown testing errors are the root-mean-squared error (RMSE) and the average percentage error (APE) averaged over 10-fold cross-validation with standard deviation of the mean .......................... 46
3.4 Mean and standard deviation for the ARD hyperparameters values ........ 49
3.5 Statistical parameters of input and output variables .......................... 51
3.6 Correlation coefficients between input variables ............................... 52
3.7 Correlation coefficients between each input variable at power \( p \) and output variable ................................................................. 55
3.8 Performance comparison of models for learning and testing ................. 57

4.1 Results presented as coefficients of success in \( \% \) in detection of scenarios of severe and minor damage in the structure. Results of testing the proposed algorithm for damage detection for various numbers of input variables (16, 3 and 2 (after applying PCA)). The last column presents the coefficients of success in damage detection for the undamaged structure ................................................................. 78
4.2 Learning (L) and testing (T) success ratios (in \( \% \)) of damage localization for three cases of damage and for different numbers of input variables ......................................................... 84
4.3 Learning (L) and testing (T) success ratios (in \( \% \)) of damage localization for six cases of damage and for different numbers of input variables ......................................................... 86

5.1 Assumed values of the aluminum plate parameters applied in the numerical experiments ................................................................. 109
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>Statistical parameters of the prior and posterior distributions for Young’s modulus (mean value, standard deviation and coefficient of variation (CoV))</td>
<td>112</td>
</tr>
<tr>
<td>5.3</td>
<td>Statistical parameters of the Gaussian prior distribution</td>
<td>113</td>
</tr>
<tr>
<td>5.4</td>
<td>Statistical parameters of the posterior distribution</td>
<td>113</td>
</tr>
</tbody>
</table>
This monograph presents the Bayesian machine learning-based approach to selected identification problems in the context of mechanics of structures and materials. Several identification problems can be formulated in this context and solved using Bayesian machine learning methods. For example, concrete properties prediction problems can be formulated as regression problems and solved using Bayesian methods for regression. Similarly, structural damage identification problems can be formulated as classification problems and solved using Bayesian methods for classification. Thus, the main aim of the dissertation is to formulate, solve and discuss selected identification problems related to mechanics of materials and structures by applying Bayesian machine learning methods.

In Chapter 1, the introduction, the motivation and objectives of the work are presented, followed by the outline of the scope of the monograph.

Chapter 2 gives the basics of Bayesian inference in the machine learning context. This chapter starts with some introductory remarks about machine learning. Then Bayesian inference basics are given. Next, an example of exact Bayesian inference is presented for a simple linear regression model. Finally, the chapter ends with some basic information about approximate methods for Bayesian inference.

Chapter 3 is devoted to the problem of concrete properties prediction using Bayesian neural networks and Gaussian processes. We concentrate on two identification problems related to prediction of plain concrete fatigue failure and prediction of high-performance concrete compressive strength.

Chapter 4 demonstrates the application of a Gaussian mixture model, generalized linear models and relevance vector machine for vibration-based structural damage identification problems. The first part of the chapter focuses on the solution of a damage detection problem and the second part – on a damage localization task.

The most common approach to solving identification problems is based on a batch approach. But there is also an on-line or sequential approach which can be useful in some identification problems. In Chapter 5 we present two applications of Bayesian filtering to two identification problems in the context of sequential estimation of elastic properties of materials.

Finally, Chapter 6 summarizes and concludes the monograph as well as presents some directions which future research could take and some prospective applications of Bayesian machine learning methods.
Bayes maschinellen Lernens in der Analyse von ausgewählten Identifikationsprobleme in der Mechanik der Werkstoffe und Strukturen

Zusammenfassung


Schließlich Kapitel 6 fasst und schließt die Monographie und Geschenke Einige zukünftige Richtungen der Arbeit und Chancen des Einsatzes von Maschinen Bayesian chine Lernmethoden.
Bayesowskie uczenie maszynowe w analizie wybranych problemów identyfikacji w mechanicznie materiałowych i konstrukcji

Streszczenie


W rozdziale pierwszym pracy przedstawiono obszerne wprowadzenie do tematyki rozprawy oraz sformułowano cele i zakres pracy. We wprowadzeniu dokonano wyrazającego przeglądu literatury dotyczącej rozpatrywanych zagadnień. Następnie przedstawiono zasadnicze cele pracy oraz omówiono szczegółowo zakres pracy.

Zasadnicza część rozdziału drugiego zawiera krótki opis podstaw teoretycznych oraz przykład wnioskowania bayesowskiego w kontekście metod uczenia maszynowego. Początkowa część rozdziału drugiego prezentuje definicję oraz podział učenia maszynowego. W tej części rozdziału przedstawiono też w zarysie wybrane zastosowania uczenia maszynowego do rozwiązywania zadań uczenia nadzorowanego takich jak zadania regresji i zadania klasyfikacji oraz do rozwiązywania zadań uczenia nienadzorowanego. W drugiej części rozdziału skupiono uwagę na podstawowych regulach rachunku prawdopodobieństwa takich jak reguła sumy, reguła iloczynu oraz twierdzenie Bayesa. W dalszej części tego rozdziału przedstawiono szczegółowo wnioskowanie bayesowskie na przykładzie modelu liniowego dla zadania regresji.

W rozdziale trzecim pracy przedstawiono zastosowanie wybranych nielinowych modeli regresyjnych - bayesowskich sieci neuronowych i procesów gaussowskich - w zadaniach identyfikacji własności betonów. Skupiono uwagę na dwóch zadaniach: identyfikacji trwałości zmnęczeniowej betonów zwykłych oraz identyfikacji wytrzymałości na ściskanie betonów wysokiej wytrzymałości.


Rozdział szósty zawiera obszerne podsumowanie monografii. W szczególności, rozdział ten obejmuje omówienie najważniejszych wniosków wyprowadzonych na podstawie przeprowadzonych analiz oraz proponowane kierunki dalszych badań wynikające z uzyskanych dotychczas wyników.