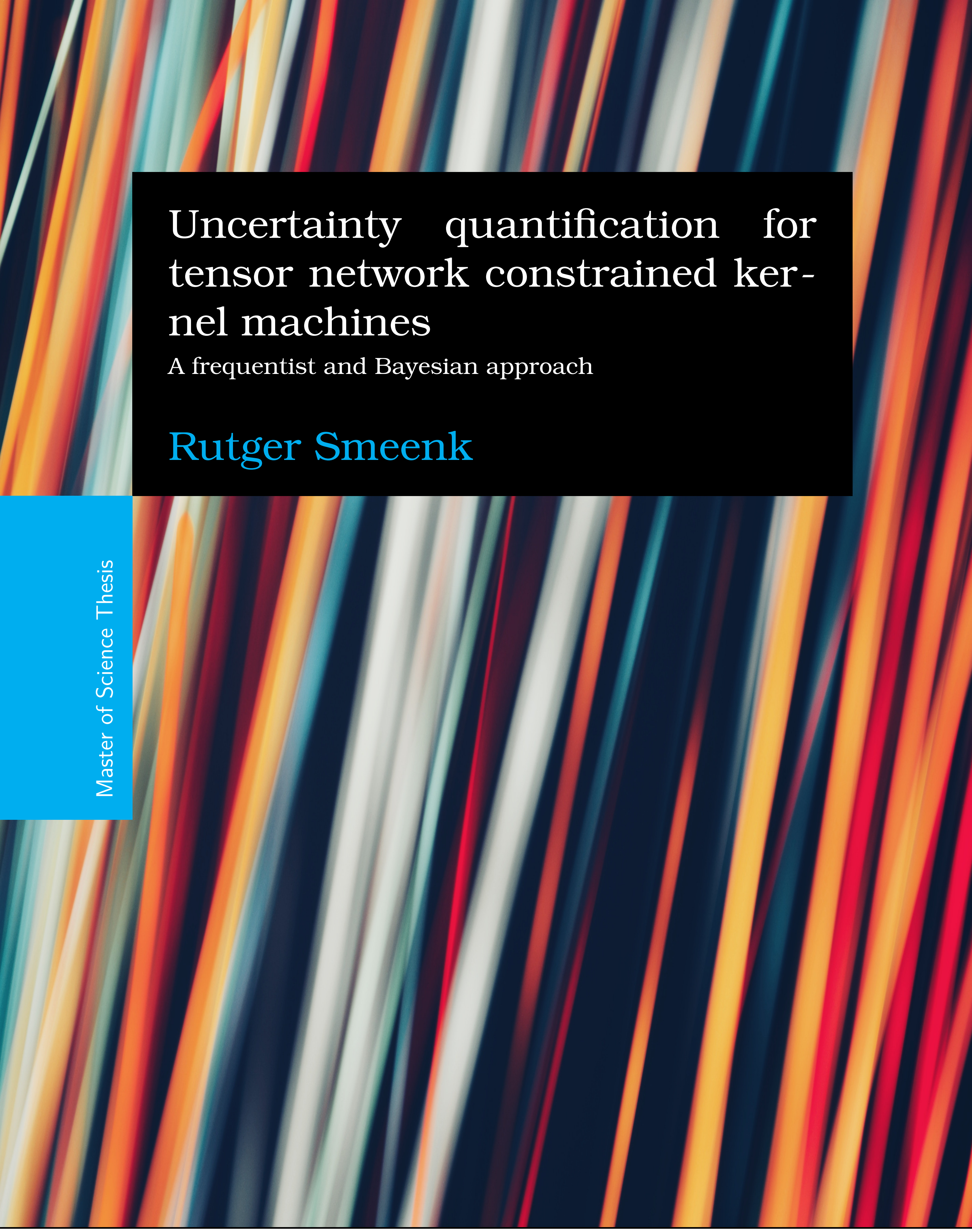


Uncertainty quantification for tensor network constrained kernel machines

A frequentist and Bayesian approach

Rutger Smeenk

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Uncertainty quantification for tensor network constrained kernel machines

A frequentist and Bayesian approach

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Rutger Smeenk

September 27, 2023

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

This research aims at quantifying the uncertainty in the predictions of tensor network constrained kernel machines, focusing on the Canonical Polyadic Decomposition (CPD) constrained kernel machine. Constraining the parameters in the kernel machine optimization problem to be a CPD results in a linear computational complexity in the number of features, whereas the original problem suffers heavily from the curse of dimensionality as the number of parameters scale exponentially. By employing a product feature map with polynomial features, the original data input is transformed to a higher-dimensional space.

Three different methods are investigated for quantifying the uncertainty of the predictions of the CPD constrained kernel machine. Firstly, the delta method is proposed which is a frequentist approach that linearizes a nonlinear parametric model around the estimated model. By estimating the covariance of the model parameters, the delta method can estimate the uncertainty in the model predictions based on the estimated parameter uncertainties. The delta method is compared to two other methods that are able to reflect the prediction uncertainty: the Bayesian method and Single Bayesian Core (SBC) method. The Bayesian method treats the parameters in the factor matrices of the CPD as probability distributions rather than single values and the SBC method incorporates both frequentist and Bayesian aspects. The three different methods are assessed and compared based on the degree to which the constructed uncertainty intervals are correct and informative.

It was found by regression and classification experiments that all three methods can provide valuable uncertainty quantification measures in terms of correctness and informativeness for the CPD constrained kernel machine. However, the Bayesian method provides in general more conservative uncertainty intervals compared to the delta and SBC method. A major drawback of the Bayesian method is its lack of scalability as the size of the mean and covariance, constructed by the unscented transform in the Bayesian method, scale exponentially. Furthermore, the delta and SBC method produce high quality uncertainty intervals and the methods provide remarkably similar uncertainty quantification on the prediction error variance.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

More than ever before, data is used to generate knowledge in research and industry. Processing data in a clever way can enhance valuable analysis and decision-making. In today's era, the availability of increasingly larger datasets leads to a need to extract important and desired information from the data while coping with the extreme computational burden that is accompanied with processing large amounts of data. Artificial intelligence (AI) algorithms process the large amounts of data in order to generate valuable information. Applications of AI algorithms can be found in both research and product development, for example, AI powered robots to grow plants in the farming industry use less resources [46], protein structure prediction for revolutionary breakthroughs in biological sciences [21] and the prediction of renewable energy availability for improving energy utilization [11]. These developments and advances are fueled by the exponential growth in both AI model size and data availability [58].

Parallel to the increase in model and data size are the negative economic, social and environmental side effects [33]. For example, the environmental impact of training large-scale machine learning models causes CO_2 emissions up to 280.000 kg and causes up to 3 million dollars for cloud compute costs for training natural language processing models [24, 50]. In addition, the social downside stems from the fact that the trend in the AI community is to seek for models that opt for the best results in the sense that these models have the highest accuracy compared to other benchmark models. Training these kind of models with increased training data comes with exponentially increased computational costs, excluding researchers with fewer resources [33]. Recently, more researchers have called for a switch in the current AI community attitude towards heavily taking into account and assessing the efficiency of algorithms instead of solely focusing on the accuracy [51, 45].

Many datasets in science and engineering, for example, remote sensing and biological data, are represented by enormous datasets that possess billions of elements and are commonly represented by large block matrices or tensors, which are multidimensional generalizations of matrices. In order to deal with such massive data structures, tensor algorithms have proven to be suitable for processing large scale datasets. This research focuses on tensor networks, which are also called tensor decompositions, as a tool for increased efficiency by reducing the computational burden of large-scale learning. Tensor networks decompose higher dimensional tensors into sparsely interconnected small scale lower dimensional core tensors. The central idea with tensor networks is to compress data by approximating it with a tensor network, resulting in a compressed data format. Hence, the goal with tensor networks is to preserve the valuable information within the data while the data is structured in a compressed format. Tensor networks allow for a vast decrease of data elements while maintaining the inherent relationships within the original data. Hence, tensor networks can be used to efficiently, and thereby sustainably, apply machine learning models on large-scale learning problems while still maintaining competitive accuracy. Tensor networks are currently not widely adopted in AI, thereby, more researchers in the AI community may grasp the opportunity of integrating tensor networks in large-scale models in order to reduce compute significantly [33, 5].

Yet another challenge facing machine learning models is an increasing need for the models to be reliable as machine learning is increasingly becoming dominant in various different fields in technology and society. This is because the results of machine learning models are not always fully reliable. More specifically, uncertainty in the model predictions can arise from stochastic environments which can also be partially unobservable. On a similar note, uncertainty can be embedded in collecting, organizing and analyzing data, where minor error can easily accumulate through positive feedback loops when analyzing big datasets [47]. Moreover, uncertainty in the model predictions results from limited data, hyperparameters, overparameterization, optimization and sampling errors as well as model misspecification [41]. Providing quantitative uncertainty measures that coexist with model predictions is extremely valuable in various real-world situations. Hence, model predictions from machine learning inference could be of little intrinsic practical value if the uncertainty of the model is not well quantified. For example, in space weather prediction, some very rare occurrences such as geomagnetic storms are hard to predict, although data on solar flare, solar wind, and changes in the Earth's magnetosphere are available. Therefore, data and model parameter uncertainty is critical in such machine learning models [47, 41]. Hence, it is important to assess to what extend the results are respectable. In this regard, uncertainty quantification of machine learning models is an important task and has seen increased interest within the scientific community in recent years.

The machine learning paradigm that is focused on in this research is supervised learning which is one of the most important cornerstones in machine learning in general and is used extensively in various industries and technologies. In supervised learning, labelled classes are present where the task is to most accurately identify unseen data [25]. Supervised kernel machines are studied in this research in which test inputs are projected to a higher dimensional feature space. The present research explores how to provide valuable uncertainty quantification for efficient and sustainable large-scale learning supervised kernel machines.

1-1 Problem formulation

This research aims at unifying kernel machines, tensor networks and uncertainty quantification. Whilst kernel machines can be formulated and solved in the primal space, rendering a direct parameterization, they are usually formulated and solved in the dual space by employing the kernel trick [20]. However, such kernel based methods require at least $\mathcal{O}(N^2)$ in storage complexity and $\mathcal{O}(N^3)$ in computational complexity, where N is the number of datapoints [59]. With large datasets, these methods become practically infeasible. On the other hand, the parameters in the primal formulation scale exponentially with the number of features in the dataset. To alleviate this exponential increase, tensor networks are employed with kernel machines. The particular tensor network that is considered is the Canonical Polyadic Decomposition (CPD). The CPD imposes inherent nonlinearity into the otherwise linear in the parameters kernel machine. The parameters in the CPD constrained kernel machine can be learned by means of the Alternating Linear Scheme (ALS) which iteratively optimizes a single factor matrix, keeping the others constant [55].

From the uncertainty quantification perspective, several methods of uncertainty quantification are explored and tested for a supervised kernel machine in which the parameters are constrained to be a CPD. Three different methods for uncertainty quantification are investigated: the delta method [10, 32, 22], the Bayesian method [34] and the Single Bayesian Core (SBC) method [35]. The delta method treats the parameters as single point values whereas the Bayesian method treats the parameters as probability distributions. The SBC method possesses both frequentist and Bayesian elements. In the uncertainty quantification assessment, both correctness and informativeness needs to be assessed. Correctness relates to accurately quantifying and representing uncertainty in a model or system. On the other hand, informativeness pertains to the extent to which the results of the uncertainty quantification process provide valuable insights and useful information. In this regard, the following research question can be stated:

> ***How to provide uncertainty quantification for tensor network constrained kernel machine, focusing on the CPD constrained kernel machine, and which of the investigated methods is most accurate in terms of correctness and informativeness?***

In order to answer this research question, two sub-questions can be formulated that will act as guidelines to achieve the main research question:

- *How can the uncertainty quantification methods be applied to tensor network constrained kernel machines?*
- *Which of the researched methods performs best in terms of correctness and informativeness?*

1-2 Implementation

All experiments were performed on a Lenovo laptop with 16 GB of RAM and an Intel Core i7-9750 CPU running at 2.60 GHz. In order to replicate the experiments performed in this research, the code for the experiments is provided in: https://github.com/RutgerSmeenk/uncertainty_quantification_CPD_kernel_machines.git.

1-3 Thesis outline

In this research, tensors, tensor operations and the CPD are discussed in Chapter 2. Knowledge on tensors and the CPD is needed in order to introduce the CPD constrained kernel machine in Chapter 3. After discussing the model that is dealt with in this research, the uncertainty quantification aspect from a frequentist point of view is discussed in Chapter 4. In turn, the uncertainty quantification aspect from a Bayesian point of view is discussed in Chapter 5. Thereafter, the assessment criteria and the experiment results are outlined in Chapter 6. Finally, a discussion on the obtained results and a conclusion are provided in Chapter 7.

Chapter 2

Tensor networks

This chapter will elaborate on tensors and tensor networks, focussing on the Canonical Polyadic Decomposition (CPD), which will be of fundamental value in overcoming the difficulty of applying kernel machines to high dimensional data, also known as the curse of dimensionality.

2-1 Tensors

2-1-1 Tensor basics

The main concepts and operations which will be described are based on the descriptions by [23, 3, 5, 40].

Tensors are multidimensional generalizations of matrices, and as such, a tensor describes a multidimensional data framework. Hence, a zero-dimensional tensor is a scalar a, a one-dimensional tensor is a vector $\mathbf{a}$, a two-dimensional tensor is a matrix $\mathbf{A}$ and a D-dimensional tensor is represented by $\boldsymbol{\mathcal{A}}$. The order of a tensor is the number of its dimensions, where modes or ways are also used for the naming of dimensions. Since only real-valued tensor entries are considered throughout this study, a tensor is denoted by $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$. Furthermore, a specific entry within the tensor is denoted by $\boldsymbol{\mathcal{A}}(i_1, i_2, \ldots, i_D)$, but can also be denoted by $a_{i_1, i_2, \ldots, i_D}$. The entry number for a specific dimension cannot be greater than the size of the dimension $1 \leqslant i_k \leqslant I_k$. Hence, $I_1, I_2, \ldots, I_D$ are the dimension sizes of the tensor. Schematically, tensors can be graphically described by diagrams. In this framework, scalars, vectors, matrices and tensors are defined in Figure 2-1. The edges within a tensor network graph represent the indices for a particular structure. In this regard, scalars have zero free edges, whereas vectors and matrices have one and two free edges respectively.

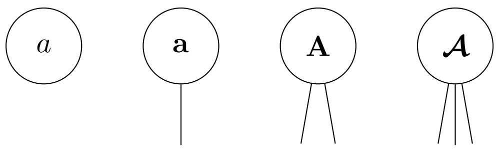


Figure 2-1: Graphical representation of a scalar a, vector $\mathbf{a}$, matrix $\mathbf{A}$ and tensor $\boldsymbol{\mathcal{A}}$

An important notion that is frequently used when working with tensors is multi-indexing in which every combination of indices can be recovered. A multi index $i = \overline{i_1 i_2 \cdots i_D}$ can take all possible combinations of values of indices $i_1, i_2, \ldots, i_D$ for $i_d = 1, 2 \ldots, I_d$, $d = 1, 2, \ldots, D$ and in a specific order. The formula for multi-indexing is given by:

$$\overline{i_1 i_2 \cdots i_D} = i_1 + (i_2 - 1)I_1 + (i_3 - 1)I_1 I_2 + \cdots + (i_D - 1)I_1 \cdots I_{D-1}. \tag{2-1}$$

A tensor can be vectorized by converting a multi-dimensional tensor into a one-dimensional vector while preserving the order of its elements. The elements of a vectorized tensor can be mapped to the tensor entry by means of the formula for multi-indexing described in equation (2-1) as follows:

$$\text{vec}(\boldsymbol{\mathcal{A}})_{\overline{i_1 i_2 \cdots i_D}} = a_{i_1, i_2, \ldots, i_D}. \tag{2-2}$$

Furthermore, matricization is frequently used when working with tensors. Hereby, a tensor is unfolded into a matrix where the elements of the original tensor are reordered. Matricization can be performed with respect to different fixed modes $d \in \{1, 2, \ldots, D\}$ of the tensor. The mode-d matricization of a D-way tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ is defined by the following matrix:

$$\mathbf{A}_{(d)} \in \mathbb{R}^{I_d \times I_1 I_2 \cdots I_{d-1} I_{d+1} \cdots I_D} \tag{2-3}$$

where I_d and $I_1 I_2 \cdots I_{d-1} I_{d+1} \cdots I_D$ are the amount of rows and columns of the matrix $\mathbf{A}_{(d)}$ respectively. By means of multi-indexing, the entries are defined as:

$$\left(\mathbf{A}_{(d)}\right)_{i_d, \overline{i_1 \cdots i_{d-1} i_{d+1} \cdots i_D}} = a_{i_1, i_2, \ldots, i_D}. \tag{2-4}$$

The mode-d fibers of the tensor $\boldsymbol{\mathcal{A}}$ are the columns of the mode-d matricization $\mathbf{A}_{(d)}$.

On the other hand, tensorization is the transformation of a vector, matrix or low-order tensor into a higher-order tensor. Tensorization is the reversed process of vectorization or matricization. The process of tensorization will be denoted by $\mathcal{T}$. When performing vectorization, matricization or tensorization, the multi-indexing formula described in equation (2-1) can be employed in order to perform the computations such that the entries are rightly reordered. In conclusion, vectorization, matricization and tensorization are reshaping operations by means of multi-indexing.

Some mathematical products are extensively used when working with tensors. Firstly, the Kronecker product $\otimes$ of matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ and matrix $\mathbf{B} \in \mathbb{R}^{K \times L}$ is given by

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \dots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \dots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \dots & a_{IJ}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{IK \times JL}. \tag{2-5}$$

Next, the Hadamard product $*$ computes the element-wise matrix product. The Hadamard product of matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ and matrix $\mathbf{B} \in \mathbb{R}^{I \times J}$ is given by

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \dots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \dots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \dots & a_{IJ}b_{IJ} \end{bmatrix} \in \mathbb{R}^{I \times J}. \tag{2-6}$$

Another important product is the Khatri-Rao product $\odot$. The Khatri-Rao product of matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ and matrix $\mathbf{B} \in \mathbb{R}^{K \times J}$ is given by

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_J \otimes \mathbf{b}_J \end{bmatrix} \in \mathbb{R}^{IK \times J} \tag{2-7}$$

where $\mathbf{a}_j$ are the columns of $\mathbf{A}$ and $\mathbf{b}_j$ are the columns of $\mathbf{B}$. The defined Khatri-Rao product constitutes the column-wise Khatri-Rao product.

The different matrix products have distinct properties. For the Hadamard and Khatri-Rao product the most important properties are:

$$\begin{aligned} \mathbf{A} * \mathbf{B} &= \mathbf{B} * \mathbf{A} \\ (\mathbf{A} * \mathbf{B}) * \mathbf{C} &= \mathbf{A} * (\mathbf{B} * \mathbf{C}) \\ \mathbf{A} \odot (\mathbf{B} \odot \mathbf{C}) &= (\mathbf{A} \odot \mathbf{B}) \odot \mathbf{C}. \end{aligned} \tag{2-8}$$

Another important property that combines the Hadamard product and Khatri-Rao product is given as follows:

$$(\mathbf{A} \odot \mathbf{A})^T (\mathbf{B} \odot \mathbf{B}) = \mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B}. \tag{2-9}$$

Other properties that are used later in this research are regarding the matrix vectorization operation and matrix calculus [40]. For the vectorization operation the following can be stated:

$$\mathrm{Tr}(\mathbf{A}^T\mathbf{B}) = \mathrm{vec}(\mathbf{A})^T \mathrm{vec}(\mathbf{B}) \tag{2-10}$$

where Tr() is the trace of a matrix. In addition, vectorization can directly be used with the Kronecker product for expressing matrix multiplication as a linear transformation of matrices as follows

$$\mathrm{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\mathrm{vec}(\mathbf{B}). \tag{2-11}$$

Moreover, concerning matrix calculus, the following derivative can be stated:

$$\frac{\partial \mathrm{Tr}(\mathbf{X}\mathbf{X}^T\mathbf{C})}{\partial \mathbf{X}} = \mathbf{X}^T\mathbf{C} + \mathbf{X}^T\mathbf{C}^T. \tag{2-12}$$

Two other properties that will be used are with regards to vector calculus. Firstly the following can be stated:

$$\frac{\partial \mathbf{A}\mathbf{x}}{\partial \mathbf{x}} = \mathbf{A} \quad \text{OR} \quad \mathbf{A}^T. \tag{2-13}$$

where $\mathbf{A}$ is not a function of $\mathbf{x}$ and the derivative equals $\mathbf{A}$ or $\mathbf{A}^T$ depending on the chosen layout of the derivative which can either be numerator, i.e. $\mathbf{y}$ and $\mathbf{x}^T$, or denominator, i.e. $\mathbf{y}^T$ and $\mathbf{x}$. Similarly, depending on the layout, the following chain rule can be defined for a vector derivative:

$$\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{x}} = \frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{x}} \quad \text{OR} \quad \frac{\partial \mathbf{u}}{\partial \mathbf{x}}\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}}. \tag{2-14}$$

2-1-2 Tensor operations

An important tensor operation is the mode-$\binom{k}{d}$ product, also known as tensor contraction of two tensors $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_K}$, which have common modes $I_d = J_k$ produces an $D + K - 2$-order tensor $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{d-1} \times I_{d+1} \times \cdots \times I_D \times J_1 \times \cdots \times J_{k-1} \times J_{k+1} \times \cdots \times J_K}$. This can be formulated as:

$$\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \times_d^k \boldsymbol{\mathcal{B}}. \tag{2-15}$$

Then, the entries are computed as follows:

$$c_{i_1,\ldots,i_{d-1},i_{d+1},\ldots,i_D,j_1,\ldots,j_{k-1},j_{k+1},\ldots,j_K} = \sum_{i_d=1}^{I_d} a_{i_1,\ldots,i_{d-1},i_d,i_{d+1},\ldots,i_D} b_{j_1,\ldots,j_{k-1},j_k,j_{k+1},\ldots,j_K}. \tag{2-16}$$

This operation is a contraction of two tensors in single common mode. Two tensors can be contracted in several modes. The tensor contraction operation can be viewed as a higher dimensional analogue of matrix multiplication, inner product and outer product. For example, considering the mode-$\binom{k}{d}$ product of two matrices $\mathbf{A}$ and $\mathbf{B}$ can be written as $\mathbf{A} \times_2^1 \mathbf{B} = \mathbf{A} \times^1 \mathbf{B} = \mathbf{A}\mathbf{B}$, where the subscript and superscript are omitted for simplicity since we have common modes $I_D = J_1$ as we are dealing with matrices. In a tensor network diagram, connected edges represent tensor contraction. Hence, when two nodes are connected this means two tensors can be contracted in order to produce a single tensor by summing over all possible values of repeated indices. For example, tensor index contraction over index i_3 between tensor $\boldsymbol{\mathcal{A}}$ and tensor $\boldsymbol{\mathcal{B}}$ which are both 3-way tensors, results in a 4-way tensor $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \times_3^3 \boldsymbol{\mathcal{B}}$ as follows:

$$\mathcal{C}(i_1, i_2, i_4, i_5) = \sum_{i_3=1}^{I_3} \mathcal{A}(i_1, i_2, i_3)\mathcal{B}(i_3, i_4, i_5). \tag{2-17}$$

Tensor index contraction of two 3-way tensors in which the third index is contracted: is visualized in Figure 2-2.

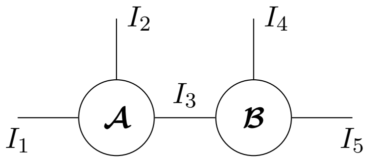


Figure 2-2: Tensor index contraction over the third index between two 3-way tensors $\mathcal{A}$ and $\mathcal{B}$

Further, the product between a tensor and a matrix is described by the mode-d product, also known as multilinear product, of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$ and a matrix $\mathbf{B} \in \mathbb{R}^{K \times I_d}$ results in a tensor

$$\mathcal{C} = \mathcal{A} \times_d \mathbf{B} \in \mathbb{R}^{I_1 \times \cdots \times I_{d-1} \times K \times I_{d+1} \times \cdots \times I_D} \tag{2-18}$$

where the entries are given by

$$c_{i_1,i_2,\ldots,i_{d-1},k,i_{d+1},\ldots,i_D} = \sum_{i_d=1}^{I_d} a_{i_1,i_2,\ldots,i_D} b_{k,i_d}. \tag{2-19}$$

Furthermore, another important tensor operation is the inner product. The inner product between two tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ is computed by summing over all indices as follows:

$$c = \langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_d=1}^{I_D} a_{i_1,i_2,\ldots,i_D} b_{i_1,i_2,\ldots,i_D} \tag{2-20}$$

where c yields a scalar. As an example, the inner product of two 3-way tensors results in a scalar $c = \langle \mathcal{A}, \mathcal{B} \rangle = \mathcal{A} \times_{1,2,3}^{1,2,3} \mathcal{B} = \mathcal{A} \bar{\times} \mathcal{B} = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \sum_{i_3=1}^{I_3} a_{i_1,i_2,i_3} b_{i_1,i_2,i_3}$. Here, $\bar{\times}$ denotes the contraction of all the modes.

The Frobenius norm of the tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ is given by $||\mathcal{A}||_F = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$. Another way to represent the inner product between two tensors is by means of the vec$(\cdot)$ function. The inner product product between two tensors can alternatively be computed by vectorizing the tensors first and then multiplying the vectors by means of the transpose as follows:

$$\langle \mathcal{A}, \mathcal{B} \rangle = \langle \text{vec}(\mathcal{A}), \text{vec}(\mathcal{B}) \rangle = \text{vec}(\mathcal{A})^T \text{vec}(\mathcal{B}) = \langle \mathcal{A}, \mathcal{B} \rangle_F. \tag{2-21}$$

Hence, the inner product $\langle \mathcal{A}, \mathcal{B} \rangle$ equals the Frobenius inner product $\langle \mathcal{A}, \mathcal{B} \rangle_F$.

There also exists an outer product between tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ and $\mathcal{B} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_K}$

$$\boldsymbol{\mathcal{A}} \circ \boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{C}} \tag{2-22}$$

where $\boldsymbol{\mathcal{C}}$ is a $(D+M)$-th order tensor with elements $c_{i_1,\ldots,i_D,j_1,\ldots,j_M} = a_{i_1,\ldots,i_D} b_{j_1,\ldots,j_M}$. Another important concept is the notion of a rank-1 tensor. A D-way tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ is a rank-1 tensor if it can be written as the outer product of D vectors. Hence,

$$\boldsymbol{\mathcal{A}} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)}, \ldots, \circ\, \mathbf{a}^{(D)} \tag{2-23}$$

where '$\circ$' represents the outer product and the vector $\mathbf{a}^{(d)} \in \mathbb{R}^{I_d}$, $d = 1, 2, \ldots, D$, has size d. The visualization of a rank-1 order tensor for a 3-way tensor is depicted in Figure 2-3.

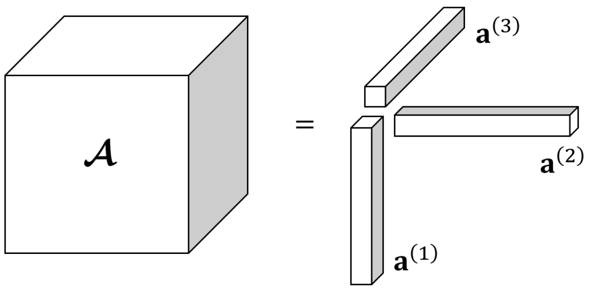


Figure 2-3: Graphical representation a 3-way rank-1 tensor

From an element wise point of view the specific element in a rank-1 tensor is computed by

$$a_{i_1,i_2,\ldots,i_D} = a^{(1)}_{i_1} a^{(2)}_{i_2} \ldots a^{(D)}_{i_D}. \tag{2-24}$$

Finally, there is a relationship between the Kronecker product and the outer product between D vectors $\mathbf{a}^{(d)} \in \mathbb{R}^{I_d}$ where $d = 1, 2, \ldots, D$ as [4]:

$$\operatorname{vec}(\mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(D)}) = \mathbf{a}^{(D)} \otimes \mathbf{a}^{(D-1)} \otimes \cdots \otimes \mathbf{a}^{(1)}. \tag{2-25}$$

2-2 Tensor networks

The main idea of low-rank tensor approximations, which are the same as low-rank tensor networks, is that by employing a tensor network format, the computational costs and storage costs can be drastically reduced. This is because tensor networks cause large-scale data to be approximated in a highly compressed and distributed format [5]. An important remark is that tensor compression by means of tensor network formats almost always comes with incurring some compression error since the low-rank tensor representation is an approximation of the original tensor. For tensors with dimension equal to two ($D = 2$) (matrices), the tensor decompositions boil down to the well-known singular value decomposition (SVD), while there exist various other decompositions for matrices such as the Schur, Jordan, LU and QR decomposition [13]. For higher order tensors ($D \geqslant 3$) the decompositions vary considerably, are constructed in different ways and have special characteristics and properties.

In tensor networks, the original tensor is decomposed in a number of factors. The number of factors in the decomposition corresponds to the tensor network rank R or the tensor network ranks $\{R_1, R_2, \ldots, R_N\}$. In general, a low-rank tensor network approximates the original tensor with much less elements within its decomposition. This can be a major advantage as the number of elements in a tensor to be stored is $\prod_{d=1}^{D} I_d \leqslant I^D$, where I_d is the size of each mode and $I = \max_d\{I_d\}$. Hence, the memory requirements scale exponentially in D, by which the storage complexity of the original tensor is $\mathcal{O}(I^D)$, which make high-dimensional data difficult to work with.

2-2-1 Canonical Polyadic Decomposition

One of the most well-known tensor decompositions is the CPD. The CPD factorizes a tensor into a sum of component rank-1 tensors. Consider a D-way tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$, the goal is to write this tensor as a linear combination of rank-1 tensors. This is given as:

$$\boldsymbol{\mathcal{A}} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(D)}. \tag{2-26}$$

In this formulation R is the tensor rank which is the smallest value for which equation (2-26) holds exactly. By limiting R, a low-rank approximation of $\boldsymbol{\mathcal{A}}$ is obtained, resulting in less elements needed to represent the tensor. The visualization of the CP decomposition for a 3-way tensor is depicted in Figure 2-4.

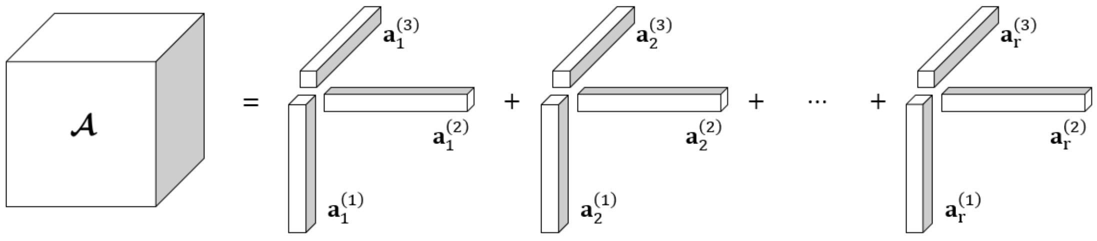


Figure 2-4: Graphical representation a 3-way CP decomposition

The factor matrices are defined as the concatenation of the components in the CPD, i.e. $\mathbf{A}^{(d)} = [\mathbf{a}_1^{(d)}\ \mathbf{a}_2^{(d)}\ \ldots\ \mathbf{a}_r^{(d)}] \in \mathbb{R}^{I_d \times R}$, $d = 1, \ldots, D$. Furthermore, the CPD can also be expressed by normalizing the columns of the factor matrices to length one:

$$\boldsymbol{\mathcal{A}} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(D)} = [\![\lambda; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(D)}]\!] \tag{2-27}$$

where in this formulation the norms are comprised in the vector $\boldsymbol{\lambda} \in \mathbb{R}^R$ and $\mathbf{A}^d \in \mathbb{R}^{I_d \times R}$, $d = 1, \ldots, D$ and $[\![\cdot]\!]$ is called the Kruskal operator. Furthermore, the mode-d matricization is given by:

$$\mathbf{A}_{(d)} = \mathbf{A}^{(d)} \boldsymbol{\Lambda} (\mathbf{A}^{(D)} \odot \cdots \odot \mathbf{A}^{(d+1)} \odot \mathbf{A}^{(d-1)} \odot \ldots \mathbf{A}^{(1)})^T \tag{2-28}$$

where $\mathbf{\Lambda} = \text{diag}(\boldsymbol{\lambda})$.

The rank of a tensor $\boldsymbol{\mathcal{A}}$, also called the CP rank, can be defined as the smallest number of rank-1 terms in an exact CP decomposition, where the CP decomposition is an exact representation of the tensor $\boldsymbol{\mathcal{A}}$. In other words, the rank of a tensor is the smallest number of rank-1 tensors that can generate $\boldsymbol{\mathcal{A}}$ as their sum. It has been proven in [14] that finding the tensor rank over the natural numbers is NP-hard. Such NP-hard problems are unknown to be solvable in polynomial time [19].

2-2-2 Basic operations of the CPD

In this subsection, some important tensor operations that are specific to the CPD are formulated. These operations are valuable at later stages of the research. Vectorization for a tensor $\boldsymbol{\mathcal{A}}$, given it is a CPD, can be performed by means of Khatri-Rao products as follows:

$$\text{vec}(\boldsymbol{\mathcal{A}}) = \left(\bigodot_{d=D}^{1} \mathbf{A}^{(d)} \right) \boldsymbol{\lambda} \tag{2-29}$$

where $\bigodot_{d=D}^{1}$ is a sequence of Khatri-Rao products, $\mathbf{A}^{(d)} \in \mathbb{R}^{I_d \times R}$ are the factor matrices in the CP decomposition and $\boldsymbol{\lambda}$ are the weights of the CPD which is a vector of non-zero entries of the diagonal core tensor $\mathbf{\Lambda} \in \mathbb{R}^{R \times R \times \cdots \times R}$. The non-normalized CPD effectively has a norm vector $\boldsymbol{\lambda}$ equal to vector of ones $\mathbf{1}_R$. This is the simplest and most trivial version of the CPD.

Another important property can now be defined by substituting equation (2-29) into the inner product formulation for tensor defined in equation (2-21) and then applying the property defined in equation (2-9). This results in:

$$\begin{aligned} ||\boldsymbol{\mathcal{A}}||_F^2 &= \langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{A}} \rangle = \text{vec}(\boldsymbol{\mathcal{A}})^T \text{vec}(\boldsymbol{\mathcal{A}}) \\ &= \boldsymbol{\lambda}^T \left(\mathbf{A}^{(1)} \odot \cdots \odot \mathbf{A}^{(D)} \right)^T \left(\mathbf{A}^{(1)} \odot \cdots \odot \mathbf{A}^{(D)} \right) \boldsymbol{\lambda} \\ &= \boldsymbol{\lambda}^T \left(\mathbf{A}^{(1)^T} \mathbf{A}^{(1)} * \mathbf{A}^{(2)^T} \mathbf{A}^{(2)} * \cdots * \mathbf{A}^{(D)^T} \mathbf{A}^{(D)} \right) \boldsymbol{\lambda}. \end{aligned} \tag{2-30}$$

In the formulation in equation (2-30), the storage complexity is vastly reduced. This is due to the sequence of Khatri-Rao products, with resulting storage complexity of $\mathcal{O}(I^D R)$, that is written in terms of a sequence of Hadamard products with a storage complexity of $\mathcal{O}(R^2)$. Likewise, the Frobenius inner product of a CPD tensor and a rank-1 tensor can be stated since a rank-1 tensor is a CPD of rank-1. This allows for computational gains as the tensors do not have to be computed explicitly first. With the CPD tensor $\boldsymbol{\mathcal{A}} = [\![\boldsymbol{\lambda}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(D)}]\!]$ and rank-1 tensor $\boldsymbol{\mathcal{B}} = \mathbf{b}^{(1)} \circ \mathbf{b}^{(2)} \circ \cdots \mathbf{b}^{(D)}$ the Frobenius inner product is given as:

$$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \left(\mathbf{b}^{(1)^T} \mathbf{A}^{(1)} * \mathbf{b}^{(2)^T} \mathbf{A}^{(2)} * \cdots * \mathbf{b}^{(D)^T} \mathbf{A}^{(D)} \right) \boldsymbol{\lambda}. \tag{2-31}$$

Chapter 3

Supervised kernel machines with the CPD

In this chapter, supervised learning with the CPD constrained kernel machine will be discussed. Firstly, supervised learning from a general machine learning perspective will be discussed. Thereafter, linear models with nonlinear feature mapping and kernel methods within supervised learning will be discussed. Finally, a product feature mapping will be elaborated upon which will form the basis for the next chapter where tensors are considered.

3-1 Kernel machines in supervised learning

Kernel machines are widely used in unsupervised and supervised learning tasks such as regression and classification. In regression tasks the goal is to predict quantitative outputs, whereas classification tasks are concerned with predicting qualitative outputs [15, 29]. Oftentimes, the dimension of the original vector space is not sufficient in order to perform complicated data analysis in the sense that data is, for example, not linearly separable in a classification task. A mapping can be constructed that maps the input space to a higher dimensional space [2]. Kernel machines provide a way of mapping the original space $\mathcal{X}$ into a reproducing kernel Hilbert space $\mathcal{H}$. In order to distinguish easily between tensors, matrices, vectors and scalars in subsequent chapters, the nonlinear feature map will be noted by the symbol $\mathbf{z}(\cdot)$. By using a right feature map $\mathbf{z}(\cdot) : \mathcal{X} \to \mathcal{H}$, the similarities in the mapped data can be analyzed as the original nonlinear problem in the original sample space $\mathcal{X}$ has become linear in $\mathcal{H}$.

3-1-1 Supervised learning

The primary objective in supervised learning is to learn a model that can accurately predict the correct output for new, unseen inputs. The supervised learning algorithm adjusts its internal parameters based on the input-output pairs in the training dataset. The training dataset is given by a matrix of inputs $\mathbf{X} \in \mathbb{R}^{N \times D}$, where N are the number of samples and D

are the number of features in the training dataset. The rows the matrix $\mathbf{X}$ are the input vectors $\mathbf{x}_n$, $n = 1, \ldots, N$. Each input vector is accompanied by an output y_n, also called observation, where all sample outputs are given by $\mathbf{y} = [y_1 \ y_2 \ \cdots \ y_N]^T$. The input/output pairs are assumed to be mutually independent and identically distributed samples of datapoints which form the training dataset. Thereby, the optimization problem in supervised learning that needs to be minimized directly with respect to the parameters is given by [12]:

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}\left(y_n, f(\mathbf{x}_n, \mathbf{w})\right) \tag{3-1}$$

where $f(\cdot, \cdot) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is a nonlinear function parameterized by weights $\mathbf{w}$ that needs to be learned and $\mathcal{L}(\cdot, \cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ is a loss function. For simplicity, the parameter vector $\mathbf{w}$ will be omitted in the notation for $f(\mathbf{x}_n, \mathbf{w})$ in this section of the survey such that the notation becomes $f(\mathbf{x}_n)$.

3-1-2 Kernel machine model

The primal formulation of a kernel machine, given by a direct parameterization between parameters $\mathbf{w}$ and feature mapping $\mathbf{z}(\mathbf{x})$, will we be the main focus throughout this research. This model is assumed to have Gaussian noise that is added to the deterministic function $f(\mathbf{x}_n, \mathbf{w})$. The following model describes a kernel machine:

$$y_n = f(\mathbf{x}_n) + e_n = \mathbf{z}(\mathbf{x}_n)^T \mathbf{w} + e_n \qquad n = 1, \ldots, N. \tag{3-2}$$

In this formulation, $\mathbf{z}(\cdot) : \mathbb{R}^D \to \mathcal{H}$ is the feature map and, unless otherwise stated, it can be assumed that e_n is a Gaussian random variable with variance σ_e^2. Furthermore, the $\mathbf{z}(\mathbf{x}_n)^T\mathbf{w}$ term can also be conveniently written as inner product as $\langle \mathbf{z}(\mathbf{x}_n), \mathbf{w} \rangle$. When the feature mapping is comprised of non-linear functions then $f(\mathbf{x}_n)$ is a non-linear function of the input data vector $\mathbf{x}$, however, $f(\mathbf{x}_n)$ is linear in the parameters $\mathbf{w}$. The kernel machine model is visualized by means of a tensor network diagram in Figure 3-1.

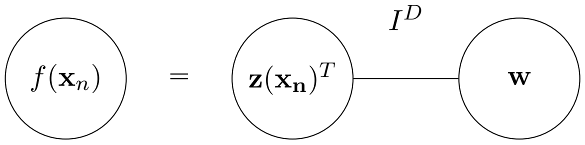


Figure 3-1: Graphical representation of the deterministic part of the kernel machine model

3-1-3 Product feature mapping

There are many ways in which the feature map $\mathbf{z}(\cdot)$ can be constructed. Consider a input matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ where the rows are the input vectors $\mathbf{x}_n$, $n = 1, \ldots, N$. In this way $x_n^{(d)}$

is the (n, d)-th element of input matrix $\mathbf{X}$. A way of constructing the feature map, is by obtaining the multidimensional feature map from the local ones as [49, 55]:

$$\mathbf{z}(\mathbf{x}_n) = \bigotimes_{d=1}^{D} \mathbf{z}_d(x_n^{(d)}) \tag{3-3}$$

where $\bigotimes_{d=1}^{D}$ is a sequence of tensor products which are Kronecker products when considering only vectors, $\mathbf{z}_d$ is the local feature map of the d-th dimension. Thus, every input element $x_n^{(d)}$ is mapped to a $\hat{M}_d$-dimensional vector. It is assumed that the same local map is applied to each element $x^{(d)}$. The dimension of the feature space is $M = \prod_{d=1}^{D} \hat{M}_d$, where $\hat{M}_1 = \cdots \hat{M}_D = \hat{M}$, and thereby $M = \hat{M}^D$. In this formulation of a multidimensional feature map, the feature map $\mathbf{z}(\cdot)$ can be viewed as a $M = \hat{M}^D$ dimensional vector or as a D-th order tensor, where N is the local dimension of $\mathbf{z}_d$. The important property of the constructed tensor is that the tensor is rank-1 as it is a product of D order-1 tensors. The feature map in equation (3-3), which is a sequence of tensor products of vectors $\mathbf{z}_d(x_n^{(d)})$, is the same as a sequence of Kronecker products since we are dealing with vectors only.

It is assumed that the same local map is applied to each element $x^{(d)}$. Hence, since $\mathbf{z}_d$ is the local feature map of the d-th dimension $\mathbf{z}_d(x_n^{(d)}) = \mathbf{z}(x_n^{(d)})$. Considering all samples and employing the product feature map, the model can then be written as:

$$\mathbf{y} = \mathbf{Z}\mathbf{w} + \mathbf{e}, \quad \text{where} \quad \mathbf{Z} = \begin{bmatrix} \mathbf{z}(x_1^{(1)})^T \otimes \cdots \otimes \mathbf{z}(x_1^{(d)})^T \otimes \cdots \otimes \mathbf{z}(x_1^{(D)})^T \\ \vdots \\ \mathbf{z}(x_n^{(1)})^T \otimes \cdots \otimes \mathbf{z}(x_n^{(d)})^T \otimes \cdots \otimes \mathbf{z}(x_n^{(D)})^T \\ \vdots \\ \mathbf{z}(x_N^{(1)})^T \otimes \cdots \otimes \mathbf{z}(x_N^{(d)})^T \otimes \cdots \otimes \mathbf{z}(x_N^{(D)})^T \end{bmatrix}. \tag{3-4}$$

Here, $\mathbf{e}$ is a vector of Gaussian random variables and $\mathbf{Z}$ is an $N \times M$ matrix, where M is the dimension of the feature space.

3-1-4 Pure power polynomial feature map

Throughout this research, the main type of feature that is employed in the product feature map are pure power polynomial features [4]. A particular feature can capture important patterns, relationships or characteristics in the data that can be highly informative and discriminative for the type of problem at hand. The pure power polynomial product feature map $\mathbf{z}(\cdot) : \mathbb{R}^D \rightarrow \mathbb{R}^{\hat{M}^D}$ for an input sample $\mathbf{x}_n \in \mathbb{R}_D$ is defined as in equation (3-3), where the mapping $\mathbf{z}_d(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^{\hat{M}}$ is defined as

$$\mathbf{z}_d(x_n^{(d)}) = \left[1, x_n^{(d)}, (x_n^{(d)})^2, (x_n^{(d)})^{\hat{M}-1}\right]. \tag{3-5}$$

Hence, the m_d-th element of the feature map vector $\mathbf{z}_d(x_n^{(d)})$ is defined as

$$\mathbf{z}_d(x_n^{(d)})_{m_d} = (x_n^{(d)})^{m_d}, \quad m_d = 0, 1, \ldots, \hat{M} - 1. \tag{3-6}$$

The feature map vector has length $\hat{M}^D = M$, since equal degree of feature map per dimension is considered. Pure power polynomial features contain more higher-order terms than the more commonly used affine polynomials. Moreover, it is guaranteed that any continuous function on a locally compact domain can be approximated arbitrarily well by polynomials of increasing degree [8, 56].

3-1-5 Quantized pure power polynomial feature map

In this research, uncertainty intervals will be visualized in a regression case for 1-dimensional data. However, the CPD is particularly useful when dealing with multi-dimensional data where each dimension captures different aspects or features of the data. CPD excels in capturing interactions and patterns among multiple dimensions, which are absent in one-dimensional data. While it can be applied to one-dimensional data, doing so does not yield meaningful results, and in many cases, it can be considered useless. In order to tackle this problem and still be able to provide both meaningful uncertainty quantification results and visualizations for 1-D data problems, the 1-D data is lifted to higher order data by utilizing quantized features [56]. For one-dimensional problems, this research assumes that $\hat{M}$ can be written as a power of 2 as such: $\hat{M} = 2^K$. Then, the following quantized vector can be defined:

$$\mathbf{s}^{(d,k)}(x_d) = \left[1, x_d^{2^{k-1}}\right]. \tag{3-7}$$

The quantized pure power $(\hat{M} - 1)$ polynomial feature map can then be stated as:

$$\mathbf{z}_d(x_n^{(d)}) = \bigotimes_{k=1}^{K} \mathbf{s}^{(d,k)}(x_d). \tag{3-8}$$

An example can be given as follows. Choosing $\hat{M}$ to be a power of 2 such as $\hat{M} = 8$, this results in $K = \log_2(8) = 3$. Therefore, the pure power quantized polynomical feature map for 1-D data is constructed from:

$$\mathbf{z}_d(x_n^{(d)}) = [1, x] \otimes [1, x^2] \otimes [1, x^3]. \tag{3-9}$$

3-1-6 Optimization problem

The deterministic part of the kernel machine in equation (3-2) is formed by an inner product between the parameters $\mathbf{w}$ and the inputs mapped into the feature space $\mathbf{z}(\mathbf{x})$. The goal is to learn the model parameters such that the deterministic part of the kernel machine optimally predicts the underlying function that governs the observations y_n. The optimization problem for the primal problem of kernel machines can be stated as [55]:

$$\min_{\mathbf{w}} \quad \sum_{n=1}^{N} \mathcal{L}(y_n, \langle \mathbf{z}(\mathbf{x}_n), \mathbf{w} \rangle) + \lambda_{\text{reg}} \langle \mathbf{w}, \mathbf{w} \rangle_p \tag{3-10}$$

where $\langle \mathbf{w}, \mathbf{w} \rangle_p$ is the p-th norm regularization term and $\lambda_{\text{reg}} \in \mathbb{R}_+$ is a scaling hyperparameter. The regularization term is included in the objective function in order to prevent overfitting of the data. An overfitted model has relatively low error on the training data but is less able to perform well on unseen data. In this way, the regularization term functions as a penalty term that usually limits the complexity of the models to avoid overfitting [52]. In this research, the regularization term is usually denoted by $R(\mathbf{w}) : \mathbb{R}^D \to \mathbb{R}_+$ which includes the scaling parameter. Throughout this research, a squared loss function and Tikhonov regularization are employed. This form of the optimization problem stated in equation (3-10) boils down to Kernel Ridge Regression (KKR). In this way, the KKR optimization problem can be stated as:

$$\min_{\mathbf{w}} \quad \sum_{n=1}^{N} (y_n, \langle \mathbf{z}(\mathbf{x}_n), \mathbf{w} \rangle)^2 + \lambda_{\text{reg}} ||\mathbf{w}||_2^2. \tag{3-11}$$

The KKR optimization problem can be used to solve both regression and classification tasks. In regression tasks the observation y_n represents a continuous variable, whereas in a binary classification task the goal is to predict the right discrete label, hence, $y_n \in \{-1, 1\}$.

The KKR optimization problem is a linear least squares problem. Such type of problems are convex quadratic programming problems which have a global optimal solution. By setting the gradient of the KKR optimization problem to zero and solving with respect to the parameters $\mathbf{w}$, the closed-form solution is obtained, which has the following form:

$$\mathbf{w}_{LS} = (\lambda_{\text{reg}}\mathbf{I} + \mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{y}. \tag{3-12}$$

A major drawback of kernel machines is their lack of scalability. More specifically, making predictions for the KKR problem requires construction and inversion of $\lambda_{\text{reg}}\mathbf{I} + \mathbf{Z}^T\mathbf{Z}$ which amounts to $NM^2 + M^3$ operations. Therefore, solving the kernel machine least squares problem renders a computational complexity of $\mathcal{O}(M^3)$ when $N \ll M$, where M is the size of the parameter vector $\mathbf{w}$, and a computational complexity of $\mathcal{O}(NM^2)$ when $N \gg M$.

3-2 Supervised learning with tensor networks

3-2-1 Curse of dimensionality

The curse of dimensionality refers to an exponentially increasing number of parameters that is required to describe data or a system in a machine learning context. By mapping an input space to a high-dimensional space, the volume associated to the dimensions of the input space increases rapidly. In general, this causes machine learning models to inhibit greater error and the learning algorithms to have increased running times. In the context of tensors, the curse of dimensionality refers to the event in which the number of elements I^D of a D-th order tensor of size $I \times I \times \cdots \times I$ grows exponentially with the order of the tensor which is D. The number of elements can become enormous when considering multi-way arrays that possess a great number of dimensions. In order to perform computations, and to store such data, excessive computational and storage resources are required [5].

The optimization problem described in equation (3-11) suffers heavily from the curse of dimensionality when employing high-dimensional data. This is the case as the product feature map described in equation (3-13) is employed. In the KKR problem with product feature map, the parameter tensor $\boldsymbol{\mathcal{W}}$ is a D-th order tensor with $M = \hat{M}^D$ elements. Thereby, the complexity of the optimization problem scales exponentially with respect to the number of features D. In this regard, when $M \ll N$, it is more advantageous to use the primal formulation compared to the dual formulation. The dual formulation is obtained by using the kernel trick [20], which scales burdensome in the number of samples as the computational and storage complexity is $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ respectively. Throughout this research, the primal problem is examined. To conclude, the optimization problem in equation (3-10) becomes computationally expensive when the dataset contains a large number of features.

3-2-2 Tensor network constrained optimization problem

Constraining the parameters within an optimization problem to be a low-rank tensor network can alleviate the curse of dimensionality. A tensorized representation of the weight vector and the feature map needs to be stated first in order to employ a tensor network on the parameters. As can be inferred from equation (2-26), an outer product of vectors can represent a tensor. The feature map in equation (3-3) is written in terms of a sequence of Kronecker products. These Kronecker products can be interchanged with with outer products allowing for a tensorized representation of the feature map. This is shown by the relation between Kronecker products and outer products of vectors in equation (2-25). Thereby, tensorization of $\mathbf{z}(x_n)$ is achieved by considering outer products. Hence,

$$\boldsymbol{\mathcal{Z}}(\mathbf{x}_n) = \mathbf{z}_1(x_n^{(1)}) \circ \mathbf{z}_2(x_n^{(2)}) \circ \cdots \circ \mathbf{z}_D(x_n^{(D)}). \tag{3-13}$$

where $\boldsymbol{\mathcal{Z}}(\cdot) = \mathbb{R}^D \rightarrow \mathbb{R}^{\hat{M}_1} \otimes \mathbb{R}^{\hat{M}_2} \otimes \cdots \otimes \mathbb{R}^{\hat{M}_D}$ is the tensorized multidimensional feature mapping [48, 55, 33]. The deterministic part of the kernel machine model can be represented in tensorized format due to the formulation of the feature map in 3-13 and tensorization of the parameter vector $\mathbf{w}$. Thereby, this tensorized formulation has the following form:

$$y_n = f(\mathbf{x}_n) + e_n = \langle \boldsymbol{\mathcal{Z}}(\mathbf{x}_n), \boldsymbol{\mathcal{W}} \rangle + e_n \qquad n = 1, \ldots, N. \tag{3-14}$$

Since $\boldsymbol{\mathcal{Z}}(\mathbf{x}_n)$ is composed of outer products, the tensor network diagram for the deterministic part of the kernel machine is visualized in Figure 3-2.

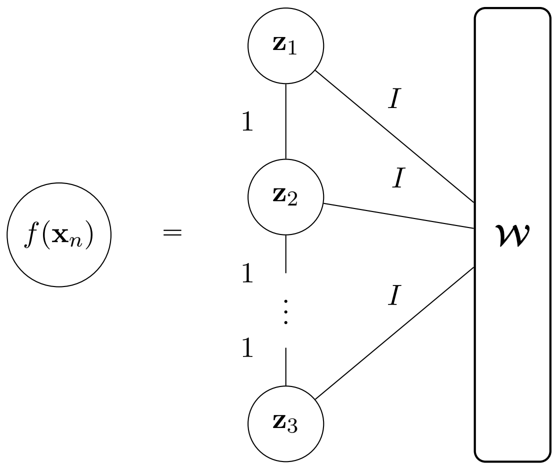


Figure 3-2: Graphical representation of deterministic part of the kernel machine model with product feature mapping

The curse of dimensionality, which is inherent to the kernel machine model in equation (3-14), can be alleviated by constraining the parameter tensor $\mathcal{W}$ to be a low-rank tensor network. More specifically, this research examines the CP decomposition of the weights. The deterministic part of the kernel machine model is visualized in the tensor network diagram in Figure 3-3, where the parameter tensor $\mathcal{W}$ is constrained to be a Canonical Polyadic Decomposition (CPD).

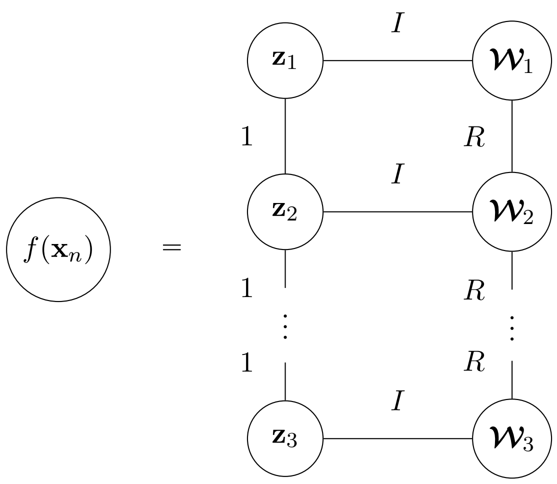


Figure 3-3: Graphical representation of the tensor based kernel machine

Ultimately, by incorporating a squared loss function, Tikhonov regularization, a product feature map and constraining the parameters to be a CPD, the following optimization problem can be stated:

$$\begin{aligned}\min_{\mathcal{W}} \quad & V_N = \sum_{n=1}^{N} \left(y_n - \langle \mathcal{Z}(\mathbf{x}_n), \mathcal{W} \rangle\right)^2 + \lambda_{\text{reg}} \langle \mathcal{W}, \mathcal{W} \rangle \\ \text{subject to} \quad & \text{CP-rank}\,(\mathcal{W}) = R.\end{aligned} \tag{3-15}$$

This optimization can also be rewritten such that the constraint is removed. This is realized by employing the Frobenius property that is described in equation (2-30). In addition, by employing the Frobenius inner product of the CPD with a rank-1 tensor described in equation (2-31), the optimization problem of equation (3-15) is equivalent to [53]:

$$\begin{aligned}\min_{\mathcal{W}} \quad V_N &= \min_{\mathcal{W}} \frac{1}{N} \sum_{n=1}^{N} \left(y_n - \left(\mathbf{z}(x_n^{(1)})^T \mathbf{W}^{(1)} * \mathbf{z}(x_n^{(2)})^T \mathbf{W}^{(2)} * \cdots \mathbf{z}(x_n^{(D)})^T \mathbf{W}^{(D)} \right) \boldsymbol{\lambda} \right)^2 \\ &+ \lambda_{\text{reg}} \boldsymbol{\lambda} \left(\mathbf{W}^{(1)^T} \mathbf{W}^{(1)} * \mathbf{W}^{(2)^T} \mathbf{W}^{(2)} * \cdots * \mathbf{W}^{(D)^T} \mathbf{W}^{(D)} \right) \boldsymbol{\lambda} \\ &= \min_{\mathcal{W}} \frac{1}{N} \sum_{n=1}^{N} \left(y_n - \left(\mathop{\circledast}_{d=1}^{D} \mathbf{z}(x_n^{(d)})^T \mathbf{W}^{(d)} \right) \boldsymbol{\lambda} \right)^2 + \lambda_{\text{reg}} \boldsymbol{\lambda}^T \left(\mathop{\circledast}_{d=1}^{D} \mathbf{W}^{(d)^T} \mathbf{W}^{(d)} \right) \boldsymbol{\lambda}.\end{aligned} \tag{3-16}$$

3-2-3 CPD-ALS

The optimization problem in equation (3-15) and (3-16) becomes non-convex as the parameters are now constrained to be a low-rank tensor network. More specifically, representing the parameters as a CPD, causes the kernel machine to no longer be linear in the parameters and the optimization problem becomes nonlinear and non-convex.

A frequently used learning algorithm for optimizing nonlinear problems when dealing with tensor networks is a block coordinate descent approach, called the Alternating Linear Scheme (ALS) algorithm [57]. The idea behind optimizing the nonlinear problem is to optimize a block of parameters while keeping the other parameters constant. In this way, the original nonlinear problem is transformed into multiple convex and analytically solvable sub-problems. This is achievable because of the multi-linear structure of the CPD. The blocks that are iteratively updated should be chosen such that the sub-problem is indeed convex and analytically solvable. Updating of the different blocks of parameters is performed by starting at the first block and sweeping to the last block where the algorithm will start to turn backwards again. The ALS algorithm is a monotonically decreasing algorithm, meaning that the objective function value consistently decreases with each iteration, that does not necessarily converges to the global optimum [18, 6].

In the case of the CPD constrained kernel machine of 3-15, the number of parameters is reduced from $\hat{M}^D$ to $D\hat{M}R$. In the Canonical Polyadic Decomposition Alternating Linear Scheme (CP-ALS) algorithm, the optimization problem is split into several sub-problems where each such problem is a linear least squares problem in which $\mathbf{W}^{(d)}$ is updated. Each linear least squares sub-problem can be solved by solving the normal equations [55]. The ALS sub-problem that needs to be solved for the optimization problem of equation (3-15) can be stated by first formulating the data-fitting term $\langle \mathcal{W}, \mathcal{Z}(\mathbf{x}) \rangle$ which can be rewritten linearly in terms of the unknown factor matrix as:

$$\langle \boldsymbol{\mathcal{W}}, \boldsymbol{\mathcal{Z}}(\mathbf{x})\rangle = \left\langle \operatorname{vec}\left(\mathbf{W}^{(d)}\right), \mathbf{z}(x_n^d) \otimes \left(\mathop{\circledast}_{p=1,p\neq d}^{D} \mathbf{z}(x_n^{(p)})^T \mathbf{W}^{(p)} \right) \right\rangle. \tag{3-17}$$

Likewise, the regularization term in equation (3-15) can be written linearly as follows:

$$\langle \boldsymbol{\mathcal{W}}, \boldsymbol{\mathcal{W}}\rangle = \left\langle \operatorname{vec}\left(\mathbf{W}^{(d)^T}\mathbf{W}^{(d)}\right), \operatorname{vec}\left(\mathop{\circledast}_{p=1,p\neq d}^{D} \mathbf{W}^{(p)^T} \mathbf{W}^{(p)} \right) \right\rangle. \tag{3-18}$$

In the new formulation, the CPD constraint is incorporated in both the data-fitting term and the regularization term. Hence, a linear least squares problem for vec $\left(\mathbf{W}^{(d)}\right)$ is obtained by substituting equation (3-17) and equation (3-18) into equation (3-15) [55]:

$$\begin{aligned} \min_{\operatorname{vec}\left(\mathbf{W}^{(d)}\right)} \quad V_N = \frac{1}{N}\sum_{n=1}^{N} \left(y_n - \left\langle \operatorname{vec}\left(\mathbf{W}^{(d)}\right), \mathbf{z}(x_n^d) \otimes \left(\mathop{\circledast}_{p=1,p\neq d}^{D} \mathbf{z}(x_n^{(p)})^T \mathbf{W}^{(p)} \right) \right\rangle \right)^2 \\ + \lambda_{\text{reg}} \left\langle \operatorname{vec}\left(\mathbf{W}^{(d)^T}\mathbf{W}^{(d)}\right), \operatorname{vec}\left(\mathop{\circledast}_{p=1,p\neq d}^{D} \mathbf{W}^{(p)^T} \mathbf{W}^{(p)} \right) \right\rangle. \end{aligned} \tag{3-19}$$

The CP-ALS sub-problem as formulated in equation (3-19) can be solved by the normal equations which requires $N\hat{M}^2R^2 + \hat{M}^3R^3$ operations. In the CP-ALS optimization algorithm, a non-convex optimization problem that consists of multiple convex and analytically solvable sub-problems is encountered. While the CP-ALS algorithm is monotonically decreasing, it does not provide a guarantee of converging to the global optimum solution. Thereby, the computational complexity of the algorithm is $ND\hat{M}^2R^2$ when $N \gg \hat{M}R$, therefore, the algorithm can be suitably used even for large N and D, given that R and $\hat{M}$ are small. With regards to the storage complexity, the required storage is $R\hat{M}D + 2R^2\hat{M}^2 + 2R\hat{M}$, which is attributed by the weight tensor in CPD format, the rank-$R\hat{M}$ Gram matrix, the regularization matrix and by the transformed responses and solution of the linear system. Hence, the storage complexity is $\mathcal{O}\left(R^2\hat{M}^2\right)$ when $R\hat{M} \gg D$, and thereby, the storage complexity is independent of the number of samples N [55].

Furthermore, the optimization problem in equation (3-19) can be solved by setting the derivative with respect to vec$(\mathbf{W}^{(d)})$ to zero since it is a linear least squares problem. In order to do this, expressions for the prediction model gradient and regularization term gradient are defined. Firstly, the prediction model can be defined as

$$f(\mathbf{x}_n, \mathbf{w}) = \left\langle \operatorname{vec}\left(\mathbf{W}^{(d)}\right), \mathbf{z}(x_n^{(d)}) \otimes \left(\mathop{\circledast}_{p=1,p\neq d}^{D} \mathbf{z}(x_n^{(p)})^T \mathbf{W}^{(p)} \right) \right\rangle. \tag{3-20}$$

The model is a linear combination of vec$(\mathbf{W}^{(d)})$ as it is written as an inner product of vec$(\mathbf{W}^{(d)})$. Thereby, the following result is obtained for the derivative of the prediction model with respect to vec$(\mathbf{W}^{(d)})$:

$$\frac{\partial f(\mathbf{x}_n, \mathbf{w})}{\partial \operatorname{vec}(\mathbf{W}^{(d)})} = \mathbf{z}(x_n^d) \otimes \left(\mathop{\circledast}_{p=1,p\neq d}^{D} \mathbf{z}(x_n^{(p)})^T \mathbf{W}^{(p)} \right). \tag{3-21}$$

The linear least squares problem is solved for all samples at once. By doing so, first the feature map for an entire input matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ is stated. This done by transforming each dimension of the input matrix $\mathbf{X}^{(d)} \in \mathbb{R}^{N}$ separately as follows:

$$\mathbf{Z}(\mathbf{X}^{(d)}) = [\mathbf{z}(x_1^{(d)})\ \mathbf{z}(x_2^{(d)}) \ldots \mathbf{z}(x_N^{(d)})] \in \mathbb{R}^{M \times N}. \tag{3-22}$$

Furthermore, the term $\mathbf{Z}(\mathbf{X}^{(d)})^T \mathbf{W}^{(d)}$ in equation (3-21) can also be written for all samples which is done as follows:

$$\mathbf{Z}(\mathbf{X}^{(d)})^T \mathbf{W}^{(d)} = \left[\mathbf{z}(x_1^{(d)})^T \mathbf{W}^{(d)}\ \mathbf{z}(x_2^{(d)})^T \mathbf{W}^{(d)}\ \ldots\ \mathbf{z}(x_N^{(d)})^T \mathbf{W}^{(d)} \right] \in \mathbb{R}^{N \times R}. \tag{3-23}$$

By considering all samples for the prediction model $\mathbf{f}(\mathbf{X}, \mathbf{w}) = [f(\mathbf{x}_1, \mathbf{w})\ f(\mathbf{x}_2, \mathbf{w})\ \ldots\ f(\mathbf{x}_N, \mathbf{w})]^T$ and the derivative for a single sample in equation (3-21), the derivative of the model for all samples with respect to $\text{vec}(\mathbf{W}^{(d)})$ is given by

$$\begin{aligned} \frac{\partial \mathbf{f}(\mathbf{X}, \mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(d)})} &= \left[\mathbf{z}(x_1^d) \otimes \left(\mathop{\circledast}_{p=1, p \neq d}^{D} \mathbf{z}(x_1^{(p)})^T \mathbf{W}^{(p)} \right)\ \mathbf{z}(x_2^d) \otimes \left(\mathop{\circledast}_{p=1, p \neq d}^{D} \mathbf{z}(x_2^{(p)})^T \mathbf{W}^{(p)} \right) \right. \\ &\quad \left. \ldots\ \mathbf{z}(x_N^d) \otimes \left(\mathop{\circledast}_{p=1, p \neq d}^{D} \mathbf{z}(x_N^{(p)})^T \mathbf{W}^{(p)} \right) \right]^T \\ &= \left(\mathbf{Z}(\mathbf{X}^{(d)}) \odot \left(\mathop{\circledast}_{p=1, p \neq d}^{D} \mathbf{Z}(\mathbf{X}^{(p)})^T \mathbf{W}^{(p)} \right)^T \right)^T. \end{aligned} \tag{3-24}$$

The prediction model $\mathbf{f}(\mathbf{X}, \mathbf{w})$ is an inner product of $\text{vec}(\mathbf{W}^{(d)})$ which can be inferred from the derivative of $\mathbf{f}(\mathbf{X}, \mathbf{w})$ with respect to $\text{vec}(\mathbf{W}^{(d)})$ in equation (3-24). Hence, the prediction model for all samples are defined as:

$$\mathbf{f}(\mathbf{X}, \mathbf{w}) = \left(\mathbf{Z}(\mathbf{X}^{(d)}) \odot \left(\mathop{\circledast}_{p=1, p \neq d}^{D} \mathbf{Z}(\mathbf{X}^{(p)})^T \mathbf{W}^{(p)} \right)^T \right)^T \text{vec}(\mathbf{W}^{(d)}). \tag{3-25}$$

In this formulation, the matrix $\mathbf{C}$ is defined as

$$\mathbf{C} = \left(\mathbf{Z}(\mathbf{X}^{(d)}) \odot \left(\mathop{\circledast}_{p=1, p \neq d}^{D} \mathbf{Z}(\mathbf{X}^{(p)})^T \mathbf{W}^{(p)} \right)^T \right)^T. \tag{3-26}$$

The derivative in equation (3-24) equals $\mathbf{C}$ but can also equal $\mathbf{C}^T$ depending on the convention of the numerator with respect to the denominator which is clarified in equation (2-13). Similarly, the regularization term $R(\mathbf{w})$ is rewritten as an inner product of $\text{vec}(\mathbf{W}^{(d)})$. The regularization term in the CP-ALS sub-problem of equation (3-19) can be stated by using the vectorization property of equation (2-10) as follows:

$$\begin{aligned} R(\mathbf{w}) &= \lambda_{\text{reg}} \left\langle \text{vec}\left(\mathbf{W}^{(d)^T}\mathbf{W}^{(d)}\right), \text{vec}\left(\underset{p=1,p\neq d}{\overset{D}{\circledast}} \mathbf{W}^{(p)^T}\mathbf{W}^{(p)}\right)\right\rangle \\ &= \lambda_{\text{reg}}\text{vec}\left(\mathbf{W}^{(d)^T}\mathbf{W}^{(d)}\right)^T \text{vec}\left(\underset{p=1,p\neq d}{\overset{D}{\circledast}} \mathbf{W}^{(p)^T}\mathbf{W}^{(p)}\right) \\ &= \lambda_{\text{reg}}\text{Tr}\left(\mathbf{W}^{(d)^T}\mathbf{W}^{(d)} \underset{p=1,p\neq d}{\overset{D}{\circledast}} \mathbf{W}^{(p)^T}\mathbf{W}^{(p)}\right). \end{aligned} \tag{3-27}$$

The following result is obtained by applying the derivative rule stated in equation (2-12) to the gradient of the regularization term:

$$\begin{aligned} \frac{\partial R(\mathbf{w})}{\partial \mathbf{W}^{(d)}} &= \left(\mathbf{W}^{(d)} \underset{p=1,p\neq d}{\overset{D}{\circledast}} \mathbf{W}^{(p)^T}\mathbf{W}^{(p)}\right) + \left(\mathbf{W}^{(d)} \underset{p=1,p\neq d}{\overset{D}{\circledast}} \mathbf{W}^{(p)^T}\mathbf{W}^{(p)}\right) \\ &= 2\lambda_{\text{reg}}\mathbf{W}^{(d)}\left(\underset{p=1,p\neq d}{\overset{D}{\circledast}} \mathbf{W}^{(p)^T}\mathbf{W}^{(p)}\right). \end{aligned} \tag{3-28}$$

The derivative of the regularization term with respect to $\text{vec}(\mathbf{W}^{(d)})$ becomes:

$$\begin{aligned} \frac{\partial R(\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(d)})} &= 2\lambda_{\text{reg}}\text{vec}(\mathbf{W}^{(d)} \underset{p=1,p\neq d}{\overset{D}{\circledast}} \mathbf{W}^{(p)^T}\mathbf{W}^{(p)}) \\ &= 2\lambda_{\text{reg}}((\underset{p=1,p\neq d}{\overset{D}{\circledast}} \mathbf{W}^{(p)^T}\mathbf{W}^{(p)}) \otimes \mathbf{I}_M)\text{vec}(\mathbf{W}^{(d)}) \end{aligned} \tag{3-29}$$

where the last equality holds by equation (2-11) and is written in order to write the expresion linear in $\text{vec}(\mathbf{W}^{(d)})$. Considering the ALS sub-problem in equation (3-19), the goal is to set the derivative equal to zero considering all samples in the optimization problem in order to solve for the parameters in the linear least squares problem. The cost function in the ALS sub-problem in equation (3-19) considering all samples can be defined as:

$$V_N = \sum_{n=1}^{N}(y_n - f(\mathbf{x}_n, \mathbf{w}))^2 + R(\mathbf{w}) = ||\mathbf{y} - \mathbf{f}||_2^2 + R(\mathbf{w}). \tag{3-30}$$

Setting the partial derivative of this cost function with respect to $\text{vec}(\mathbf{W}^{(d)})$ to zero can be done by using equation (2-14) and plugging in the two expressions given in equation (3-24) and (3-28):

$$\begin{aligned} \frac{\partial V_N}{\partial \text{vec}(\mathbf{W}^{(d)})} &= -2\frac{\partial \mathbf{f}}{\partial \text{vec}(\mathbf{W}^{(d)})}(\mathbf{y} - \mathbf{f}) + \frac{\partial R(\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(d)})} \\ &= -\frac{2}{N}\mathbf{C}^T(\mathbf{y} - \mathbf{C}\text{vec}(\mathbf{W}^{(d)})) + 2\lambda_{\text{reg}}((\underset{p=1,p\neq d}{\overset{D}{\circledast}} \mathbf{W}^{(p)^T}\mathbf{W}^{(p)}) \otimes \mathbf{I}_M)\text{vec}(\mathbf{W}^{(d)}). \end{aligned} \tag{3-31}$$

Solving this linear system with respect to $\text{vec}(\mathbf{W}^{(d)})$ amounts to solving the following linear system:

$$(\mathbf{C}^T\mathbf{C} + \lambda_{\text{reg}} N((\mathop{\circledast}_{p=1,p\neq d}^{D} \mathbf{W}^{(p)^T}\mathbf{W}^{(p)}) \otimes \mathbf{I}_M))\text{vec}(\mathbf{W}^{(d)}) = \mathbf{C}^T\mathbf{y}. \tag{3-32}$$

Ultimately, making predictions can be inferred from the cost function in equation (3-16). With the obtained factor matrices from the CP-ALS algorithm, predictions by the estimated model can be produced as:

$$f(\mathbf{x}_{*n}, \hat{\mathbf{w}}) = \langle \boldsymbol{\mathcal{Z}}(\mathbf{x}_n), \boldsymbol{\mathcal{W}} \rangle = \left(\mathop{\circledast}_{d=1}^{D} \mathbf{z}(x_n^{(d)})^T \mathbf{W}^{(d)} \right) \boldsymbol{\lambda} \tag{3-33}$$

where $\mathbf{x}_{*n}$ denotes a test input. The CP-ALS algorithm is a monotonically decreasing learning algorithm that has a linear complexity in both sample size and dimensionality. A single sweep in the algorithm updates the factor matrices in the order $1 \rightarrow D$ and back from $D \rightarrow 1$ [55]. The CP-ALS algorithm is stated in the following algorithm:

Algorithm 1 MATLAB Function: `CP-ALS[55]`

```
1:  procedure CP ALS($\mathbf{X}, \mathbf{y}, M, R, \lambda_{\text{reg}}$, number of sweeps)
2:      $D \leftarrow size(\mathbf{X}, 2)$                                  ▷ Extract number of features
3:
4:      $\mathbf{ZW} \leftarrow 1$                                           ▷ Initialization of algorithm
5:      $\boldsymbol{\Gamma} \leftarrow 1$
6:      for $d \leftarrow D$ to 1 do
7:          $\mathbf{W}^{(d)} \leftarrow randn(M, R)$
8:          $\mathbf{W}^{(d)} \leftarrow \mathbf{W}^{(d)}/||\mathbf{W}^{(d)}||$
9:          $\mathbf{ZW} \leftarrow \mathbf{ZW} * \mathbf{Z}(\mathbf{X}^{(d)})^T\mathbf{W}^{(d)}$
10:         $\boldsymbol{\Gamma} \leftarrow \mathbf{W}^{(d)^T} * \mathbf{W}^{(d)}$
11:     end for
12:
13:     for $i \leftarrow 1$ to number of sweeps do                    ▷ Sweeping over the factor matrices
14:         for $d \leftarrow 1$ to $D$ do
15:             $\mathbf{ZW_d} \leftarrow \mathbf{ZW}/(\mathbf{Z}(\mathbf{X}^{(d)})^T\mathbf{W}^{(d)})$
16:             $\boldsymbol{\Gamma}_\mathbf{d} \leftarrow \boldsymbol{\Gamma}/(\mathbf{W}^{(d)^T}\mathbf{W}^{(d)})$
17:             $\mathbf{C} \leftarrow \left(\mathbf{Z}(\mathbf{X}^{(d)}) \odot \left(\circledast_{p=1,p\neq d}^{D} \mathbf{Z}(\mathbf{X}^{(p)})^T\mathbf{W}^{(p)}\right)^T\right)^T$
18:             $\boldsymbol{\Lambda}_{\text{reg}} \leftarrow \lambda_{\text{reg}}(\mathbf{I}_M \otimes \boldsymbol{\Gamma}_d)$
19:             $\mathbf{A} \leftarrow \mathbf{C}^T\mathbf{C} + \boldsymbol{\Lambda}_{\text{reg}}$
20:             $\mathbf{b} \leftarrow \mathbf{C}^T\mathbf{y}$
21:             $\mathbf{w} \leftarrow solve(\mathbf{A}, \mathbf{b})$
22:             $\mathbf{W} \leftarrow reshape(\mathbf{w}, (M, R)))$
23:             $\mathbf{ZW} \leftarrow \mathbf{ZW_d} * (\mathbf{Z}(\mathbf{X}^{(d)})^T\mathbf{W})$
24:             $\boldsymbol{\Gamma} \leftarrow \boldsymbol{\Gamma}_\mathbf{d}(\mathbf{W}^T, \mathbf{W})$
25:             $\mathbf{W}^{(d)} \leftarrow \mathbf{W}$
26:         end for
27:     end for
28:     return $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \ldots, \mathbf{W}^{(D)}$            ▷ Return the factor matrices
29: end procedure
```

Chapter 4

Uncertainty quantification: A frequentist approach

By minimizing the cost function of the constrained Canonical Polyadic Decomposition (CPD) kernel machine, model parameters can be learned and, in turn, predictions can be made. However, those predictions are not provided with any quantification on how certain the predictions are. This is a downside as it becomes increasingly important to evaluate the reliability, predictability and efficacy of machine learning models before the models are directly applied in practice [47, 1]. This is because machine learning models are increasingly adopted in different fields, from investment opportunities and medical diagnoses to sports games and weather forecasting [1].

4-1 Prediction uncertainty

4-1-1 Sources of uncertainty

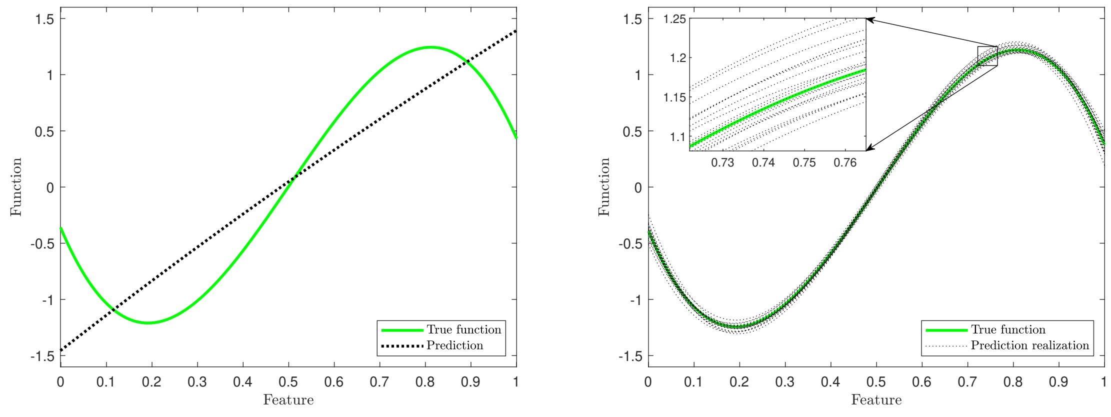
Before providing the details concerning the delta method, it is important to precisely set out the sources that contribute to the uncertainty in the predictions. The prediction error in equation (4-1) depends on two main types of uncertainties: epistemic uncertainty and aleatory uncertainty. Epistemic uncertainty is concerned with the lack of knowledge about a particular environment, process or system. In contrast, aleatory uncertainty concerns inherent variability in a particular system. This kind of stochasticity is especially present in human and natural systems [26, 54]. This research focuses on the prediction error variance. The prediction error can be stated as follows:

$$\epsilon_n(\mathbf{x}_n, \hat{\mathbf{w}}) = y_n - f(\mathbf{x}_n, \hat{\mathbf{w}}) = e_n + f^*(\mathbf{x}_n) - f(\mathbf{x}_n, \hat{\mathbf{w}}) \tag{4-1}$$

where $f^*(\mathbf{x}_n)$ is the prediction of the *true system* for sample n, $f(\mathbf{x}_n, \hat{\mathbf{w}})$ is the prediction with a model for parameter values $\hat{\mathbf{w}}$ and e_n is an additive, independent and identically distributed random noise variable attributed by an observation. The true function, or system, can also be stated for all samples as $f^* = [f^*(\mathbf{x}_1)\ f^*(\mathbf{x}_2)\ \dots\ f^*(\mathbf{x}_N)]^T = [f_1^*\ f_2^*\ \dots\ f_N^*]^T$. The prediction error in equation (4-1) depends on the uncertainty in the training data (aleatory uncertainty) which is attributed by e_n. On the other hand, the prediction error depends on the uncertainty in the model (epistemic uncertainty) which is given by the model error $f^*(\mathbf{x}_n) - f(\mathbf{x}_n, \hat{\mathbf{w}})$. It is important to note that there will always exist a prediction error no matter how close the prediction $f(\mathbf{x}_n, \hat{\mathbf{w}})$ is to the prediction of the true system $f^*(\mathbf{x}_n)$. This is because, even if the true system is known, there will always be aleatory uncertainty coming from the independent error term e_n.

In order to further decompose the prediction error, a distinction should be made between two different sources of error which together form the model error $f^*(\mathbf{x}_n) - f(\mathbf{x}_n, \hat{\mathbf{w}})$. *Structural model errors* are the type of errors that come from deficiencies in the model structure. These type of errors are caused by inherent incapability of producing correct model outputs. As such, even with perfect and infinite estimation data, the model prediction deviates from the true system [32, 31, 44]. *Estimation errors* refer to model errors that are caused by the unpredictable fluctuations or variations in the observed data used for model estimation [32, 31, 44]. These errors can arise from various sources, such as measurement errors, inherent variability, or unaccounted factors. It should be emphasized that estimation errors are epistemic sources of uncertainty whereas the observation noise is aleatory uncertainty. This is because the observation noise is inherent to the data and can not be accounted for, whereas estimation errors are due to the lack of acquiring the true model parameters, which in principal can be accounted for. For example, increased sample size, more accurate parameter estimation techniques, or improved model calibration reduce parameter estimation errors.
Despite having a well-structured model without any structural model errors, the model predictions will still contain errors due to the presence of these estimation errors. In other words, these errors are not a result of the model's structural flaws, such as choosing a too simplistic model structure which is unable to explain the data, but rather arise from the inherent uncertainty and noise in the data.

An illustration of structural model error and estimation error is provided in Figure 4-1. In Figure 4-1a, a polynomial of degree 2 is fitted to data with an underlying true function. It can be observed that a polynomial of degree 2 is inherently unable to explain the data in the sense that the underlying true function is not well approximated. In Figure 4-1b, the same kind of model is used to fit the data as the model that is used to generate the data. There is approximately no structural model error, however, different realizations of the prediction still deviate from each other. This can be ascribed to estimation errors which are due to the use of different training and test data, different noise realizations on the data and different parameter initialization.



(a) Structural model error **(b)** Estimation error

Figure 4-1: Illustration of estimation error and structural model error

Whether the prediction error in equation (4-1) contains structural model errors depends on whether the true system f^* is part of the model structure $\mathcal{M}$. The model set can first be defined as

Definition 4.1 (Model set [27, 31])**.** A model set $\mathcal{M}^*$ is a collection of models where each model predicts the current output of a system given information about inputs.

Most often a pertinent model set is non-countable as it contains an infinite number of models. Thereby, a search for the best model within the model set is done over an area where the model is differentiable with respect to the parameters and that the parameters live within a open set. This is done by restricting the model class to contain only parametric models $f(\mathbf{x}, \mathbf{w})$ with a finite number of parameters. This research only considers parametric models with well defined gradients of the model with respect to the parameters. The model structure can be defined as

Definition 4.2 (Model structure [27, 31])**.** A model structure $\mathcal{M}$ is a differentiable mapping from a connected open subset $D_{\mathcal{M}}$ of $\mathbb{R}^d$ to a model set $\mathcal{M}^*$, such that the gradient of the predictor set are smooth. Thereby

$$\mathcal{M} : \mathbf{w} \in D_{\mathcal{M}} \rightarrow \mathcal{M}(\mathbf{w}) = f(\mathbf{x}, \mathbf{w}) \in \mathcal{M}^*. \tag{4-2}$$

Furthermore, $\mathbf{w}^0$ is defined as the parameter vector for the true system and $\mathbf{w}^*$ is the asymptotic value of the parameter estimate. In other words, $\mathbf{w}^*$ is the parameter vector that asymptotically minimizes a cost function. Two cases can be distinguished when the prediction error in equation (4-1) is further decomposed.

1. *The true system is in the model structure* ($f^* \in \mathcal{M}$). There exists a $\mathbf{w}^0$ such that $f^*(\mathbf{x}_n) = f(\mathbf{x}_n, \mathbf{w}^0)$ for all samples. In this case, $\hat{\mathbf{w}} \rightarrow \mathbf{w}^0$ as $N \rightarrow \infty$. Thereby, the model parameter error $\mathbf{w}^0 - \hat{\mathbf{w}}$ will be an estimation error caused by data disturbances. Hence, the prediction error of equation (4-1) can be stated as:

$$\epsilon_n(\mathbf{x}_n, \hat{\mathbf{w}}) = e_n + \underbrace{f(\mathbf{x}_n, \mathbf{w}^0) - f(\mathbf{x}_n, \hat{\mathbf{w}})}_{\text{estimation error}}. \tag{4-3}$$

2. *The true system is not in the model structure* ($f^* \notin \mathcal{M}$). In this case the asymptotic value of the parameter estimate can be defined as $\hat{\mathbf{w}} \to \mathbf{w}^*$ as $N \to \infty$. Now, the prediction error of equation (4-1) can be stated as:

$$\epsilon_n(\mathbf{x}_n, \hat{\mathbf{w}}) = e_n + \underbrace{f^*(\mathbf{x}_n) - f(\mathbf{x}_n, \mathbf{w}^*)}_{\text{structural model error}} + \underbrace{f(\mathbf{x}_n, \mathbf{w}^*) - f(\mathbf{x}_n, \hat{\mathbf{w}})}_{\text{estimation error}}. \tag{4-4}$$

Moreover, the parameter estimation error be further decomposed by considering the different sources of error of parameter estimates:

$$\hat{\mathbf{w}} = \mathbf{w}^* + \underbrace{\hat{\mathbf{w}} - \mathbf{w}^* - \mathbf{w}^b}_{\text{Stochastic error}} + \underbrace{\mathbf{w}^b}_{\text{Bias error}}. \tag{4-5}$$

Here, $\mathbf{w}^b$ constitutes bias in the parameters. Therefore, the parameter stochastic error and the parameter bias error constitute errors in the estimated parameters.

4-1-2 Parameter covariance for linear least squares

The parameter covariance can be propagated through the variance of the model output when the prediction error is sufficiently small. Therefore, an expression for the parameter covariance needs to be obtained. Before obtaining an analytical expression for the parameter covariance for the nonlinear CPD constrained kernel machine, it is interesting to obtain an expression for a model that is linear in the parameters first. This case can clearly show why it is important to assume that the true function is part of the model structure $f^* \in \mathcal{M}$. Consider a model that is linear in the parameters and the following least squares optimization problem needs to be solved [32, 43, 30]:

$$V_N(\hat{\mathbf{w}}) = \sum_{n=1}^{N} \frac{1}{2}\epsilon_n^2(\mathbf{x}_n, \hat{\mathbf{w}}). \tag{4-6}$$

The parameters $\mathbf{w}^* = \hat{\mathbf{w}} + \Delta\mathbf{w}$ are defined as the parameter vector that asymptotically minimizes the cost function in equation (4-6) and $\Delta\mathbf{w}$ must be small. A quadratic approximation of the cost function around the estimated parameters $\hat{\mathbf{w}}$ is given by the second order Taylor expansion:

$$V_N(\mathbf{w}*) = V_N(\hat{\mathbf{w}} + \Delta\mathbf{w}) \approx V_N(\hat{\mathbf{w}}) + \Delta\mathbf{w}^T V_N'(\hat{\mathbf{w}}) + \frac{1}{2}\Delta\mathbf{w}^T V_N''(\hat{\mathbf{w}})\Delta\mathbf{w} \tag{4-7}$$

where $V_N'(\hat{\mathbf{w}}) = \frac{\partial}{\partial \mathbf{w}} V_N(\mathbf{w})\big|_{\mathbf{w}=\hat{\mathbf{w}}}$ is the Gradient of the cost function and $V_N''(\hat{\mathbf{w}}) = \frac{\partial^2}{\partial \mathbf{w}^2} V_N(\mathbf{w})\big|_{\mathbf{w}=\hat{\mathbf{w}}}$ is the Hessian of the cost function. The Gradient of the cost function can be stated as:

$$V_N'(\hat{\mathbf{w}}) = \sum_{n=1}^{N} \epsilon_n'(\mathbf{x}_n, \hat{\mathbf{w}})\epsilon_n(\mathbf{x}_n, \hat{\mathbf{w}}) = \boldsymbol{\epsilon}'(\mathbf{X}, \hat{\mathbf{w}})^T \boldsymbol{\epsilon}(\mathbf{X}, \hat{\mathbf{w}}) \tag{4-8}$$

where $\epsilon_n'(\mathbf{x}_n, \hat{\mathbf{w}}) = \frac{\partial}{\partial \hat{\mathbf{w}}}\epsilon_n(\mathbf{x}_n, \hat{\mathbf{w}})\big|_{\mathbf{w}=\hat{\mathbf{w}}}$ and where $\boldsymbol{\epsilon}(\mathbf{X}, \hat{\mathbf{w}})$ and $\boldsymbol{\epsilon}'(\mathbf{X}, \hat{\mathbf{w}})$ denote the vector for all prediction errors and matrix of derivatives for all prediction errors respectively. Then, the Hessian of the cost function can be defined as:

$$\begin{aligned} V_N''(\hat{\mathbf{w}}) &= \sum_{n=1}^{N} \epsilon_n'(\mathbf{x}_n, \hat{\mathbf{w}})^T \epsilon_n'(\mathbf{x}_n, \hat{\mathbf{w}}) \\ &= \boldsymbol{\epsilon}'(\mathbf{X}, \hat{\mathbf{w}})^T \boldsymbol{\epsilon}'(\mathbf{X}, \hat{\mathbf{w}}) + \sum_{n=1}^{N} \epsilon_n(\mathbf{x}_n, \hat{\mathbf{w}})\epsilon_n''(\mathbf{x}_n, \hat{\mathbf{w}}) \end{aligned} \tag{4-9}$$

where $\epsilon_n''(\mathbf{x}_n, \mathbf{w}) = \frac{\partial^2}{\partial \mathbf{w}^2}\epsilon_n(\hat{\mathbf{w}})\big|_{\mathbf{w}=\hat{\mathbf{w}}}$. The derivative of the Taylor expansion in equation (4-7) with respect to $\Delta\mathbf{w}$ can be taken and the following expression for $\Delta\mathbf{w}$ is obtained:

$$V_N''(\hat{\mathbf{w}})\Delta\mathbf{w} = -V_N'(\hat{\mathbf{w}}). \tag{4-10}$$

Since models that are linear in the parameters are considered, $\boldsymbol{\epsilon}'(\mathbf{X}, \hat{\mathbf{w}})$ is a constant matrix and $\epsilon_n''(\mathbf{x}_n, \hat{\mathbf{w}})$ is zero. Hence the second order derivative of the cost function can be defined as

$$V_N''(\hat{\mathbf{w}}) = \boldsymbol{\epsilon}'(\mathbf{X}, \hat{\mathbf{w}})^T \boldsymbol{\epsilon}'(\mathbf{X}, \hat{\mathbf{w}}). \tag{4-11}$$

It is assumed that the prediction error has zero mean and variance σ_e^2, i.e., the true system is in the model structure, thereby, $\mathbf{w}^* = \mathbf{w}^0$. By substitution of equation (4-10), the covariance of the estimated parameters can be stated as:

$$\begin{aligned} \text{Cov}(\hat{\mathbf{w}} - \mathbf{w}^0) &= \text{Cov}(\hat{\mathbf{w}}) = \text{Cov}((V_N''(\hat{\mathbf{w}}))^{-1} V_N'(\hat{\mathbf{w}})) \\ &= \text{Cov}((V_N''(\hat{\mathbf{w}}))^{-1} \boldsymbol{\epsilon}'(\mathbf{X}, \hat{\mathbf{w}})^T \boldsymbol{\epsilon}(\mathbf{X}, \hat{\mathbf{w}}) \\ &= \sigma_e^2 (V_N''(\hat{\mathbf{w}}))^{-1} \boldsymbol{\epsilon}'(\mathbf{X}, \hat{\mathbf{w}})^T \boldsymbol{\epsilon}'(\mathbf{X}, \hat{\mathbf{w}}) (V_N''(\hat{\mathbf{w}}))^{-1} \\ &= \sigma_e^2 (V_N''(\hat{\mathbf{w}}))^{-1} V_N''(\hat{\mathbf{w}}) (V_N''(\hat{\mathbf{w}}))^{-1} = \sigma_e^2 (V_N''(\hat{\mathbf{w}}))^{-1}. \end{aligned} \tag{4-12}$$

The matrix $-E\left[V_N''(\hat{\mathbf{w}})\right] = \mathcal{I}_{\mathbf{w}}$ is the well-known Fisher information matrix. Hence, the covariance matrix of the least squares estimate is proportional to the inverse of the Fisher information matrix. In general, the Fisher information matrix can be used to compute the covariance matrix of maximum likelihood estimators, which are unbiased. Moreover, the inverse of the Fisher information matrix gives a lower bound on the parameter covariance of unbiased estimators which is known as the Cramér–Rao lower bound:

$$\text{Cov}(\hat{\mathbf{w}}) \geq \mathcal{I}_{\mathbf{w}}^{-1} \tag{4-13}$$

where $\geq$ imposes positive-definiteness.

Maximizing the log-likelihood function with the assumption that the noise is Gaussian distributed is equivalent to minimizing a sum of squares cost function. Hence, a least-squares approach to obtaining optimal parameter estimates is a specific case of maximum likelihood [2]. Simply maximizing the likelihood function can lead to over-fitting of the model on the data which is a general property of maximum likelihood as. Over-fitting is closely linked to high variance, meaning the error introduced by the sensitivity of the model to small fluctuations or noise in the training data. High variance of the model causes the model to capture the noise in the training data, leading to a high degree of flexibility and poor performance on unseen data [2]. Regularization can control the over-fitting of the model, which is especially advantageous for models with many parameters. A downside of regularization is the introduction biased estimators which lead to a bias to the model predictions which shift the average model prediction away from the underlying true function leading to poor performance on both training data and test data. The trade-off between model flexibility and bias is known as the bias-variance trade-off. In addition, the Cramér–Rao lower bound in equation (4-13) is not directly applicable as the parameters in this research inhibit deliberate bias introduced by the regularization term.

It is important to note that the final parameter covariance expression in equation (4-12) can only be used when the underlying true system is part of the model structure. This is because, there are no structural model errors in this case. Hence, the asymptotic prediction error $\epsilon_n(\mathbf{x}_n, \mathbf{w}^*) = e_n$ only contains the aleatory error term e_n which is the inherent randomness that is incorporated in the observations. In contrast, when the underlying true system is not part of the model structure, the asymptotic prediction error $\epsilon_n(\mathbf{x}_n, \mathbf{w}^*) = e_n + f^*(\mathbf{x}_n) - f(\mathbf{x}_n, \mathbf{w}^*)$ contains the aleatory term e_n as well as a structural model error term $f^*(\mathbf{x}_n) - f(\mathbf{x}_n, \mathbf{w}^*)$ which is defined in equation (4-4). In this case, the gradient of the prediction error $\epsilon_n(\mathbf{x}_n, \mathbf{w}^*)$ depends on the input $\mathbf{x}_n$, thereby, the independence between the two terms is lost [44]. Hence, $E[\nabla f(\mathbf{x}_n, \mathbf{w}^*)^T \epsilon_n(\mathbf{x}_n, \mathbf{w}^*)^T \epsilon_n(\mathbf{x}_n, \mathbf{w}^*) \nabla f(\mathbf{x}_n, \mathbf{w}^*))] \neq E[\nabla f(\mathbf{x}_n, \mathbf{w}^*)^T E[\epsilon_n(\mathbf{x}_n, \mathbf{w}^*)^T \epsilon_n(\mathbf{x}_n, \mathbf{w}^*)] \nabla f(\mathbf{x}_n, \mathbf{w}^*))]$.

When the structural model error is unknown, the parameter covariance matrix cannot be computed as higher-order moments of the input $\mathbf{x}_n$ show up in the computation of the general expression. These terms cannot be computed when the structural model errors are unknown and it becomes impossible to obtain a closed-form expression of the parameter covariance matrix. Therefore, when there is significant structural model error, the parameter covariance expression underestimates the variability which causes a far too optimistic approximation of the uncertainty of the estimates [17, 44]. In [30], a asymptotic parameter covariance expression is derived for least squares estimators of nonlinear models. It was found that the parameter covariance in the general case suffers from the same problem as the linear case in that the structural model error should be exactly known in order to accurately compute the parameter covariance. In practice, the structural model error is hardly ever known. In [44], it is emphasized that the parameter covariance matrix underestimates the true value as it does not take into account the structural model errors and the input when wrongly assuming that there is no structural model and using the corresponding expressions accordingly.

4-1-3 Asymptotic parameter covariance for nonlinear least squares

In [27, 28], an asymptotic covariance expression is derived for parameters that minimize a least squares cost function. These results treat $\hat{\mathbf{w}} - \mathbf{w}^*$ as a random variable and it is shown that the parameter estimation error $\sqrt{N}(\hat{\mathbf{w}} - \mathbf{w}^*)$ is Gaussian. The provided expressions are asymptotic results, meaning asymptotic in the number of samples N. However, the resulting expressions can be approximated by non-asymptotic estimates which provide reasonable and valuable results.

Not in the model structure This heuristic approach is a review from [28, 27] and starts with the general case that the true function is not necessarily in the model structure ($f^* \notin \mathcal{M}$). Consider the following estimated parameters:

$$\hat{\mathbf{w}} = \arg\min \frac{1}{N} \sum_{n=1}^{N} \epsilon_n^2(\mathbf{x}_n, \hat{\mathbf{w}}). \tag{4-14}$$

Hence, the cost function that is optimized is given by $V_N(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \epsilon_n^2(\mathbf{x}_n, \mathbf{w})$. The error term can be stated as follows:

$$\epsilon_n(\mathbf{x}_n, \hat{\mathbf{w}}) = y_n - f(\mathbf{x}_n, \hat{w}) = e_n + f^*(\mathbf{x}_n) - f(\mathbf{x}_n, \mathbf{w}). \tag{4-15}$$

where $f^*(\mathbf{x}_n)$ is the true function, $f(\mathbf{x}_n, \mathbf{w})$ is the nonlinear model and e_n is zero mean Gaussian noise with variance σ_e^2. Furthermore, the gradient of the model evaluated at the asymptotic parameter estimates $\mathbf{w}^*$ and the gradient of the prediction error evaluated at $\mathbf{w}^*$ are equal in the following way:

$$\nabla f(\mathbf{x}_n, \mathbf{w}^*) = -\frac{\partial}{\partial \mathbf{w}} \epsilon_n(\mathbf{x}_n, \mathbf{w})\big|_{\mathbf{w}=\mathbf{w}^*} = \frac{\partial}{\partial \mathbf{w}} f(\mathbf{x}_n, \mathbf{w})\big|_{\mathbf{w}=\mathbf{w}^*}. \tag{4-16}$$

Also, consider the following notation for the gradient and the Hessian of the cost function:

$$V_N'(\mathbf{w}^*) = \frac{\partial}{\partial \mathbf{w}} V_N\big|_{\mathbf{w}=\mathbf{w}^*} \qquad V_N''(\mathbf{w}^*) = \frac{\partial^2}{\partial \mathbf{w}^2} V(\mathbf{w})\big|_{\mathbf{w}=\mathbf{w}^*}. \tag{4-17}$$

It can be immediately stated that

$$V_N'(\hat{\mathbf{w}}) = 0 \tag{4-18}$$

as $\hat{\mathbf{w}}$ minimizes V_N. Then, by employing a Taylor series expansion for the gradient of the cost function around $\mathbf{w}^*$, the following equation is obtained:

$$0 = V_N'(\mathbf{w}^*) + V_N''(\xi_N)(\hat{\mathbf{w}} - \mathbf{w}^*) \tag{4-19}$$

where ξ_N is a value between $\hat{\mathbf{w}}$ and $\mathbf{w}^*$. Next, the notion of an asymptotic mean for a random variable $g(t)$ should be defined which can be stated as follows:

$$\bar{E}[g(t)] = \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} E[g(t)]. \tag{4-20}$$

Then, it can be shown that $V_N''(\mathbf{w})$ converges uniformly in $\mathbf{w}$ to $\bar{V}''(\mathbf{w})$ which gives:

$$V_N''(\xi_N) \to \bar{V}''(\mathbf{w}^*) \qquad \text{as} \quad N \to \infty \text{ w.p. 1.} \tag{4-21}$$

Given this convergence, equation (4-19) can be rewritten in the following form:

$$(\hat{\mathbf{w}} - \mathbf{w}^*) = -[\bar{V}''(\mathbf{w}^*)]^{-1} V_N'(\mathbf{w}^*). \tag{4-22}$$

Here, the second term can be written as:

$$-V_N'(\mathbf{w}^*) = \frac{1}{N} \sum_{n=1}^{N} 2\nabla f(\mathbf{x}_n, \mathbf{w}^*)\epsilon_n(\mathbf{x}_n, \mathbf{w}^*). \tag{4-23}$$

In [28], the gradient of the cost function as defined in equation (4-16) can be expressed as a sum of an asymptotically normal distributed random variable with zero mean and the difference can be stated as

$$D_N = E\left[\frac{1}{N}\sum_{n=1}^{N} \nabla f(\mathbf{x}_n, \mathbf{w}^*)\epsilon_n(\mathbf{x}_n, \mathbf{w}^*) + \nabla R(\mathbf{w}^*) - \bar{E}[\nabla f(\mathbf{x}_n, \mathbf{w}^*)\epsilon_n(\mathbf{x}_n, \mathbf{w}^*) - \nabla R(\mathbf{w}^*)]\right]. \tag{4-24}$$

Also, by definition, the asymptotic mean of the gradient of the cost function is

$$\bar{V}'(\mathbf{w}) = -\bar{E}[2\nabla f(\mathbf{x}_n, \mathbf{w}^*)\epsilon_n(\mathbf{x}_n, \mathbf{w}^*)] = 0 \tag{4-25}$$

and therefore, the expression in equation (4-23) is asymptotically normal distributed with mean 0 and covariance Q, apart from the difference which is assumed to converge to 0 fast enough, i.e. $\sqrt{N}D_N \to 0$ as $N \to \infty$. Therefore, the gradient of the cost function in equation (4-23) is a sum of random variables with zero mean values. Given this, the following final result is proven in [30]:

$$\sqrt{N}(\hat{\mathbf{w}} - \mathbf{w}^*) \to \mathcal{N}(0, \mathbf{P}_{\mathbf{w}^*}) \quad \text{as} \quad N \to \infty \tag{4-26}$$

where

$$\begin{aligned} \mathbf{P}_{\mathbf{w}^*} &= [\bar{V}''(\mathbf{w}^*)]^{-1} Q [\bar{V}''(\mathbf{w}^*)]^{-1} \\ Q &= \lim_{N\to\infty} NE\left[[V_N'(\mathbf{w}^*)]^T [V_N'(\mathbf{w}^*)]\right]. \end{aligned} \tag{4-27}$$

In the model structure The second case is a special case of the first case. In this case, the underlying true system is part of the model structure ($f^* \in \mathcal{M}$). Therefore, in this case $\mathbf{w}^* = \mathbf{w}^0$, and thereby, the prediction error $\epsilon_n(\mathbf{x}_n, \mathbf{w}^*) = e_n$ has zero mean and variance σ_e^2. As a consequence of true system being in the model structure, $D_N = 0$ and $Q = \sigma_e^2 \bar{V}''(\mathbf{w}^0)$. In addition, $\bar{V}''(\mathbf{w}^0)$ can be written as:

$$\begin{aligned}\bar{V}''(\mathbf{w}^0) &= 2\bar{E}\left[\nabla f(\mathbf{x}_n, \mathbf{w}^0)^T \nabla f(\mathbf{x}_n, \mathbf{w}^0)\right] - \bar{E}\left[2\nabla f(\mathbf{x}_n, \mathbf{w}^0)^T e_n\right] \\ &= 2\bar{E}\left[\nabla f(\mathbf{x}_n, \mathbf{w}^0)^T \nabla f(\mathbf{x}_n, \mathbf{w}^0)\right].\end{aligned} \tag{4-28}$$

Therefore, the parameter covariance can be reduced to

$$\begin{aligned}\mathbf{P}_{\mathbf{w}^0} &= \sigma_e^2 \boldsymbol{\mathcal{I}}_{\mathbf{w}}^{-1} \\ \boldsymbol{\mathcal{I}}_{\mathbf{w}} &= 2\bar{E}\left[\nabla f(\mathbf{x}_n, \mathbf{w}^0)^T \nabla f(\mathbf{x}_n, \mathbf{w}^0)\right].\end{aligned} \tag{4-29}$$

In this formulation of the parameter covariance, $\boldsymbol{\mathcal{I}}_{\mathbf{w}}$ is the Fisher information matrix of the parameters.

Approximating the covariance When applying the delta method in order to quantify the uncertainty in the case of a dataset, the simplified version as in equation (4-29) can be used. This is because the structural model error is unknown in this case. Therefore, it is assumed that the true underlying model is part of the model structure, and thereby it is assumed that there is no structural model error within the prediction error. The expression for the parameter covariance that is used is thus the one in equation (4-29), however, this expression is asymptotic in the number of samples. In order to approximate the expression with finite amount of data the following approximation can be used [27]:

$$\begin{aligned}\hat{\mathbf{P}}_N &= \hat{\sigma}_e^2 \left[\frac{2}{N}\sum_{n=1}^{N} \nabla f(\mathbf{x}_n, \hat{\mathbf{w}})^T \nabla f(\mathbf{x}_n, \hat{\mathbf{w}})\right]^{-1}, \\ \hat{\sigma}_e^2 &= \frac{1}{N}\sum_{n=1}^{N} \epsilon_n^2(\mathbf{x}_n, \hat{\mathbf{w}}).\end{aligned} \tag{4-30}$$

4-2 Delta method

Here, the main theory and results on the delta method are discussed. As opposed to a Bayesian approach where model parameters are treated as random variables, the delta method is a frequentist approach that treats model parameters as single values.

4-2-1 Parameter covariance for the CPD constrained kernel machine

In order to compute the prediction error variance for the CPD constrained kernel machine, the covariance matrix of the parameter estimates $\mathbf{P}_{\hat{\mathbf{w}}}$ needs to be constructed as this covariance of the parameters affects the uncertainty in the predictions. More specifically, the covariance of the model parameters can be propagated to the covariance of the model output. Thereby, the variance originating from estimation errors $\sigma^2_{\hat{f}_n}$ can be estimated. In what follows, a heuristic approach to derive parameter covariance expressions is provided. The cost function that is dealt with in equation (3-19) comprises of a mean squared error term and a regularization term. The following assumptions must hold in order for the delta method to provide accurate uncertainty quantification results:

- The true system is described by $\mathbf{w}^* = \mathbf{w}^0$ (i.e. $f^* \in \mathcal{M}$).
- $\bar{V}''(\mathbf{w}^0)$ is invertible.
- Training data and test data have the same second order properties, meaning statistical properties related to the data distribution are similar.

Then, the approach described in section 4-1-3 can directly be applied to give the following asymptotic covariance [27, 9]:

$$\mathbf{P}_{\mathbf{w}^0} = [\bar{V}''(\mathbf{w}^0) + \mathbf{H}_R(\mathbf{w}^0)]^{-1}[\sigma_e^2 \bar{V}''(\mathbf{w}^0)][\bar{V}''(\mathbf{w}^0) + \mathbf{H}_R(\mathbf{w}^0)]^{-1}. \tag{4-31}$$

where $\mathbf{H}_R(\mathbf{w})$ is the Hessian of the regularization term evaluated at $\mathbf{w}^0$. This expression is asymptotic in the number of samples which is not feasible in practice. The following approximation can be used in order to approximate the expression with a finite amount of data:

$$\begin{aligned} \hat{\mathbf{P}}_N &= \hat{\sigma}_e^2 \left[\hat{V}_N'' + \mathbf{H}_R(\hat{\mathbf{w}})\right]^{-1} \hat{V}_N'' \left[\hat{V}_N'' + \mathbf{H}_R(\hat{\mathbf{w}})\right]^{-1}, \\ \text{where} \quad \hat{\sigma}_e^2 &= \frac{1}{N}\sum_{n=1}^{N} \epsilon_n^2(\mathbf{x}_n, \hat{\mathbf{w}}), \\ \text{and} \quad \hat{V}_N''(\hat{\mathbf{w}}) &= \frac{2}{N}\sum_{n=1}^{N} \nabla f(\mathbf{x}_n, \hat{\mathbf{w}})^T \nabla f(\mathbf{x}_n, \hat{\mathbf{w}}). \end{aligned} \tag{4-32}$$

The error term can directly be stated from the optimization problem in equation (3-19), which is

$$\epsilon_n(\mathbf{x}_n, \hat{\mathbf{w}}) = y_n - f_n(\mathbf{x}_n, \hat{\mathbf{w}}) = y_n - \left\langle \operatorname{vec}\left(\hat{\mathbf{W}}^{(d)}\right), \mathbf{z}(x_n^d) \otimes \left(\mathop{\circledast}_{p=1, p\neq d}^{D} \mathbf{z}(x_n^{(p)})^T \hat{\mathbf{W}}^{(p)} \right) \right\rangle. \tag{4-33}$$

Furthermore, it can be observed that $\sum_{n=1}^{N} \nabla f(\mathbf{x}_n, \hat{\mathbf{w}})^T \nabla f(\mathbf{x}_n, \hat{\mathbf{w}})$ constitutes the Jacobian when considering all data samples. In this Jacobian, the prediction model for all samples $\mathbf{f}(\mathbf{X}, \mathbf{w})$ are considered and the gradient with respect to each factor matrix $\operatorname{vec}(\mathbf{W}^{(d)})$ is concatenated. This is done as it is clear from the optimization problem in equation (3-19) that both the mean squared error term and the regularization term, which are functions of the CPD constrained parameters in the cost function, can be written linearly in terms of a vectorized factor matrix. This is a special property of the CPD constrained kernel machine. In this regard, the Jacobian can be stated as follows:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f(\mathbf{x}_1,\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(1)})}\Big|_{\mathbf{w}=\hat{\mathbf{w}}} & \frac{\partial f(\mathbf{x}_1,\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(2)})}\Big|_{\mathbf{w}=\hat{\mathbf{w}}} & \cdots & \frac{\partial f(\mathbf{x}_1,\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(D)})}\Big|_{\mathbf{w}=\hat{\mathbf{w}}} \\ \frac{\partial f(\mathbf{x}_2,\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(1)})}\Big|_{\mathbf{w}=\hat{\mathbf{w}}} & \frac{\partial f(\mathbf{x}_2,\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(2)})}\Big|_{\mathbf{w}=\hat{\mathbf{w}}} & \cdots & \frac{\partial f(\mathbf{x}_2,\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(D)})}\Big|_{\mathbf{w}=\hat{\mathbf{w}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(\mathbf{x}_N,\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(1)})}\Big|_{\mathbf{w}=\hat{\mathbf{w}}} & \frac{\partial f(\mathbf{x}_N,\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(2)})}\Big|_{\mathbf{w}=\hat{\mathbf{w}}} & \cdots & \frac{\partial f(\mathbf{x}_N,\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(D)})}\Big|_{\mathbf{w}=\hat{\mathbf{w}}} \end{bmatrix} \tag{4-34}$$

where the Jacobian has size $N \times D\hat{M}R$. Recall the prediction model for all samples from equation (3-25):

$$\mathbf{f}(\mathbf{X},\mathbf{w}) = \left(\mathbf{Z}(\mathbf{X}^{(d)}) \odot \left(\mathop{\circledast}_{p=1,p\neq d}^{D} \mathbf{Z}(\mathbf{X}^{(p)})^T\mathbf{W}^{(p)}\right)^T\right)^T \text{vec}(\mathbf{W}^{(d)}) = \mathbf{C}^{(d)}\text{vec}(\mathbf{W}^{(d)}) \tag{4-35}$$

which is linear in $\text{vec}(\mathbf{W}^{(d)})$ keeping all other factor matrices constant. Thereby, the Jacobian in equation (4-34) can be written as

$$\begin{aligned} \mathbf{J} &= \left[\frac{\partial \mathbf{f}(\mathbf{X},\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(1)})}\Big|_{\mathbf{w}=\hat{\mathbf{w}}} \frac{\partial \mathbf{f}(\mathbf{X},\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(2)})}\Big|_{\mathbf{w}=\hat{\mathbf{w}}}, \dots, \frac{\partial \mathbf{f}(\mathbf{X},\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(D)})}\Big|_{\mathbf{w}=\hat{\mathbf{w}}}\right] \\ &= \left[\hat{\mathbf{C}}^{(1)}\ \hat{\mathbf{C}}^{(2)}, \dots, \hat{\mathbf{C}}^{(D)}\right]. \end{aligned} \tag{4-36}$$

where the hat on top of $\mathbf{C}$ denotes terms that are evaluated at the final estimated parameters such that $\hat{\mathbf{C}}^{(d)}$ is evaluated at the estimated factor matrices except the d-th. Lastly, in order to state the Hessian of the regularization term, first recall the gradient of the regularization term with respect to one of the factor matrices from equation (3-29):

$$\frac{\partial R(\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(d)})} = 2\lambda_{\text{reg}}((\mathop{\circledast}_{p=1,p\neq d}^{D} \mathbf{W}^{(p)^T}\mathbf{W}^{(p)}) \otimes \mathbf{I}_M)\text{vec}(\mathbf{W}^{(d)}). \tag{4-37}$$

To compute the Hessian of the entire regularization term, the second-order derivatives with respect to each factor matrix have to be by obtained by taking the second derivative with respect to each combination of vectorized factor matrices: $\frac{\partial^2 R(\mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(d)})\partial \text{vec}(\mathbf{W}^{(k)})}$ for $d = 1, \dots, D$ and $k = 1, \dots, D$. An analytical expression for the off-diagonal blocks of this Hessian can be hard to compute as the derivative with respect to $\text{vec}(\mathbf{W}^{(d)})$ in equation (4-37) is not straightforward. Therefore, an approximation of the Hessian is computed by considering the block-diagonal elements as follows:

$$\mathbf{H}_R(\mathbf{w}) = 2\lambda_{\text{reg}} \begin{bmatrix} \overline{\mathbf{W}}_{\backslash 1} & 0 & \dots & 0 \\ 0 & \overline{\mathbf{W}}_{\backslash 2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \overline{\mathbf{W}}_{\backslash D} \end{bmatrix} \tag{4-38}$$

where $\overline{\mathbf{W}}_{\setminus d} = (\circledast_{p=1,p\neq d}^{D} \mathbf{W}^{(p)^T}\mathbf{W}^{(p)}) \otimes \mathbf{I}_M$. As the delta method will be combined with a cross-validation procedure, which will be further elaborated on in section 6-2-2, the delta method has 'freedom' to allow for less or more regularization by means of the λ_{reg} regularization hyperparameter. In this regard, it is assumed that the impact of only considering the block diagonal terms of the Hessian of the regularization term remains limited.

Ultimately, following the form in equation (4-32), the parameter covariance can be approximated by:

$$\hat{\mathbf{P}}_N = \hat{\sigma}_e^2(2\mathbf{J}^T\mathbf{J} + \mathbf{H}_R(\hat{\mathbf{w}}))^{-1}(2\mathbf{J}^T\mathbf{J})(2\mathbf{J}^T\mathbf{J} + \mathbf{H}_R(\hat{\mathbf{w}}))^{-1}. \tag{4-39}$$

4-2-2 Prediction uncertainty by the delta method

The delta method can be used to approximate the prediction error variance for a function of a random variable when the function is well-behaved and can be approximated by its first-order Taylor series expansion. In this regard, the most important assumptions for the method to be accurate are [39]:

- The nonlinear function can be well-approximated by a linear function in the vicinity of the linearization point.
- The random estimators involved in the function should have approximately Gaussian distributions, which is often the case with high sample size, and are consistent, meaning that they converge to their true values when the sample size increases.
- The random variables involved in the delta method should be independent.
- By the Central Limit Theorem the random parameter vector $\hat{\mathbf{w}}$ should have an approximate Gaussian distribution: $\sqrt{N}(\hat{\mathbf{w}} - \mathbf{w}^*) \to \mathcal{N}(0, \mathbf{P}_{\mathbf{w}^*})$ as $N \to \infty$.

The prediction error variance quantifies the uncertainty in the predictions of the function due to the uncertainty in the model parameters. The prediction uncertainty can be stated by this linearization technique as follows [28, 22, 10]:

$$f(\mathbf{x}, \hat{\mathbf{w}}) \approx f(\mathbf{x}_n, \mathbf{w}^*) + \nabla f(\mathbf{x}_n, \mathbf{w}^*)^T(\hat{\mathbf{w}} - \mathbf{w}^*). \tag{4-40}$$

For ease of notation, the estimated model $f(\mathbf{x}, \hat{\mathbf{w}})$ is denoted by $\hat{f}$. Given the last assumption in 4-2-2 it holds by the delta method that

$$\sqrt{N}\left(\hat{f}_n - f(\mathbf{x}_n, \mathbf{w}^*)\right) \to \mathcal{N}(0, \mathbb{V}(\hat{f}_n)) \quad \text{as} \quad N \to \infty. \tag{4-41}$$

The expression for the prediction error can then be deduced for the n-th sample as follows:

$$\begin{aligned}
\mathbb{V}(\hat{f}_n) &\approx \mathbb{V}\left(f(\mathbf{x}_n, \mathbf{w}^*) + \nabla f(\mathbf{x}_n, \mathbf{w}^*)^T(\hat{\mathbf{w}} - \mathbf{w}^*)\right) \\
&= \mathbb{V}\left(f(\mathbf{x}_n, \mathbf{w}^*) + \nabla f(\mathbf{x}_n, \mathbf{w}^*)^T\hat{\mathbf{w}} - \nabla f(\mathbf{x}_n, \mathbf{w}^*)^T\mathbf{w}^*\right) \\
&= \mathbb{V}\left(f(\mathbf{x}_n, \mathbf{w}^*)^T\hat{\mathbf{w}}\right) \\
&= \nabla f(\mathbf{x}_n, \mathbf{w}^*)^T\mathbf{P}_{\mathbf{w}*}\nabla f(\mathbf{x}_n, \mathbf{w}^*).
\end{aligned} \tag{4-42}$$

With finite amount of data, equation (4-42) can be approximated by replacing $\mathbf{w}^*$ with $\hat{\mathbf{w}}$. By considering all samples at once, the following expression provides the Jacobian when considering test data, which is denoted by a lower subscript ($_*$):

$$\begin{aligned}\mathbf{g} &= \left[\frac{\partial \mathbf{f}(\mathbf{X}_*, \mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(1)})}\bigg|_{\mathbf{w}=\hat{\mathbf{w}}} \frac{\partial \mathbf{f}(\mathbf{X}_*, \mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(2)})}\bigg|_{\mathbf{w}=\hat{\mathbf{w}}}, \dots, \frac{\partial \mathbf{f}(\mathbf{X}_*, \mathbf{w})}{\partial \text{vec}(\mathbf{W}^{(D)})}\bigg|_{\mathbf{w}=\hat{\mathbf{w}}} \right] \\ &= \left[\hat{\mathbf{C}}_*^{(1)} \; \hat{\mathbf{C}}_*^{(2)}, \dots, \hat{\mathbf{C}}_*^{(D)} \right].\end{aligned} \tag{4-43}$$

By substituting the approximated parameter covariance of equation (4-39) into equation (4-42) and considering the Jacobian of equation (4-43), the prediction error variance can be stated as:

$$\begin{aligned}\mathbb{V}(\hat{f}) &= \text{diag}\left(\hat{\sigma}_e^2 \mathbf{g}(2\mathbf{J}^T\mathbf{J} + \mathbf{H}_R(\hat{\mathbf{w}}))^{-1}(2\mathbf{J}^T\mathbf{J})(2\mathbf{J}^T\mathbf{J} + \mathbf{H}_R(\hat{\mathbf{w}}))^{-1}\mathbf{g}^T\right), \\ \text{where} \quad \hat{\sigma}_e^2 &= \frac{1}{N}\sum_{n=1}^{N}(y_n - f_n(\mathbf{x}_n, \hat{\mathbf{w}}))^2,\end{aligned} \tag{4-44}$$

and where $\mathbb{V}(\hat{f}) = [\mathbb{V}(\hat{f}_1) \; \mathbb{V}(\hat{f}_2), \dots, \mathbb{V}(\hat{f}_N)]^T$ the prediction error variance vector corresponding to the predictions and σ_e^2 is the observation noise variance. The diagonal in equation (4-44) is taken as the diagonal elements of the covariance matrix represent the variances of the individual predictions. Since $\mathbf{J}$ is of size $N \times D\hat{M}R$ and $\mathbf{g}$ is of size $N_{\text{test}} \times D\hat{M}R$, the bottleneck in computing $(2\mathbf{J}^T\mathbf{J} + \mathbf{H}_R(\hat{\mathbf{w}}))^{-1}(2\mathbf{J}^T\mathbf{J})(2\mathbf{J}^T\mathbf{J} + \mathbf{H}_R(\hat{\mathbf{w}}))^{-1})$ is the construction and inversion of $\mathbf{J}^T\mathbf{J}$ which has a computational complexity of $\mathcal{O}((D\hat{M}R)^2N + (D\hat{M}R)^3)$. Multiplying the parameter covariance with the gradient of the model with respect to the factor matrices evaluated at the estimated parameters from the left and its transpose on the right in equation (4-44) has a computational complexity of $\mathcal{O}((D\hat{M}R)^2N_{\text{test}})$.

4-2-3 Uncertainty intervals

It is important to explicitly indicate what the variance in the model outcome consists of since the uncertainties in the predictions originate from different sources. Recall the general case of the prediction error in equation (4-4). It is assumed that the aleatory uncertainty is independent of the epistemic uncertainty. From equation (4-44), the variance of the estimated test inputs $\mathbb{V}(f_*) = [\mathbb{V}(f_{*1}) \; \mathbb{V}(f_{*2}), \dots, \mathbb{V}(f_{*N})]^T$ is a sum of the prediction error variance and the observation noise variance, as such

$$\mathbb{V}(f_*) = \mathbb{V}(\hat{f}) + \sigma_e^2. \tag{4-45}$$

Hence, the error originating from structural model errors is not included in the estimated prediction error. It is important to distinguish two main uncertainty measures: confidence intervals (CIs) and prediction intervals (PIs). CIs are concerned with the variance originating from model errors. Hence, CIs try to quantify the uncertainty between the prediction $f(\mathbf{x}_n, \hat{\mathbf{w}})$ and the underlying true function f^*. In this regard, the noise variance term σ_e^2 in equation (4-45) is omitted. On the other hand, PIs are concerned with the variance originating from both the model errors and the observation noise σ_e^2. Hence, PIs try to quantify the uncertainty between the observations y_n and the prediction $f(\mathbf{x}_n, \hat{\mathbf{w}})$. Thus, CIs do not take into account the inherent variability in the individual predictions, thereby, they will be narrower than PIs. PIs in a classification task are only concerned with the model errors as this gives information about the uncertainty associated with the model predictions. More specifically, the PIs in a classification task give information on the range of values at which the classifier casts doubt on its prediction with a certain confidence level. Thereby, the PIs in a classification task are equal to the CIs in a regression task. Thereby, approximate 95% CIs for a regression task are computed by:

$$\mathbf{f}(\mathbf{X}, \mathbf{w}) \pm 2\sqrt{\mathbb{V}(\hat{f})}. \tag{4-46}$$

Then, approximate 95% PIs for a regression task are computed by:

$$\mathbf{f}(\mathbf{X}, \mathbf{w}) \pm 2\sqrt{\mathbb{V}(\hat{f}) + \sigma_e^2}. \tag{4-47}$$

It is important to emphasize that these approximate uncertainty intervals hold as it is assumed that the noise variance is homoscedastic and Gaussian distributed. Confidence intervals for a classification task can be directly be constructed based on the regression framework [7]. For a classification task, confidence intervals are computed by:

$$\text{sign}\left(\mathbf{f}(\mathbf{X}, \mathbf{w}) \pm 2\sqrt{\mathbb{V}(\hat{f})}\right). \tag{4-48}$$

Chapter 5

Uncertainty quantification: A Bayesian approach

The machine learning prediction model that is considered in this research is the Canonical Polyadic Decomposition (CPD) constrained kernel machine. The optimization problem that is solved in order to fit this model to data is given in equation (3-15). However, such a prediction model does not provide any uncertainty quantification by itself. The frequentist approach to model uncertainty quantification was based on the delta method. The other two methods that will quantify the uncertainty in the form of prediction intervals (PIs) will be a Bayesian approach and the Single Bayesian Core (SBC) method which contains both frequentist and Bayesian aspects.
In the frequentist framework, the coverage probability of an uncertainty interval refers to the percentage of times that the interval will contain future observations for PIs and will contain the underlying true function for confidence intervals (CIs) over an infinite number of repeated sampling. It is a long-run property based on the concept of repeated sampling. On the other hand, Bayesian prediction intervals do not have a direct uncertainty interval coverage probability interpretation as frequentist intervals. In a Bayesian approach, prediction intervals are constructed based on the posterior distribution, which represents the updated belief about the parameters after incorporating the observed data and prior information. A Bayesian prediction interval provides a probability distribution of possible values for the future observation, and the interval represents the range of values with a certain probability. Despite the fact that uncertainty intervals of a frequentist and Bayesian approach differ slightly in their interpretation, it is still interesting to evaluate which constructed uncertainty intervals actually deliver more accurate uncertainty measures on a test set.
Furthermore, it should be noted that the observation noise variance σ_e^2 is either processed as a hyperparameter denoted by $\bar{\sigma}_e^2$ or an estimator denoted by $\hat{\sigma}_e^2$.

5-1 Bayesian inference with stochastic cores

The Bayesian approach provides an inherently different way of approaching quantifying the uncertainty compared to frequentist approach. Bayesian machine learning prediction models provide a probabilistic framework to make predictions and quantify uncertainty in model parameters and predictions. A Bayesian framework starts with prior beliefs about model parameters and updates them based on observed data to obtain the posterior distribution, which reflects the updated beliefs about the parameters.

5-1-1 Bayesian CP-ALS

In [34], Bayesian inference in performed for low-rank tensor approximation. Ultimately, this leads to a probabilistic interpretation of ALS. The goal is to incorporate the same strategy in order to provide a Bayesian framework of the Canonical Polyadic Decomposition Alternating Linear Scheme (CP-ALS) algorithm. In order for this method to be accurate, the following assumptions can be stated:

- The prior distribution of each parameter factor matrix is assumed to be Gaussian and statistically independent.
- The observation noise σ_e^2 is assumed to be Gaussian, independent and identically distributed.

Modelling a prior First, a prior distribution for the model parameters is specified. The prior distribution represents our initial beliefs about the parameters before observing any data. It encapsulates any existing information, expert knowledge, or assumptions about the model. In the case of the CP-ALS algorithm a multivariate Gaussian prior is assigned to every factor matrix of the CPD. It is important to note that $\mathbf{w}^{(d)} \in \mathbb{R}^{\hat{M}R\times 1}$ is the vectorization of the d-th factor matrix $\text{vec}(\mathbf{W}^{(d)})$. In mathematical terms the following prior is obtained:

$$p(\mathbf{w}^{(d)}) = \mathcal{N}(\mathbf{m}_d^0, \mathbf{P}_d^0), \quad d = 1, \dots, D. \tag{5-1}$$

Here, $\mathbf{m}_d^0$ is the prior mean and $\mathbf{P}_d^0$ is the prior covariance matrix. The factor matrices are assumed to be statistically independent. Hence, the joint prior distribution is given as

$$p(\{\mathbf{w}^{(d)}\}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{m}_1^0 \\ \mathbf{m}_2^0 \\ \vdots \\ \mathbf{m}_D^0 \end{bmatrix}, \begin{bmatrix} \mathbf{P}_1^0 & 0 & \cdots & 0 \\ 0 & \mathbf{P}_2^0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{P}_D^0 \end{bmatrix}\right). \tag{5-2}$$

Here, $\{\mathbf{w}^{(d)}\}$ denotes the priors of all factor matrices. By the statistical independence assumption, the joint prior distribution is written as follows:

$$p(\{\mathbf{w}^{(d)}\}) = p(\mathbf{w}^{(1)})p(\mathbf{w}^{(2)})\dots p(\mathbf{w}^{(D)}). \tag{5-3}$$

By the same independence, the prior on one factor matrix conditioned on all other factor matrices is written as follows:

$$p(\mathbf{w}^{(d)})|\{\mathbf{w}^{(d)}\}_{\setminus d}) = p(\mathbf{w}^{(d)}). \tag{5-4}$$

Here, $\{\mathbf{w}^{(d)}\}_{\setminus d}$ is the collection of all factor matrices except for the d-th.

Computing the posterior The goal is to find the joint posterior distribution $p(\{\mathbf{w}^{(d)}\}|\mathbf{y})$ by means of Bayes' rule. In order to do this, first the posterior distribution of one factor matrix is stated, given the measurement and the other factor matrices. Therefore, the likelihood function needs to be defined. The likelihood function represents the probability of observing the data given the model and its parameters. It quantifies how well the model explains the observed data for a specific set of parameter values. The likelihood is defined as:

$$\begin{aligned} p(\mathbf{y}|\{\mathbf{w}^{(d)}\}) &= \mathcal{N}(f(\mathbf{x}_n, \mathbf{w}), \sigma_e^2 \mathbf{I}) \\ \text{where} \quad f(\mathbf{x}_n, \mathbf{w}) &= \left(\mathbf{w}^{(d)}\right)^T \mathbf{z}(x_n^{(d)}) \otimes \left(\underset{p=1, p\neq d}{\overset{D}{\circledast}} \mathbf{z}(x_n^{(p)})^T \mathbf{W}^{(p)} \right). \end{aligned} \tag{5-5}$$

Bayes' theorem is used to update the prior distribution to obtain the posterior distribution of the model parameters. It combines the prior distribution, the likelihood function, and the observed data. Given that the factor matrices are statistically independent and are multilinear, the posterior distribution $p(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\setminus d}, \mathbf{y}) = \mathcal{N}(\mathbf{m}_d^+, \mathbf{P}_d^+)$ of the d-th factor matrix given the observations and all other factor matrices can be stated by Bayes' rule:

$$p(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\setminus d}, \mathbf{y}) = \frac{p(\mathbf{y}|\{\mathbf{w}^{(d)}\})p(\mathbf{w}^{(d)})}{p(\mathbf{y}|\{\mathbf{w}^{(d)}\}_{\setminus d})}. \tag{5-6}$$

This posterior is Gaussian since the likelihood $p(\mathbf{y}|\{\mathbf{w}^{(d)}\})$ and the prior $p(\mathbf{w}^{(d)})$ are Gaussian and have mean $\mathbf{m}_d^+$ and covariance $\mathbf{P}_d^+$ [42]:

$$\begin{aligned} \mathbf{m}_d^+ &= \mathbb{E}(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\setminus d}, \mathbf{y}) = \left[(\mathbf{P}_d^0)^{-1} + \frac{\mathbf{C}^T\mathbf{C}}{\bar{\sigma}_e^2}\right]^{-1} \left[\frac{\mathbf{C}^T\mathbf{y}}{\bar{\sigma}_e^2} + (\mathbf{P}_d^0)^{-1}\mathbf{m}_d^0\right] \\ \mathbf{P}_d^+ &= \mathbb{V}(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\setminus d}, \mathbf{y}) = \left[(\mathbf{P}_d^0)^{-1} + \frac{\mathbf{C}^T\mathbf{C}}{\bar{\sigma}_e^2}\right]^{-1} \end{aligned} \tag{5-7}$$

where $\mathbb{E}(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\setminus d}$ denotes the posterior mean, $\mathbb{V}(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\setminus d}$ denotes the posterior covariance and the matrix $\mathbf{C}$ is defined in equation (3-26).

Bayesian CP-ALS Given the update equations in equation (5-7), the goal is now to find the joint posterior distribution of all factor matrices. Hence, $p(\{\mathbf{w}^{(d)}\}|\mathbf{y})$ is the posterior joint distribution of all factor matrices given the observations $\mathbf{y}$. Given the statistical independence assumption given in equation (5-3) and by utilizing Bayes' rule, the joint posterior can be stated as:

$$p(\{\text{vec}(\mathbf{W}^{(d)})\})|\mathbf{y}) = \frac{p(\mathbf{y}|\{\mathbf{w}^{(d)}\})p(\mathbf{w}^{(1)})p(\mathbf{w}^{(2)})\dots p(\mathbf{w}^{(D)})}{p(\mathbf{y})}. \tag{5-8}$$

Therefore, similar to the CP-ALS algorithm, a block coordinate descent algorithm can be applied by conditioning the posterior distribution of one factor matrix on all other factor matrices. This itterative algorithm will be called Bayesian CP-ALS which sequentially updates the factor matrices by using the update equations in equation (5-7). The updating equations incorporate a prior covariance and a prior mean of the d-th factor matrix. The Bayesian CP-ALS algorithm is formulated in Appendix A-2.

5-1-2 The unscented algorithm

In the Bayesian CP-ALS algorithm the posterior distribution $p(\mathbf{w}^{(d)})|\{\mathbf{w}^{(d)}\}_{\backslash d}, \mathbf{y})$ for each factor matrix is computed. The mean of each factor matrix can be used similar to the conventional CP-ALS to make predictions. However, with regards to uncertainty quantification, it is desirable to have an expression for the parameter covariance for the estimated parameters in the CPD. This would give the most plausible and complete picture of the uncertainty concerning the estimated parameters in the model. This is not provided by the Bayesian CP-ALS algorithm as only the conditional mean and covariance of each separate factor matrix are obtained. The estimate for the full parameter tensor $\boldsymbol{\mathcal{W}}$ is computed by means of a nonlinear function since the factor matrices together form the CPD. However, the low-rank tensor estimate $\boldsymbol{\mathcal{W}}$ is not Gaussian since the nonlinear function that computes the low-rank tensor estimate is dependent on the posterior distributions of the factor matrices.

The Unscented Transform (UT) is a numerical technique used for propagating probability distributions through nonlinear functions in order to approximate the distribution's mean $\mathbf{m}_{\text{UT}}$ and covariance $\mathbf{P}_{\text{UT}}$ [42, 34]. The UT approximates the transformation using a set of carefully chosen sample points, known as sigma points, and their corresponding weights. More specifically, the UT approximates the mean and covariance of a distribution that is a nonlinear function of a known distribution which is the distribution $\mathbf{h} \sim \mathcal{N}(\mathbf{m}, \mathbf{P})$, where the mean $\mathbf{m} \in \mathbb{R}^{D\hat{M}R \times 1}$ and covariance $\mathbf{P} \in \mathbb{R}^{D\hat{M}R \times D\hat{M}R}$ are build from the estimated means and covariances of the factor matrices as follows:

$$\mathbf{h} \sim \mathcal{N}(\mathbf{m}, \mathbf{P}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_D \end{bmatrix}, \begin{bmatrix} \mathbf{P}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{P}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{P}_D \end{bmatrix}\right). \tag{5-9}$$

Sigma points selection A set of sigma points around the mean of the given probability distribution are selected as follows:

$$\begin{aligned}
\mathbf{x}^{(0)} &= \mathbf{m}, \\
\mathbf{x}^{(i)} &= \mathbf{m} + \sqrt{D\hat{M}R + \tau}\left[\sqrt{\mathbf{P}}\right]_i, \quad i = 1, \ldots, D\hat{M}R, \\
\mathbf{x}^{(i+M)} &= \mathbf{m} - \sqrt{D\hat{M}R + \tau}\left[\sqrt{\mathbf{P}}\right]_i, \quad i = 1, \ldots, D\hat{M}R.
\end{aligned} \tag{5-10}$$

Here, $\sqrt{\mathbf{P}}$ denotes the Cholesky factor in order that $\sqrt{\mathbf{P}}\sqrt{\mathbf{P}}^T = \mathbf{P}$ and $\left[\sqrt{\mathbf{P}}\right]_i$ is the i-th column of the matrix $\sqrt{\mathbf{P}}$. Also, the scaling parameter can be defined as $\tau = \alpha_{\mathrm{UT}}^2(D\hat{M}R + \kappa) - D\hat{M}R$, where α is a scaling parameter that determines the spread of the sigma points and κ is another scaling parameter which ensures that the sigma points are symmetrically placed around the mean. The scaling parameter α is often set to a small positive value to avoid placing sigma points too close to the mean.

Propagation through nonlinear function Each sigma point is passed through the nonlinear function, and the resulting transformed points are obtained. First, the sigma points are reshaped into its corresponding factor matrices. Thus, for sigma points $\mathbf{x}^{(0)}$, $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(i+M)}$ where $i = 1, \ldots, D\hat{M}R$, the corresponding factor matrices have the form $\mathbf{W}_i^{(d)}$ where $d = 1, \ldots, D$ and $i = 0, \ldots, D\hat{M}R$. Then, the sigma points are propagated through the nonlinearity by reconstructing the vectorized low-rank tensor estimate as also described in equation (2-29):

$$\mathbf{s}^{(i)} = \mathrm{vec}(\boldsymbol{\mathcal{W}}_i) = \left(\bigodot_{d=D}^{1} \mathbf{W}_i^{(d)}\right)\boldsymbol{\lambda}, \quad i = 0, \ldots, 2D\hat{M}R. \tag{5-11}$$

where $\mathbf{s}^{(i)}$ are the i-th transformed sigma points.

Approximating the unscented mean and covariance The approximated mean $\mathbf{m}_{\mathrm{UT}}$ and covariance $\mathbf{m}_{\mathrm{UT}}$ can be computed as follows:

$$\begin{aligned}
\mathbf{m}_{\mathrm{UT}} &= \sum_{i=0}^{2M} w_i^{\mathbf{m}} \mathbf{s}^{(i)} \\
\mathbf{P}_{\mathrm{UT}} &= \sum_{i=0}^{2M} w_i^{\mathbf{P}} \left(\mathbf{s}^{(i)} - \mathbf{m}_{\mathrm{UT}}\right)\left(\mathbf{s}^{(i)} - \mathbf{m}_{\mathrm{UT}}\right)^T
\end{aligned} \tag{5-12}$$

where $w_i^{\mathbf{m}}$ and $w_i^{\mathbf{P}}$ are the weighting factors which are defined similarly as in [34] as follows:

$$\begin{aligned}
w_0^{\mathbf{m}} &= \frac{\tau}{D\hat{M}R + \tau}, \quad w_0^{\mathbf{P}} = w_0^{\mathbf{m}} + (1 - \alpha^2 + \beta), \\
w_i^{\mathbf{m}} &= w^{\mathbf{m}} = w_i^{\mathbf{P}} = w^{\mathbf{P}} = \frac{1}{2(D\hat{M}R + \tau)}, \quad i = 1, \ldots 2D\hat{M}R.
\end{aligned} \tag{5-13}$$

In [34] the, $\alpha = 0.001$, $\kappa = 3 - M$ and $\beta = 2$ are used as constants in the weighting factors. These values were chosen as these were suggested by literature [16, 42]. Ultimately, the unscented mean and covariance are used to predict test inputs. Hence, a particular test input $\mathbf{z}_* = \mathbf{z}(\mathbf{x}_{n*})$ is mapped to a feature map given in equation (3-3) and the feature map for all test inputs is stated as $\mathbf{Z}_* = \mathbf{Z}(\mathbf{X}_*)$. The mean and prediction error variance can be stated as follows:

$$\begin{aligned}\mathbb{E}(\hat{f}) &= \mathbf{Z}_*\mathbf{m}_{\mathrm{UT}},\\ \mathbb{V}(\hat{f}) &= \mathrm{diag}(\mathbf{Z}_*\mathbf{P}_{\mathrm{UT}}\mathbf{Z}_*^T).\end{aligned} \tag{5-14}$$

The unscented transform algorithm is given in Appendix A-3. The Bayesian method with the unscented transform is expensive from both a storage complexity and computational complexity. The unscented mean $\mathbf{m}_{\mathrm{UT}} \in \mathbb{R}^{M\times 1}$ and covariance $\mathbf{P}_{\mathrm{UT}} \in \mathbb{R}^{M\times M}$ scale exponentially in the number of features as $M = \hat{M}^D$, rendering a storage complexity of $\mathcal{O}(\hat{M}^{2D})$. Also, the prediction error variance for all test inputs $\mathbb{V}(\hat{f})$ is computed by multiplying the $\mathbf{P}_{\mathrm{UT}}$ from the left with $\mathbf{Z}_*$ and from the right with $\mathbf{Z}_*^T$ and taking the diagonal of the resulting matrix. The computational complexity for this is $\mathcal{O}(NM)$ which also scales exponentially in the number of features since $M = \hat{M}^D$. Furthermore, in the construction of $\mathbf{P}_{\mathrm{UT}}$ in the UT, the bottleneck is a series of outer products of $\mathbf{m}_{\mathrm{UT}}$ that are required. This has a computational complexity of $\mathcal{O}(D\hat{M}RM^2)$, or similarly, $\mathcal{O}(DR\hat{M}^{2D+1})$. Therefore, the computational complexity can quickly become large when the number of features increases. In addition, when the unscented covariance is constructed, computing the prediction error variance $\mathbb{V}(\hat{f})$ has a computational complexity of $\mathcal{O}(\hat{M}^D N_{\mathrm{test}})$.

5-2 Bayesian inference with single stochastic core

Another method of uncertainty quantification incorporates both Bayesian and frequentist aspects. This method only regards the stochasticity of one of the computed factor matrices and is based on [36]. This method is referred to as the SBC method. Firstly, the SBC method computes the factor matrices by means of the conventional CP-ALS algorithm. Using these factor matrices, a single Bayesian update step is performed. In this method, similar assumptions as noted in 5-1-1 apply.

First, it is important to note that the $\mathbf{C}$ matrix can also be defined by the following notation $\mathbf{Z}\mathbf{W}_{\backslash d}$. Here, $\mathbf{Z}$ denotes the training inputs that are mapped to the feature space and $\mathbf{W}_{\backslash d}$ denotes a matrix that is computed from all factor matrices except the d-th. This corresponds to the terms that are present in the matrix $\mathbf{C}$ in equation (3-26) where $\mathbf{Z}(\mathbf{X}^{(d)})$ is present for all dimensions $d = 1, \ldots, D$ and all factor matrices are present except for the d-th. The mean and covariance of the posterior distribution for one of the factor matrices $p(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\backslash d}, \mathbf{y})$ can be defined as:

$$\begin{aligned}\mathbb{E}(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\backslash d}, \mathbf{y}) &= \left[(\mathbf{P}_d^0)^{-1} + \frac{\mathbf{W}_{\backslash d}^T\mathbf{Z}^T\mathbf{Z}\mathbf{W}_{\backslash d}}{\bar{\sigma}_e^2}\right]^{-1}\left[\frac{\mathbf{W}_{\backslash d}^T\mathbf{Z}^T\mathbf{y}}{\bar{\sigma}_e^2} + (\mathbf{P}_d^0)^{-1}\mathbf{m}_d^0\right],\\ \mathbb{V}(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\backslash d}, \mathbf{y}) &= \left[(\mathbf{P}_d^0)^{-1} + \frac{\mathbf{W}_{\backslash d}^T\mathbf{Z}^T\mathbf{Z}\mathbf{W}_{\backslash d}}{\bar{\sigma}_e^2}\right]^{-1}.\end{aligned} \tag{5-15}$$

This posterior distribution can be thought of as a posterior distribution for a projected model since $\mathbf{Z}\mathbf{W}_{\backslash d}$ is a projection of the basis functions into a smaller subspace. In order to project the posterior distribution back to the original space, $p(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\backslash d}, \mathbf{y})$ is projected back to $p(\mathbf{w}|\mathbf{y})$ as follows:

$$
\begin{aligned}
\mathbb{E}(\mathbf{w}|\mathbf{y}) &= \mathbf{W}_{\backslash d}\mathbb{E}(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\backslash d}, \mathbf{y}) \\
\mathbb{V}(\mathbf{w}|\mathbf{y}) &= \mathbf{W}_{\backslash d}\mathbb{V}(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\backslash d}, \mathbf{y})\mathbf{W}_{\backslash d}^T.
\end{aligned}
\tag{5-16}
$$

In practice, the the computation for the mean and covariance in equation (5-16) is not computed explicitly. Instead, it is desired to compute the mean and variance for unseen test inputs $\mathbf{Z}_*$. Therefore, $\mathbf{Z}_*\mathbf{W}_{\backslash d}$ is computed efficiently by exploiting the structure of the matrices, which results in a matrix of size $N_{\text{test}} \times \hat{M}R$. More specifically, $\mathbf{Z}_*\mathbf{W}_{\backslash d}$ is computed by using equation (3-26) but instead of using the training inputs $\mathbf{X}$, the test inputs $\mathbf{X}_*$ are adopted. Using the estimated parameters and the test matrix in the $\mathbf{C}$ matrix is denoted by $\hat{\mathbf{C}}_*^{(d)}$. Therefore, the predictions $\mathbb{E}(\hat{f})$ and the total error variance $\mathbb{V}(\hat{f})$ can be stated as:

$$
\begin{aligned}
\mathbb{E}(\hat{f}) &= \mathbf{z}_*\mathbf{W}_{\backslash d}\mathbb{E}(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\backslash d}, \mathbf{y}) = \hat{\mathbf{C}}_*^{(d)}\left[(\mathbf{P}_d^0)^{-1} + \frac{\hat{\mathbf{C}}^T\hat{\mathbf{C}}}{\hat{\sigma}_e^2}\right]^{-1}\left[\frac{\hat{\mathbf{C}}^T\mathbf{y}}{\hat{\sigma}_e^2} + (\mathbf{P}_d^0)^{-1}\mathbf{m}_d^0\right], \\
\mathbb{V}(\hat{f}) &= \operatorname{diag}\left(\mathbf{Z}_*\mathbf{W}_{\backslash d}\mathbb{V}(\mathbf{w}^{(d)}|\{\mathbf{w}^{(d)}\}_{\backslash d}, \mathbf{y})\mathbf{W}_{\backslash d}^T\mathbf{Z}_*^T\right) = \operatorname{diag}\left(\hat{\mathbf{C}}_*^{(d)}\left[(\mathbf{P}_d^0)^{-1} + \frac{\hat{\mathbf{C}}^T\hat{\mathbf{C}}}{\hat{\sigma}_e^2}\right]^{-1}\hat{\mathbf{C}}_*^{(d)^T}\right).
\end{aligned}
\tag{5-17}
$$

The observation noise variance for the SBC method in an estimator $\hat{\sigma}_e^2$ for regression tasks and an hyperparameter $\bar{\sigma}_e^2$ in classification tasks. In section 6-2, this will further be elaborated on. The sole interest of the SBC method is the prediction error variance $\mathbb{V}(\hat{f})$ in equation (5-17). The prediction $\mathbb{E}(\hat{f})$ is not actually used in the SBC method as the obtained factor matrices by the conventional CP-ALS algorithm are used for making predictions. The difference in the computational complexity of the SBC method compared to the delta method depends on the computation and inversion of $\hat{\mathbf{C}}^T\hat{\mathbf{C}}$ which has a computational complexity of $\mathcal{O}((\hat{M}R)^2N + (\hat{M}R)^3)$. Then, by multiplying with $\hat{\mathbf{C}}_*^{(d)}$ from the left and $\hat{\mathbf{C}}_*^{(d)^T}$ from the right, and taking the diagonal of the resulting matrix, the prediction error variance for all test inputs $\mathbb{V}(\hat{f})$ is computed. Multiplying $\left[(\mathbf{P}_d^0)^{-1} + \frac{\hat{\mathbf{C}}^T\hat{\mathbf{C}}}{\hat{\sigma}_e^2}\right]^{-1}$ with $\hat{\mathbf{C}}_*^{(d)}$ from the left and its transpose from the right exhibits a computational complexity of $\mathcal{O}((\hat{M}R)^2N_{\text{test}})$, which is less compared to the delta method as $\hat{\mathbf{C}}_*^{(d)}$ is effectively a single term of the gradient of the model with respect to the factor matrices evaluated at the estimated parameters (equation (4-43)).

Chapter 6

Quantitative assessment and experiment results

The three uncertainty quantification methods that were investigated in this research are the delta method, the Bayesian method and the Single Bayesian Core (SBC). In order to compare the quality of the uncertainty intervals that can be constructed with each method, an assessment measure is needed. This results section will focus on confidence intervals (CIs) and prediction intervals (PIs). PIs are designed to provide a range of values in which a future observation is likely to fall with a specified level of confidence. This target of individual predictions is more intuitive and straightforward to interpret than estimating the uncertainty between the prediction and the underlying true function, provided by CIs, in regression tasks. This is because the underlying true function is unknown when quantifying the uncertainty of the predictions for a dataset. Moreover, when assessing the quality of PIs, their accuracy can be validated by comparing them to the actual outcomes of future observations for regression experiments on actual datasets. It can be calculated how often test inputs fall within the PI with the specified confidence level. This direct validation makes it easier to assess whether the prediction intervals are providing reliable uncertainty estimates.

In the experiments, the goal is to assess and compare the quality of the constructed CIs and PIs for the three different methods. The quality of the intervals is assessed from a correctness and informativeness perspective. Correctness relates to the capability of accurateness of the interval: does the interval approximately cover the underlying true function for CIs and the test inputs for PIs about 95% of the time. Here, 95% is the chosen confidence level. Ideally, the coverage of CIs and PIs should be close to the nominal confidence level. Furthermore, the quality of the intervals is assessed from an informativeness perspective, where the width is the point of focus.

6-1 Performance metrics

6-1-1 Model prediction performance measure

It is important to acquire models with high predictive power, besides the need for valuable and qualitative uncertainty quantification measures in the form of PIs. The model prediction performance can be evaluated by means of the mean squared error (MSE) in a regression task. Given either the prediction for the n-th test sample $f(\mathbf{x}_n, \hat{\mathbf{w}})$ or the mean predictive distribution $\mathbb{E}(\hat{f}_n)$ for the n-th test sample, the MSE can be stated as (here stated for $\mathbb{E}(\hat{f}_n)$):

$$\text{MSE} = \frac{1}{N} \sum_{n=1}^{N} (y_n - \mathbb{E}(\hat{f}_n)). \tag{6-1}$$

In a classification task, the model prediction performance can be evaluated by means of the misclassification rate (MR). The MR can be stated as:

$$\text{MR} = \frac{1}{N} \sum_{n=1}^{N} (y_n - \text{sign}(\mathbb{E}(\hat{f}_n))). \tag{6-2}$$

Here, y_n represents a variable that can take the values $\{-1, 1\}$ as in this research binary classification problems are considered.

6-1-2 CI and PI quality measure

This research employs a particular PI assessment measure that is used to assess the quality of a PI. This assessment measure is also used in the K-fold cross-validation which is used for hyperparameter tuning. The PI assessment criteria does not only provide a measure for the amount of test inputs that fall within the PI, but also takes into account the width of the PI.

The core characteristic of a PI is the prediction interval coverage probability (PICP) which is measured by counting the number of observations that are covered by the constructed PI. The PICP is given as follows:

$$\text{PICP} = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \pi_n \quad \text{where} \quad \pi_n = \begin{cases} 1, & y_n \in [L_n,\ U_n] \\ 0, & y_n \notin [L_n,\ U_n]. \end{cases} \tag{6-3}$$

Here, N_{test} are the number of samples in the test set and L_n and U_n are the lower and upper bound for the n-th constructed PI. L_n and U_n are computed by $\mathbb{E}(\hat{f}) - 2\sqrt{\mathbb{V}(\hat{f}) + \sigma_e^2}$ and $\mathbb{E}(\hat{f}) + 2\sqrt{\mathbb{V}(\hat{f}) + \sigma_e^2}$ respectively. Reasonable PIs should give a value that is near or larger than the confidence level with which the PIs are constructed. However, by increasingly widening the PIs, the PICP will increase. Hence, the quality of PIs can not only rely on the PICP as this would result in rewarding wide PIs that are in practice way to conservative and do not provide a useful uncertainty information. In that regard, the width of PIs should be taken into account and should be integrated in the uncertainty quantification measure. This is realized by the mean prediction Interval width (MPIW) which records to average width of PIs and can be stated as:

$$\text{MPIW} = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} (U_n - L_n). \tag{6-4}$$

The MPIW provides a quality measure for PIs and must be combined with PICP in order to take both coverage probability and PI width into account. It is desired that this new measure prioritizes PICP the most as this is the core feature of uncertainty quantification and it is very undesirable when a relatively low PICP is obtained, as this would mean that the theoretically obtained PIs are not correct. Incorporating both uncertainty quality assessment measures in a single measure is realized by the coverage width-based criterion (CWC). The CWC can be stated as follows:

$$\text{CWC} = \text{MPIW} + \gamma(\text{PICP})e^{-\eta(\text{PICP}-\mu)} \tag{6-5}$$

where $\gamma(\text{PICP})$ is defined as

$$\gamma = \begin{cases} 0, & \text{PICP} \geqslant \mu \\ 1, & \text{PICP} < \mu. \end{cases} \tag{6-6}$$

In this formulation, both η and μ are hyperparameters. These values have to be chosen in such a way that a lower coverage probability is penalized desirably. The μ hyperparameter can be set is set based on the confidence level α with which the constructed PIs are associated. More specifically, this value can be set to $1 - \alpha$. When increasing η, the overall penalty of PICP being lower than μ increases. The main principles that are desired are that the CWC should be large when PICP $< \mu$ regardless of the width of the PIs that is evaluated by MPIW and that MPIW should be the dominant factor of the CWC when PICP $> \mu$.

The same metrics can be adopted for CIs as the PICP, MPIW and CWC for PIs. The only difference will be to evaluate the confidence interval coverage probability (CICP) by means of the true function f^* instead of the test input y_n which is done for the PICP in equation (6-3). Thus, L_n and U_n in equation (6-3) and (6-4) are computed by $\mathbb{E}(\hat{f}_n) - 2\sqrt{\mathbb{V}(\hat{f}_n)}$ and $\mathbb{E}(\hat{f}_n) + 2\sqrt{\mathbb{V}(\hat{f}_n)}$ for the n-th prediction respectively. The CICP can exactly similar be stated for confidence intervals in regression tasks, only the evaluation is with respect to the true underlying function instead of an observation. Therefore, this method can only be accurately applied in synthetic experiments where the true underlying function is known.

In conclusion, the CWC assessment criteria compromises between informativeness and correctness with regards to PIs. Informativeness is enforced by the desire for PIs that are as narrow as possible. However, the narrowness could lead to a decrease in correctness as observations in the test set fall outside the PIs resulting in a low quality PIs as the coverage probability decreases. The CWC tries to compromise between the conflicting perspectives: informativeness and correctness [22].

The CWC assessment measure can similarly stated for a classification task. Here, the CICP measure for classification is used. The CICP for classification gives the proportion of rightly labelled test inputs to all test inputs. Only the predicted probabilities that fall outside the confidence intervals are taken into account. It is calculated by counting the number of rightly classified labels outside the CI (RI) and dividing it by the total number of test inputs, given by both rightly classified labels outside the confidence interval (RI) and the wrongly classified labels outside the confidence interval (WR).

$$\mathrm{CICP} = \frac{\mathrm{RI}}{\mathrm{RI} + \mathrm{WR}} \tag{6-7}$$

Ideally, the CICP should closely match the nominal confidence level $1 - \alpha$. For example, confidence intervals constructed with a 95% confidence level would ideally amount to 95% of the test inputs being rightly labelled outside the confidence intervals. For classifiers that have a very low classification rate, the CICP can be above $1 - \alpha$ as there are few or zero wrongly classified labels outside the CI.

Furthermore, the time it takes for the construction of the PIs and CIs, which is the uncertainty interval construction execution time, is also recorded for all experiments. This is done in order to provide a statement on which method could be preferably used when similar performance is achieved in terms of correctness and informativeness.

6-2 Hyperparameters and K-fold cross-validation

6-2-1 Hyperparameters for regression and classification

In regression tasks, the delta method incorporates a single hyperparameter which is the regularization hyperparameter λ_{reg}. In the delta method for regression tasks, the noise variance $\hat{\sigma}_e^2$ is estimated, and is therefore is not a hyperparameter. The Bayesian method has two hyperparameters which are the scaling parameter of the prior covariance ρ and the noise variance $\bar{\sigma}^2$. The SBC method incorporates both λ_{reg} and ρ. The noise variance $\hat{\sigma}_e^2$ is being estimated in the SBC method in regression tasks. In contrast, classifications tasks adopt an extra hyperparameter for the delta method and SBC method. This is because the noise variance can not be estimated, which is done for regression tasks, because it is assumed that the binary labels are strictly correct. However, setting the noise variance to zero would indicate zero noise inherent to the data resulting in zero aleatory uncertainty which is generally not plausible. In order to overcome this, the noise variance is set as a hyperparameter for the delta and SBC method, which is specified by a bar on top of the noise variance $\bar{\sigma}_e^2$. For the Bayesian method, the noise variance was already set as a hyperparameter. To keep the number of hyperparameters limited, the regularization parameter for the SBC method is set equal to the regularization parameter as in the delta method.

6-2-2 K-fold cross-validation

For all three methods, the hyperparameters are tuned by means of K-fold cross-validation. The goal of hyperparameter tuning is to find the optimal values for hyperparameters that result in the best performance of the model and the quality of the uncertainty quantification measure on test data. K-fold cross-validation plays a crucial role in this process by providing a robust and reliable way to estimate the model's performance for different hyperparameter configurations. The basic idea behind K-fold cross-validation is to divide the dataset into a training set and a test set. In turn, the test set is used to produce k subsets or folds of approximately equal size. The model is then trained and evaluated k times, with each fold serving as the test set once and the remaining $k-1$ folds used as the training data. The two evaluation criteria that are used are the mean squared error MSE and the coverage width-based criterion CWC for a regression task and the MR and CWC for a classification task. MSE and MR are employed in order to evaluate the fit of the model to the data and the CWC is exploited in order to assess and evaluate the uncertainty quantification part e.i. the PIs. The K-fold cross-validation procedure is given in Appendix A-1.

Cross-validation delta method For the three proposed uncertainty quantification methods for the CPD constrained kernel machine, different hyperparameters control the learning process. In the conventional CP-ALS with the delta method, the regularization hyperparameter λ_{reg} controls the learning process of the parameters and the shape of the prediction intervals, since this hyperparameter is included in the delta method. The noise variance σ_e^2 can be estimated with the expression for $\hat{\sigma}_e^2$ in equation (4-44). The following range for the regularization hyperparameter is used in the K-fold cross-validation procedure for regression tasks:

$$\lambda_{\text{reg}} \in [1^{-2},\ 1^{-3},\ 1^{-4},\ 1^{-5},\ 1^{-6},\ 1^{-7},\ 1^{-8}].$$

For classification tasks, the range for λ_{reg} is set similar. For the noise variance in classification tasks the following range of hyperparameters is chosen in the K-fold cross-validation procedure:

$$\bar{\sigma}_e^2 \in [0.01,\ 0.1,\ 0.5,\ 1,\ 10,\ 40,\ 80,\ 120].$$

Cross-validation Bayesian method In the Bayesian method, two hyperparameters play a part in the learning process. The first one being the ρ which is contained in the prior covariance $\mathbf{P}_d^0 = \rho\mathbf{I}$ for $d = 1, \ldots, D$. Hence, the prior for each factor matrix is set as a scaled identity term. By setting ρ to a relatively higher value, a lower certainty is implied on the prior mean and a lower value implies higher certainty on the prior mean. The second one being the noise variance $\bar{\sigma}^2$. Since the two hyperparameters directly influence the learning process in the Bayesian method, the shape of the PIs is also directly influenced. The following range for the scaling prior hyperparameter of the prior is used in the K-fold cross-validation procedure for regression tasks:

$$\rho \in [1^{-4},\ 1^{-3},\ 1^{-2},\ 1^{-1},\ 1,\ 5,\ 10,\ 50,\ 100].$$

Furthermore, the following range for the noise variance hyperparameter is used:

$$\bar{\sigma}_e^2 \in [0.001,\ 0.005,\ 0.01,\ 0.05,\ 0.1,\ 0.5].$$

For classification tasks, the range of the scaling prior hyperparameter is the same as the in regression with the Bayesian method and the noise variance in the K-fold cross-validation procedure is set the same as the noise variance as in classification for with the delta method.

Cross-validation SBC method In the SBC method, conventional CP-ALS is applied first, in turn, one of the estimated factor matrices is used for a single stochastic factor matrix update which constitutes this uncertainty quantification method. Hence, λ_{reg} and ρ are two hyperparameters that can be controlled in the procedure. First, the the regularization hyperparameter λ_{reg} is controls the learning process of the conventional CP-ALS and afterwards ρ and controls the Bayesian inference step for a single stochastic core. The noise variance σ^2 can be estimated in regression tasks with the expression for $\hat{\sigma}^2$ in equation (4-44). The same ranges for λ_{reg}, ρ and $\bar{\sigma}_e^2$ are used in the cross-validation procedure as in the delta method and Bayesian method respectively.

6-2-3 Choosing the hyperparameters

In each method the hyperparameters are chosen similarly. That is, the hyperparameters that produce the minimum mean MSE are collected. More specifically, the hyperparameters that produce MSE values that are up to a maximum percentage p higher than the minimum MSE value produced by one set of hyperparameters. Then, across the selected hyperparameter values, the value with the smallest CWC is chosen to be the hyperparameter in the regular learning procedure. Similarly, this K-fold cross-validation technique is applied in a classification task which employs MR and CWC.

6-2-4 Data preprocessing

In all simulations in this research the data is preprocessed. For regression and classification tasks, input data preprocessing is the same. Input data is scaled to the range $[0, 1]$ for each feature in the following way:

$$\mathbf{X}_{\text{scale}}^{(d)} = \frac{\mathbf{X}^{(d)} - \min(\mathbf{X}^{(d)})}{\max(\mathbf{X}^{(d)}) - \min(\mathbf{X}^{(d)})}. \tag{6-8}$$

Here, $\mathbf{X}_{\text{scale}}^{(d)}$ is the preprocessed input data for feature d. The corresponding output data $\mathbf{y}$ is also preprocessed. In a regression task, the continuous output values are normalized as follows,

$$\mathbf{y}_{\text{scale}} = \frac{\mathbf{y} - \bar{\mathbf{y}}}{\sigma_{\mathbf{y}}} \tag{6-9}$$

where $\bar{\mathbf{y}}$ and $\sigma_{\mathbf{y}}$ are the mean and variance of $\mathbf{y}$ respectively. In a classification task, the binary output labels are set to be -1 and 1.

6-3 Illustrative comparison of uncertainty intervals

6-3-1 Uncertainty quantification on 1-D synthetic data

In this simulation, the three uncertainty quantification methods are tested on one-dimensional (1-D) data. This is because, 1-D data allows for convenient visualization of the uncertainty intervals. The Canonical Polyadic Decomposition (CPD) is a tensor decomposition method that aims to factorize multi-dimensional data arrays into a sum of simpler components. It is particularly useful for capturing multi-modal relationships and interactions in data with multiple modes or dimensions. Therefore, applying the convention or Bayesian Canonical Polyadic Decomposition Alternating Linear Scheme (CP-ALS) algorithm on 1-D data is not meaningful as the summation of rank-1 tensors in the CPD would amount to the summation of vectors. Hence, the 1-D data is lifted to higher order data by utilizing quantized features.

Simulated data from the magic tyre formula [38] is being estimated by the CPD constrained kernel machine for the 1-D synthetic data experiment. The magic tyre formula is used for the analysis of tyre behavior in different driving situations. The magic tyre formula is stated as:

$$f(\mathbf{x}_n, \mathbf{w}) = w_1 \sin\left(w_2 \arctan\left((1-w_3)\mathbf{x}_n + w_3/w_4 \arctan(w_4\mathbf{x}_n)\right)\right) \tag{6-10}$$

where $\mathbf{x}_n$ is the measured wheel slip at time n and arbitrary values are chosen for the parameters to get a desired shape. However, the magic tire formula does not align with the structural framework of the CPD kernel machine i.e. $f^* \notin \mathcal{M}$. Consequently, the actual data that is used in this experiment is the data that estimates the simulation data of the magic tyre formula as this data generated by the CPD constrained kernel machine and must therefore belong to the model structure. Thus, this reference model has the property that it practically belongs to the model structure $f^{\text{ref}} \in \mathcal{M}$. Next, the estimation data $\mathbf{y}^{\text{truth}}$, viewed as the output from the reference model, is considered to be the ground truth in the following simulation. Thus,

$$\mathbf{y} = f^* + \mathbf{e}, \quad \mathbf{e} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) \tag{6-11}$$

where the true function vector is $f^* = [f^*(\mathbf{x}_1)\ f^*(\mathbf{x}_2)\ \dots\ f^*(\mathbf{x}_N)]^T = [f_1^*\ f_2^*\ \dots\ f_N^*]^T$ and $\mathbf{e}$ is a realization of random noise in which the samples are uncorrelated. The signal-to-noise ratio (SNR) is given by:

$$\text{SNR}_{\text{dB}} = 10\log_{10}\frac{||f^*||^2}{||\mathbf{e}||^2} \tag{6-12}$$

and is set to 20dB. The following settings are set exactly equal for the three methods: quantized pure power polynomial features, normally distributed random initialization of the factor matrices, random permutations in the data, $\hat{M} = 8$, $R = 10$, $\eta = 50$, $p = 0.5$, $\mu = 95$, the number of sweeps is 10, all training data is used for cross-validation and the initialization of the factor matrices in the CP-ALS and Bayesian CP-ALS is equal. Then, confidence intervals capture the uncertainty arising from sampling variability or measurement error in the data, which are the estimation errors, and the CIs are plotted in Figure 6-1 for all three methods.

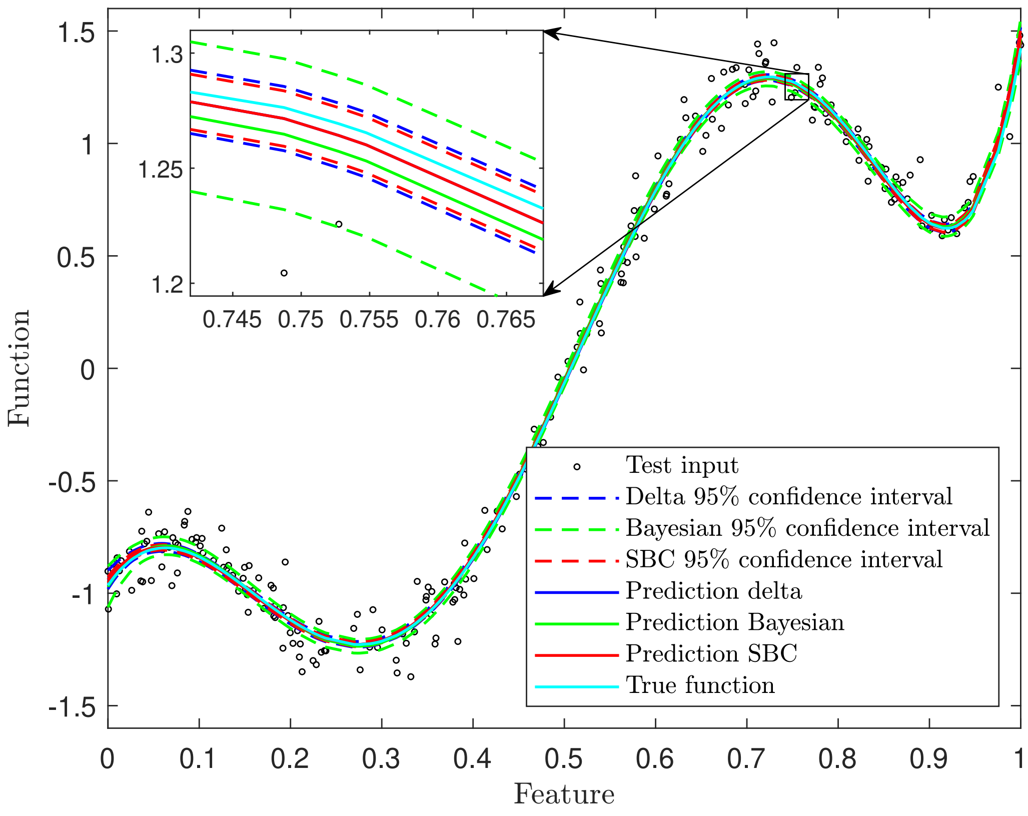


Figure 6-1: Regression with confidence intervals for the delta, Bayesian and SBC methods on 1-D synthetic data

In Figure 6-1, the confidence intervals should desirably bound the true underlying model with about 95% of the time which is approximated by a $2\mathbb{V}(\hat{f})$ confidence interval. For all three methods, this is criteria met. It can be observed that the delta and SBC intervals are very similar and the Bayesian interval is significantly wider than the delta and SBC intervals. Furthermore, the same experiment can be repeated for the construction of prediction intervals for the three methods. Besides the estimation error, the prediction intervals incorporate the estimated observation noise. The prediction intervals along with the prediction of each method is plotted in figure 6-2.

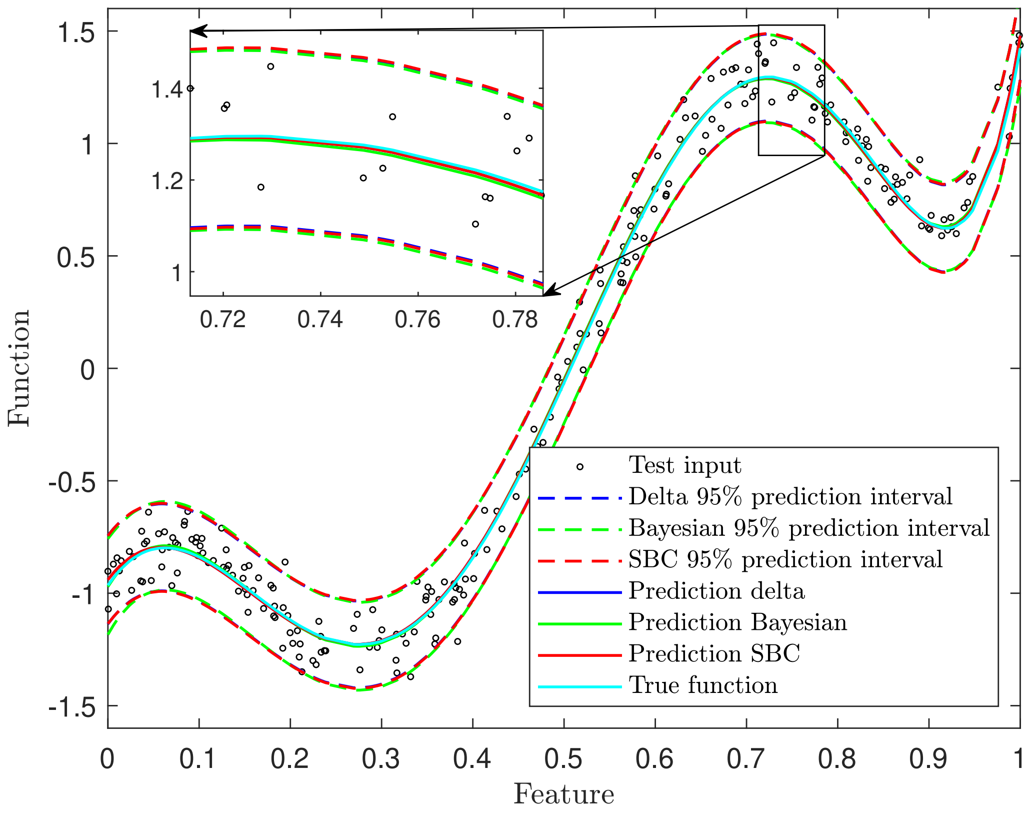


Figure 6-2: Regression with prediction intervals for the delta, Bayesian and SBC methods on 1-D synthetic data

It can be observed that for prediction intervals the difference between the Bayesian method compared to the delta and SBC method is smaller. Since the standard deviation of the estimated test inputs is defined as $\mathbb{V}(f_*) = \mathbb{V}(\hat{f}) + \sigma_e^2$, the smaller difference in this example mainly due to the magnitude of the observation noise compared to the estimation error.

6-3-2 Uncertainty quantification on 1-D dataset

It is valuable to perform an experiment on actual 1-D dataset in order to visualize how the different methods differ in their uncertainty intervals. The quality of confidence intervals is hard to assess when constructed for predictions for an actual dataset as the underlying true function is unknown. However, prediction intervals can be assessed due to their nature in which test inputs should be bounded. In this regard, prediction intervals will be focused on in regression tasks. Regression on a climate dataset is performed and prediction intervals are constructed with each of the three methods. This climate dataset represents the global land and surface temperature in July from 1850 till 2023 and was downloaded from NOAA National Centers for Environmental information [37]. In this experiment, $N = 174$ and $D = 1$ and 70% from the data is allocated to training data and 30% is used for testing. The results are plotted in figure 6-3.

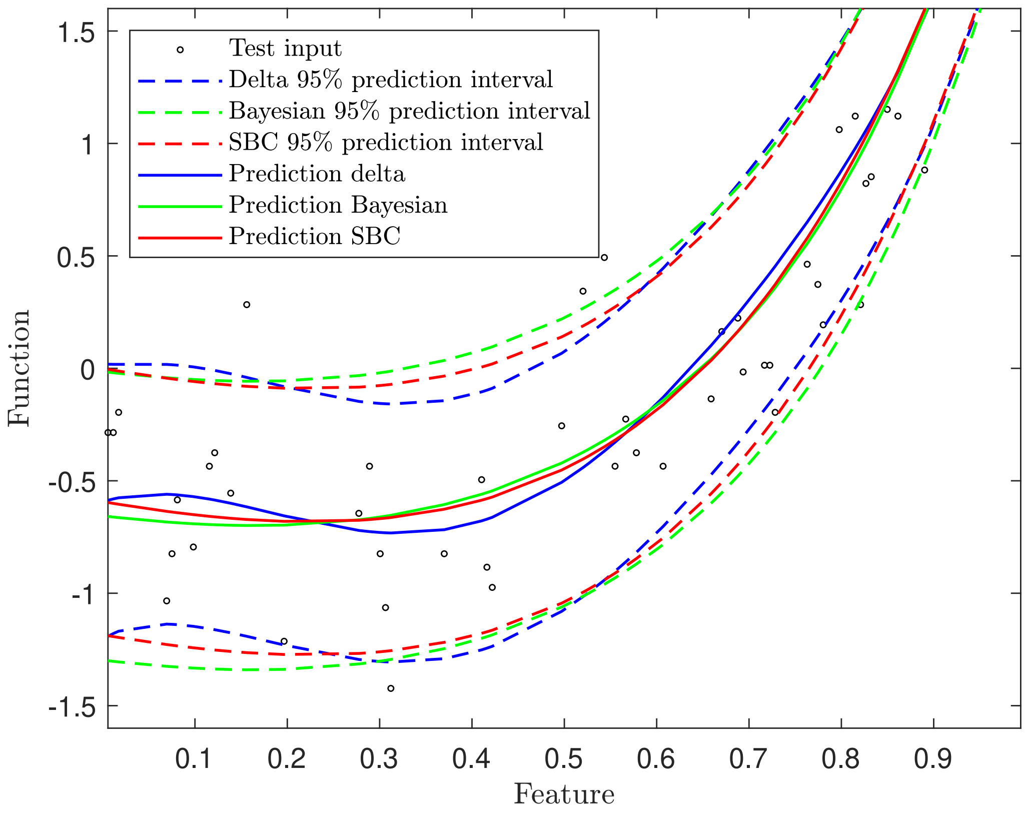
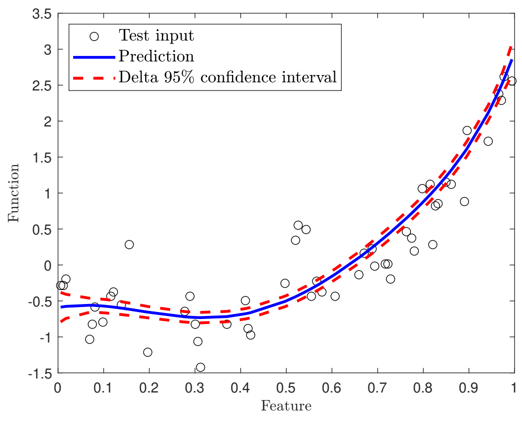
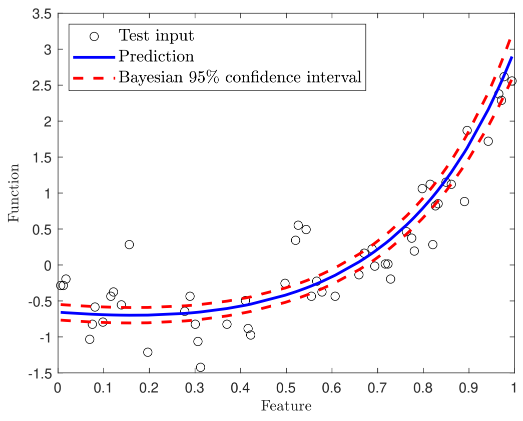


Figure 6-3: Regression with prediction intervals for the delta, Bayesian and SBC methods on 1-D climate dataset

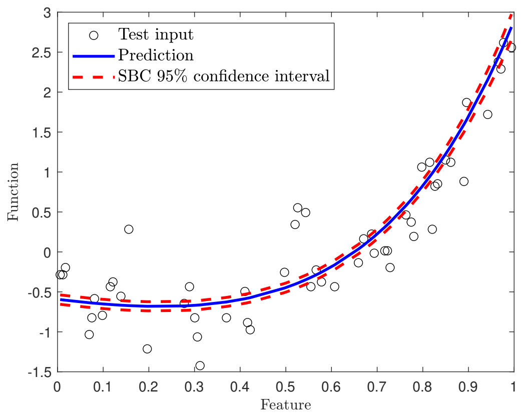
Again, the delta method and SBC method show very similar prediction intervals and the Bayesian method is somewhat more conservative compared to the delta and SBC method. Besides prediction intervals on the 1-D climate dataset, it is interesting to see what is the sole contribution of the prediction error variance $\mathbb{V}(\hat{f})$. Confidence intervals are only concerned with the prediction error variance. Confidence intervals are constructed with the delta, Bayesian and SBC method and are plotted in figure 6-4.



(a) Delta



(b) Bayesian

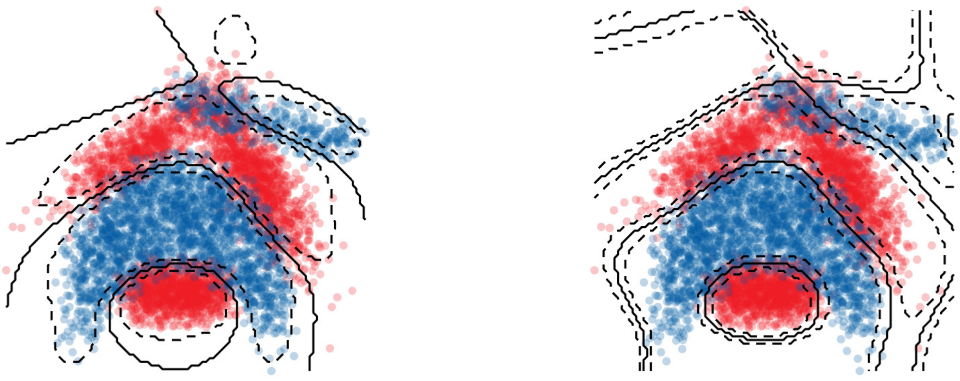


(c) SBC

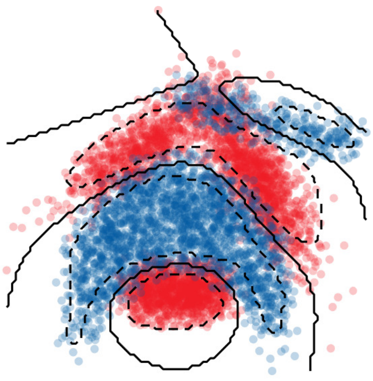
Figure 6-4: Prediction with confidence intervals for the delta, Bayesian and SBC methods on 1-D climate dataset

6-3-3 Confidence intervals for 2-D classification tasks

Confidence intervals can directly be assessed on their quality in classification tasks. The confidence intervals are interpreted as follows: every point within the two outer bands, the classifier casts doubt with significance level α on its label. In other words, the classifier is confident with confidence level $1 - \alpha$ on its label. For a $2\mathbb{V}(\hat{f})$ confidence interval, this significance level α is about 5%. It is important to emphasize how the PICP measure works in a classification task. A maximum of significance level α of miss-classified test inputs of all test inputs that fall outside the respective confidence interval would be allowed. Classifiers with confidence intervals for the three methods are visualized for two classification datasets. Firstly, the banana dataset ($N = 5300$ and $D = 2$) is used in figure 6-5.



(a) Delta **(b)** Bayesian

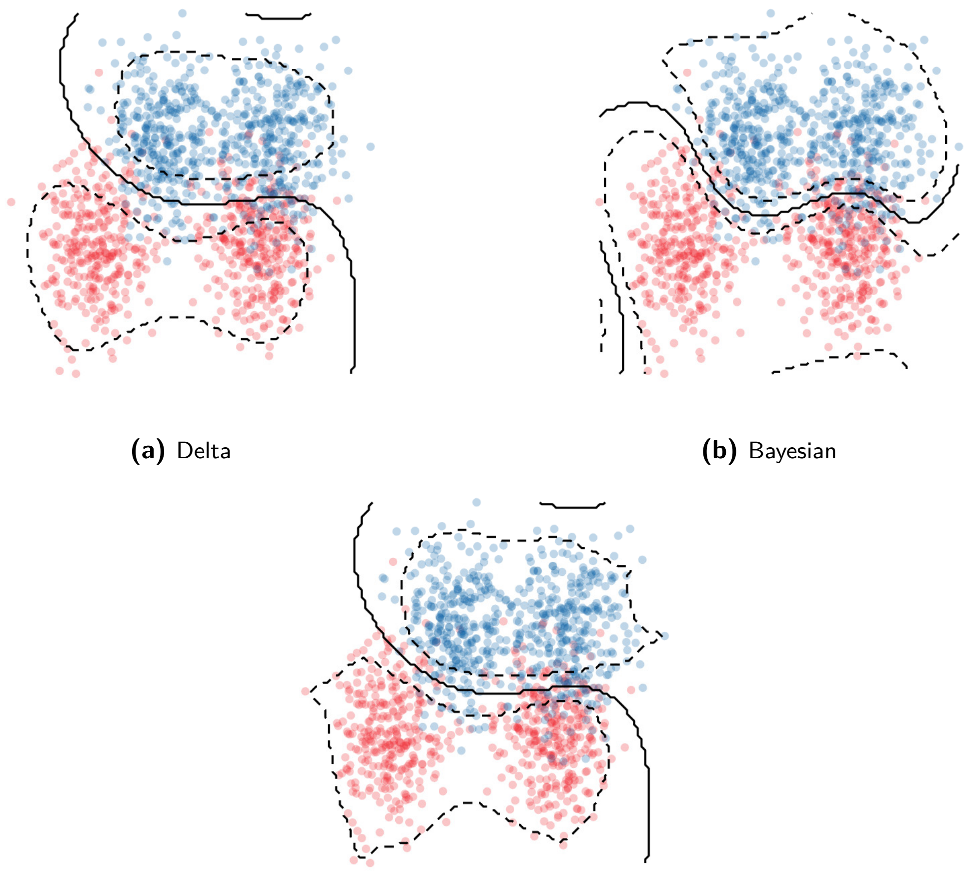


(c) SBC

Figure 6-5: Classifier with confidence intervals for the delta, Bayesian and SBC methods on 2-D banana dataset. The dashed lines represent the 95% confidence interval on the classifier. The solid line represents the classifier. The classifier casts doubt on the labels with confidence level 95% between the solid line and the dashed lines. The classifier is certain on the labels with confidence level 95% between the dashed lines.

From this visualization it can be observed that the three different methods produce somewhat different CIs. More specifically, the CIs differ especially in their coverage of the labels and their widths, thereby being more conservative or optimistic. A conservative CI is wider than necessary to ensure a specified level of confidence. On the other hand, optimistic CIs aim to provide a narrower range, allowing for higher precision but with the possible cost of underestimating the uncertainty. From Figure 6-5, it can be observed that the Bayesian method is most optimistic as it provides the widest intervals (most space and highest coverage between the dashed lines) and as a result covers more labels than the delta and SBC method. The delta method is most conservative as it provides the narrowest CIs (least space and least coverage between dashed lines) resulting in less labels being covered compared to Bayesian and SBC.

Another visualization of the confidence intervals is carried out where the ripley dataset ($N = 1250$ and $D = 2$) is used. The results of this experiment are visualized in figure 6-6.


(a) Delta (b) Bayesian

(c) SBC

Figure 6-6: Classifier with confidence intervals for the delta, Bayesian and SBC methods on 2-D ripley dataset. The dashed lines represent the 95% confidence interval on the classifier. The solid line represents the classifier. The classifier casts doubt on the labels with confidence level 95% between the solid line and the dashed lines. The classifier is certain on the labels with confidence level 95% between the dashed lines.

Similar to the banana dataset, it can be observed that the CI constructed with the delta method is the narrowest (providing the least space between the dashed lines), yielding the most conservative method. The Bayesian method provides the most optimistic intervals (providing the most space between the dashed lines). The SBC method is less optimistic than the Bayesian method and less conservative than the delta method.

6-4 Quantitative assessment of uncertainty quantification on synthetic data

These experiments are conducted in order to quantitatively investigate which method provides the most accurate uncertainty quantification in terms of correctness and informativeness. In this experiments with synthetic data, the underlying true function is known. Therefore, CIs are focused on as they provide an uncertainty measure that captures the deviation of the prediction relative to the true function which is caused by estimation error. In this regard, the CICP and mean confidence Interval width (MCIW) metrics are used.

6-4-1 Experiment settings and hyperparameters

6-4-2 Synthetic 3-D data experiment

Similar to the 1-D synthetic data experiment, data is generated from which is it is known that assumptions such as the noise variance being Gaussian distributed and independent hold. Moreover, the assumption that the true system is in the model structure ($f^* \in \mathcal{M}$), holds as the underlying true function is generated to be data from a CPD constrained kernel machine. This is done as follows: three factor matrices $\mathbf{W}^{(d)}$, $d = 1, 2, 3$ are constructed by randomly sampling from a Gaussian distribution with mean zero and a standard deviation of 1. Each factor matrix is assigned a size of $\hat{M} \times R$. Arbitrary input data is constructed creating a regularly spaced grid of input values. Then, data is generated by

$$\mathbf{f} = \left(\mathbf{Z}(\mathbf{X}^{(1)})^T\mathbf{W}^{(1)} * \mathbf{Z}(\mathbf{X}^{(2)})^T\mathbf{W}^{(2)} * \mathbf{Z}(\mathbf{X}^{(3)})^T\mathbf{W}^{(3)}\right)\mathbf{1}_R$$

which is the prediction that is also provided in equation (3-16) for each sample individually and the norms $\boldsymbol{\lambda}$ are equal to $\mathbf{1}_R$ as the norms are stored in the first factor matrix. Gaussian distributed noise is added with a SNR of 10 dB. Ultimately, 4000 samples were artificially constructed, where 70% was used for training and 30% was used for testing. 40% of the training data was used in the K-fold cross-validation procedure.

Table 6-1: Results 3-D synthetic experiment for the delta method with 95% nominal confidence level

Run	**Delta**						
	CICP	**MCIW**	**CWC**	**MSE**	**Time (s)**	λ_{reg}	$\hat{\sigma}_e^2$
1	1	0.1014	0.1014	0.0969	0.2884	1^{-4}	0.0997
2	0.9425	0.0900	1.5450	0.1048	0.2999	1^{-3}	0.0994
3	0.9650	0.0924	0.0924	0.0963	0.3332	1^{-4}	0.1004
4	0.9925	0.1015	0.1015	0.1029	0.3648	1^{-4}	0.0991
5	0.9663	0.1014	0.1014	0.1077	0.3077	1^{-4}	0.0986
6	0.9100	0.0926	7.4816	0.1057	0.3068	1^{-3}	0.0991
7	0.9588	0.0911	0.0911	0.0946	0.3197	1^{-3}	0.1006
8	0.9563	0.0908	0.0108	0.0912	0.3646	1^{-3}	0.1009
9	0.9788	0.1181	0.1181	0.0944	0.3331	1^{-6}	0.0097
10	0.9363	0.0895	2.0782	0.1042	0.3185	1^{-3}	0.0995
Mean	0.9606	0.0969	1.1802	0.0999	0.3237	N/A	0.0997

Table 6-2: Results 3-D synthetic experiment for the Bayesian method with 95% nominal confidence level

Run	**Bayesian**						
	CICP	**MCIW**	**CWC**	**MSE**	**Time (s)**	ρ	$\bar{\sigma}_e^2$
1	0.9925	0.1314	0.1314	0.0971	0.3638	0.1	0.1
2	0.9913	0.1300	0.1300	0.1049	0.2851	0.1	0.1
3	0.9713	0.1002	0.1002	0.0962	0.2766	0.01	0.1
4	0.9900	0.1286	0.1286	0.1035	0.2993	0.1	0.1
5	0.9850	0.1318	0.1318	0.1078	0.3165	0.1	0.1
6	0.9713	0.1416	0.1416	0.1058	0.2993	0.1	0.1
7	0.9800	0.1019	0.1019	0.0946	0.3020	0.01	0.1
8	0.9913	0.1469	0.1469	0.0915	0.2785	1	0.1
9	0.9988	0.1308	0.1308	0.0941	0.2791	0.1	0.1
10	0.9813	0.1257	0.1257	0.1039	0.2797	0.1	0.1
Mean	0.9853	0.1269	0.1269	0.0999	0.2980	N/A	N/A

Table 6-3: Results 3-D synthetic experiment for the SBC method with 95% nominal confidence level

Run	**SBC**							
	CICP	**MCIW**	**CWC**	**MSE**	**Time (s)**	λ_{reg}	ρ	$\hat{\sigma}_e^2$
1	0.9363	0.1035	2.0922	0.0969	0.2495	1^{-2}	50	0.0992
2	0.9800	0.1015	0.1015	0.1048	0.4038	1^{-2}	100	0.0985
3	0.9550	0.1024	0.1024	0.0963	0.2875	1^{-2}	100	0.0995
4	0.9575	0.1033	0.1033	0.1029	0.2622	1^{-2}	100	0.0987
5	0.9388	0.1025	1.8576	0.1077	0.2690	1^{-2}	100	0.0982
6	0.9213	0.1037	4.3139	0.1057	0.2759	1^{-2}	100	0.0981
7	0.9638	0.1006	0.1006	0.0946	0.3205	1^{-2}	100	0.0997
8	0.9550	0.1047	0.1047	0.0912	0.2591	1^{-2}	100	0.0999
9	0.9725	0.1043	0.1043	0.0944	0.2519	1^{-2}	100	0.0997
10	0.9275	0.1014	3.1817	0.1042	0.2747	1^{-2}	100	0.0988
Mean	0.9507	0.1028	1.2062	0.0999	0.2854	N/A	N/A	0.0990

From these results it can be observed that the average CICP for both the delta method (0.9606) and the SBC method (0.9505) matches well with the 95% nominal confidence level across all runs. On the other hand, the CICP of the Bayesian method (0.9853) is on average significantly larger with higher variability across the different runs. The coverage of the Bayesian is essentially too great for what is asked by the nominal confidence level which does not necessarily mean low quality CIs or PIs. However, higher coverage probability does generally result in a higher uncertainty interval width based measure, rendering less informative CIs and PIs. The MCIW is on average considerably higher for the Bayesian method (0.1269) compared to the delta method (0.0969) and SBC method (0.1028), which are very similar. As a supplementary point, the mean uncertainty interval construction execution

time is 0.3237 for the delta method, 0.2980 for the Bayesian method and 0.2854 for the SBC method.
In this experiment, where the assumptions of Gaussian noise, independence and the true system being in the model structure, the delta method and SBC method show to be valuable and convincing uncertainty quantification methods.

6-4-3 Synthetic 8-D data experiment

This experiment is performed in order to provide an extra experiment of the quality of CIs on synthetically generated data. In this experiment, higher-dimensional synthetic data is considered, therefore, the Bayesian method will not be involved in the experiment as the unscented mean and covariance in the Bayesian method scale exponentially in their storage complexities. Again, the experiment is set up such that the noise variance being Gaussian distributed and being independent hold. Moreover, the assumption that the true system is in the model structure ($f^* \in \mathcal{M}$) is guaranteed by generated data directly stemming from the CPD constrained kernel machine.
First, synthetic input data is generated arbitrarily: two exponential growth functions, three exponential decay functions, a squared sine function, two quadratic functions, each with different ranges and the same number of samples ($N = 2000$), to construct a dataset with 8 features. Then, 8 random factor matrices are constructed, each with size $\hat{M} \times R$. where $\hat{M} = 20$ and $R = 10$. Output data is generated with the input data and factor matrices. In turn, Gaussian distributed noise is added to the output data with SNR $= 10$. The constructed dataset is partitioned in 80% training data and 20% test data. In the K-fold cross-validation procedure 40% of the training data is used. Furthermore, the nominal confidence level is set at 95% which is approximated by two standard deviations. The obtained results are stated in Table 6-4 and 6-5.

Table 6-4: Results 8-D synthetic experiment for the delta method with 95% nominal confidence level

Run	**Delta**						
	CICP	**MCIW**	**CWC**	**MSE**	**Time (s)**	λ_{reg}	$\hat{\sigma}_e^2$
1	1	0.0938	0.0938	0.0983	1.0319	1^{-2}	0.1001
2	0.9375	0.0989	1.9672	0.1048	1.0645	1^{-3}	0.0980
3	0.9650	0.0920	0.0920	0.1042	1.0382	1^{-2}	0.0982
4	0.9175	0.1010	5.1794	0.0995	1.0474	1^{-3}	0.0992
5	0.9700	0.0991	0.0991	0.1038	1.1054	1^{-4}	0.0983
6	0.9463	0.1003	1.3066	0.1028	1.1583	1^{-3}	0.0987
7	0.8975	0.1144	13.9190	0.1015	1.0089	1^{-6}	0.0986
8	0.9625	0.0945	0.0945	0.1132	1.0418	1^{-2}	0.0963
9	0.9663	0.0914	0.0914	0.0984	1.2635	1^{-2}	0.0998
10	0.9750	0.0944	0.0944	0.0982	1.4061	1^{-2}	0.1000
Mean	0.9537	0.0980	2.2937	0.1025	1.1166	N/A	0.0987

Table 6-5: Results 8-D synthetic experiment for the SBC method with 95% nominal confidence level

Run	**SBC**							
	CICP	**MCIW**	**CWC**	**MSE**	**Time (s)**	λ_{reg}	ρ	$\hat{\sigma}_e^2$
1	0.9550	0.1155	0.1155	0.0983	0.7420	1^{-2}	10	0.0996
2	0.9650	0.1300	0.1300	0.1048	0.8326	1^{-2}	100	0.0977
3	0.9538	0.1337	0.1337	0.1042	0.8284	1^{-2}	100	0.0975
4	0.9275	0.1291	3.2093	0.0995	0.9376	1^{-2}	100	0.0988
5	0.9850	0.1285	0.1285	0.1038	0.7528	1^{-2}	50	0.0979
6	0.9375	0.1383	2.0065	0.1028	0.9186	1^{-6}	50	0.0980
7	0.8950	0.1462	15.7889	0.1015	1.0824	1^{-8}	100	0.0980
8	0.9413	0.1307	1.6795	0.1132	1.3671	1^{-3}	100	0.0956
9	0.9275	0.1332	3.2134	0.0984	1.2778	1^{-2}	100	0.0991
10	0.9438	0.1403	1.5071	0.0982	0.7857	1^{-8}	100	0.0993
Mean	0.9431	0.1326	2.7913	0.1025	0.9525	N/A	N/A	0.0981

From these results can be concluded that both the delta method and the SBC method are well able to approximate the 95% nominal confidence level as the average CICP is 0.9537 and 0.9431 for the delta and SBC method respectively. Thereby, from a correctness perspective, both methods produce accurate CIs. Furthermore, the average MCIW is somewhat higher for the SBC method (0.1326) compared to the delta method (0.0980). Therefore, the delta method is able to construct CIs of slightly better quality from an informativeness perspective. The results from the 3-D synthetic data experiment and the 8-D synthetic data experiment are in agreement with each other. Additionally, the mean uncertainty interval construction execution time is 1.1166 for the delta method and 0.9525 for the SBC method.

6-5 Quantitative assessment of uncertainty quantification on datasets

In this section, six experiments are conducted which each provide quantitative results. The six experiments are collected from the UCI Machine Learning Repository. The experiments consist of three regression tasks and three binary classification tasks. The exact datasets that are used with the number of samples N and number of features D are listed in table 6-6.

Table 6-6: Dataset Information

Case Study	**Dataset**	**Samples**	**Attributes**	**Task**
1	Power plant	9568	4	Regression
2	Airfoil	1503	5	Regression
3	Concrete	1030	8	Regression
4	Ripley	1250	2	Classification
5	Banknote	1372	4	Classification
6	Raisin	900	7	Classification

6-5-1 Experiment settings and hyperparameters

In the experiments, different hyperparameters are set to be equal for the delta, Bayesian and SBC methods. The hyperparamter p is set at 0.5 for all experiments, which means that in the K-fold cross-validation procedure only the set of hyperparameters is taken into account that correspond to a MSE or MR that is 50% above the minimum MSE or MR that is computed. In this way, the set of hyperparameters that produce high MSE or MR are omitted. Furthermore, the confidence level $\mu = 1 - \alpha$ where α is the significance level, is set to 95%, yielding $\alpha = 0.05$. Furthermore, the hyperparameter η, which determines the dominance of the penalty term that arises when the PI coverage probability is less than μ, is set to 50. A value of 50 strikes a certain balance between the penalty arising from the width of the interval (MPIW) and PICP such that none of the penalty terms becomes way too dominant. However, a value of 50 will still greatly penalize PIs which have a lower PICP than 50% In the K-fold cross-validation process, 5 folds are used in each experiment. Furthermore, only a certain percentage of the entire training set is used for cross-validation. This percentage differs for each experiment and is chosen mainly on the size of the training dataset. The exact choices for hyperparameters and other experiment settings are summarized in table 6-7.

Table 6-7: Experiment choices

Case Study	$\hat{M}$	R	p	μ	η	K	**Cross-validation**(%)	**Feature**
1	8	10	0.5	0.95	50	5	10	Polynomial
2	4	20	0.5	0.95	50	5	20	Polynomial
3	3	10	0.5	0.95	50	5	60	Polynomial
4	12	10	0.1	0.95	50	5	60	Polynomial
5	6	10	0.1	0.95	50	5	60	Polynomial
6	3	10	0.1	0.95	50	5	100	Polynomial

A notable choice for one of the hyperparameters in table 6-7 is the choice for the dimension of the feature map $\hat{M}$. It would make more sense to increase $\hat{M}$ as the number of features in the dataset D increases. However, the Bayesian method becomes infeasible as the covariance matrix $\mathbf{P}_{\text{UT}}$ has size $I^D \times I^D$, which is the covariance matrix of the full weight tensor and thereby it scales exponentially in the number of features. For example, for a dataset with 8 features, $\mathbf{P}_{\text{UT}}$ has size $10^8 \times 10^8$ which is not practically feasible for this research. For the purpose of the analysis of the three methods, $\hat{M}$ is kept low such that a comparison with the Bayesian method can still be made.

6-5-2 Results on datasets

In table 6-8, the best CWC (CWC_{Best}) of the 10 runs for each experiment and for each method is noted. Furthermore, CWC_{Med} and CWC_{SD} are the median CWC and the standard deviation of the CWC values respectively.

Table 6-8: CWC metrics for each experiment

Case	**Delta**			**Bayesian**			**SBC**		
Study	CWC_{Best}	CWC_{Med}	CWC_{SD}	CWC_{Best}	CWC_{Med}	CWC_{SD}	CWC_{Best}	CWC_{Med}	CWC_{SD}
1	0.9276	**0.9304**	**0.0043**	**0.9042**	1.2665	0.4324	0.9385	0.9492	0.0045
2	**1.6513**	3.2381	2.6188	2.8639	**2.8746**	**0.0766**	1.8613	3.2622	1.7547
3	1.3983	4.6618	8.2224	1.3967	**2.9126**	**0.4821**	**1.3896**	2.9840	2.1301
4	1.0607	**1.3735**	**0.5692**	0.6575	1.9477	2.5997	**0.6316**	1.4674	1.7751
5	0.0674	0.0720	**0.0037**	0.0925	0.0978	0.0097	**0.0408**	**0.0471**	0.0071
6	1.7480	3.4022	3.8523	**1.1585**	4.1216	3.2383	1.2740	**1.6873**	**2.1514**

It can be observed from Table 6-8 that for regression tasks (experiment 1-3), each method obtains the best CWC_{Best} once. Furthermore, delta method attains the lowest $\text{CWC}_{\text{Median}}$ and CWC_{SD} once and the Bayesian method two times. Also, the results from the delta method are considerably similar to the results of the SBC method. For the classification tasks (experiment 4-6), the delta and SBC method method perform best on almost all CWC metrics for each experiment. Most notable is the SBC method which attains the lowest CWC_{Best} and $\text{CWC}_{\text{Median}}$ twice.

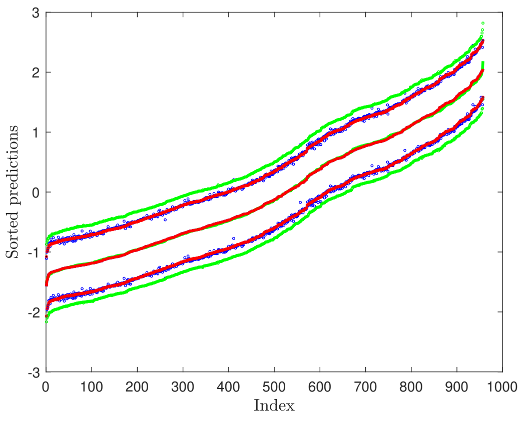
In order to draw conclusions on the proposed methods, it needs to be investigated how the CWC metrics are composed. The PICP, MPIW, CWC, MSE or MR, time of PI construction and the hyperparameters for all runs, experiments and methods are presented in Appendix B.

Power plant In this regression experiment, a few important elements can be observed. The Bayesian method provides the most competitive CWC_{Best} measure, whereas the delta method provides the best CWC_{Med} measure. By analyzing the Tables in B-1, it can be observed that the average empirical PICP, which is 0.9631, 0.9731 and 0.9639 for the delta, Bayesian and SBC method respectively, matches well with the nominal confidence level of 95 %. This is because, the average PICP is slightly greater than the nominal confidence level which causes the CWC to be equal to the MPIW, as the PIs are not penalized for their coverage probability when PICP $> 1-\alpha$. With regards to the MPIW, the methods perform equally well, however, the MPIW is somewhat more variable across the different runs for the Bayesian method which is not desirable. The mean MSE of the Bayesian method is only slightly higher compared to delta and SBC. As a supplementary point, the mean uncertainty interval construction execution time is 1.1832 for the delta method, 2.0965 for the Bayesian method and 0.9396 for the SBC method.

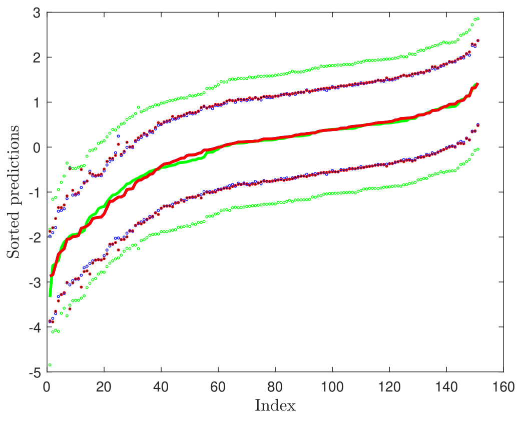
Airfoil In this regression experiment, the delta method provides the most competitive CWC_{Best} measure and the Bayesian method has the best CWC_{Med} and CWC_{SD} measure. Furthermore, the difference between the PICP of the delta method and SBC method compared to the Bayesian method is significant. The delta method and SBC method produce competitive PICP measures as the average PICP, 0.9404 and 0.9430 for the delta and SBC method respectively, is close the nominal confidence level. The methods are also similar in their MPIW measure since the average MPIW of the delta method is 1.8043 and the PIs constructed with the SBC method are only slightly wider (1.8696). On the other hand, the PIs constructed by the Bayesian method are considerably wider resulting in higher average PICP (0.9834) than the nominal confidence level. In this regard, the Bayesian method constructs conservative PIs. Also, the MSE for the Bayesian method is slightly higher compared to the other methods. The meanMSE of the Bayesian method is slightly higher compared to delta and SBC. In addition, the mean uncertainty interval construction execution time is 0.2772 for the delta method, 1.0241 for the Bayesian method and 0.1834 for the SBC method.

Concrete In this regression experiment, the SBC method provides the most competitive CWC_{Best} measure and the Bayesian method has the best CWC_{Med} and CWC_{SD} measure. Furthermore, the average PICP for the delta method and SBC method is slightly less then the nominal confidence level of 95 % as the average PICP is 0.9311 and 0.9408 for the delta and SBC method respectively. Thereby, both methods construct prediction intervals that are on average slightly too optimistic. On the other hand, the Bayesian PICP are significantly higher than the nominal confidence level (0.9903). These observations also explain the difference of the MPIW measure, where the MPIW is significantly greater for the Bayesian method (2.7664) compared to the average MPIW for the delta method (1.4070) and SBC method (1.4653). As a result, the Bayesian method is considerably less informative. The meanMSE of the Bayesian method is slightly higher compared to delta and SBC. In addition, the mean uncertainty interval construction execution time is 0.0896 for the delta method, 1.3130 for the Bayesian method and 0.0710 for the SBC method.

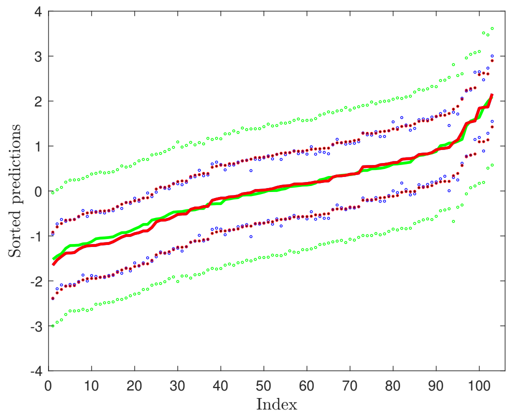
For the three regression experiments, it is not possible to plot the predictions with uncertainty intervals as the datasets contain higher dimensional data ($D > 3$). However, the predictions can be ordered and plotted with their corresponding uncertainty interval. In Figure 6-7, predictions (full middle lines) are sorted and plotted with PIs (dots) for the three methods in a single plot for run 1 of the three regression experiments. This Figure gives a visualization of how the width of the constructed PIs of the three different methods relate to each other. It can be observed that the Bayesian PIs are generally the widest and the width of the delta and SBC constructed PIs are notably similar.



(a) Power plant



(b) Airfoil

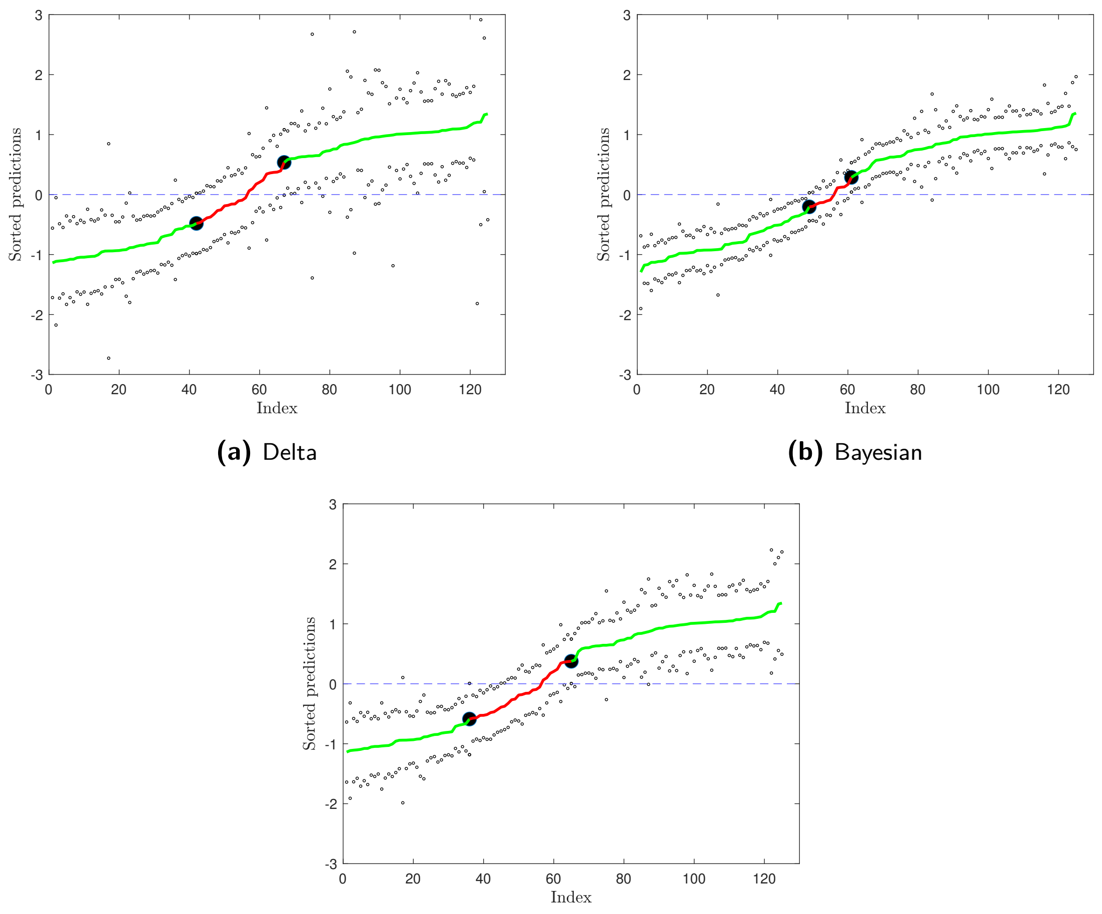


(c) Concrete

Figure 6-7: Sorted predictions (middle full line) and corresponding prediction intervals (dots below and above middle line) for run number 1 of the Power plant, Airfoil and Concrete experiments, where blue corresponds to the delta method, green is the Bayesian method and red is the SBC method. The middle blue line aligns with the red line as the predictions of delta and SBC equal.

Ripley In this classification experiment, the SBC method provides the most competitive $\mathrm{CWC}_{\mathrm{Best}}$ measure and the delta method has the best $\mathrm{CWC}_{\mathrm{Med}}$ and $\mathrm{CWC}_{\mathrm{SD}}$ measure. Furthermore, all three methods are able to produce confidence intervals that on average approximate the nominal confidence level of 95%, which can be concluded by the average CICP which are 0.9729, 0.9566 and 0.9569 for the delta, Bayesian and SBC method respectively. Therefore, form a correctness respective, the three methods all produce high quality CIs. From an informativeness perspective, the average MCIW is smallest for the SBC method (1.1262), followed by the delta method (1.3404) and Bayesian method (1.8457). In this regard, the SBC method performs best when taking into account both correctness and informativeness of the constructed CIs. As a supplementary point, the mean uncertainty interval construction execution time is 0.0831 for the delta method, 0.0831 for the Bayesian method and 0.0833 for the SBC method.

Similar to regression tasks, the predictions before taking the $\mathrm{sign}(\cdot)$ can be sorted and plotted with their corresponding PI. In the plot, the middle full line represents the sorted prediction of the classifier. The small dots above and below the middle full line are the corresponding prediction intervals. The horizontal blue dashed line at 0 represents the decision boundary. In the red region between the two big dots, the classifier casts doubt on the corresponding label with a confidence level 95%. Likewise, in the green region outside the big dots, the classifier is certain on the corresponding label with a confidence level of 95%. This type of plot is constructed for the Ripley, Banknote and Raisin experiments in Figure 6-8, Figure 6-9 and Figure 6-10 respectively.



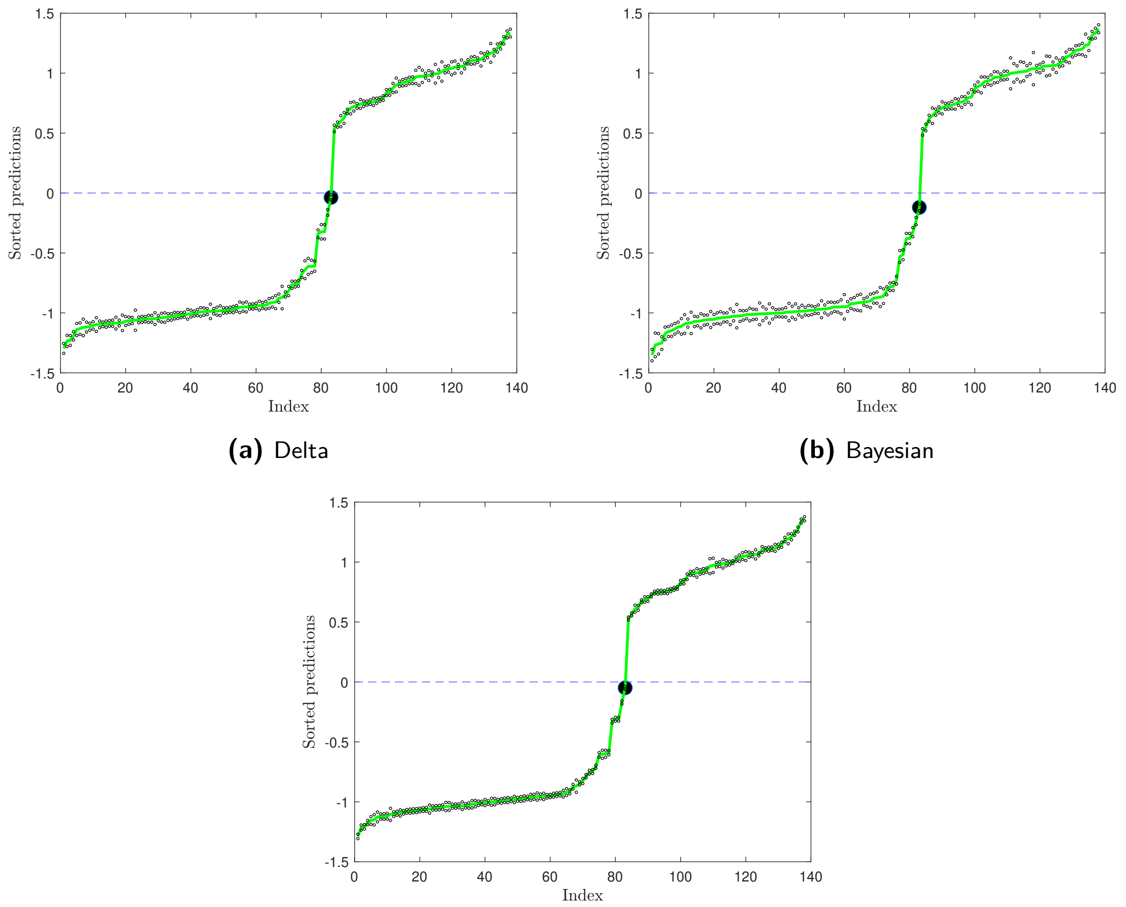
(a) Delta

(b) Bayesian

(c) SBC

Figure 6-8: Sorted predictions (middle full line) and corresponding prediction intervals (dots below and above middle line) for run number 1 of the Ripley experiment, the classifier casts doubt with significance level of 5% on the labels of prediction lying between the two big dots (in red) and is confident with significance level of 5% on the labels of the predictions outside the big dots (in green).

Banknote In this classification experiment, the SBC method provides the most competitive $\mathrm{CWC}_{\mathrm{Best}}$ and $\mathrm{CWC}_{\mathrm{Med}}$ measure and the delta method has the best $\mathrm{CWC}_{\mathrm{SD}}$. Furthermore, for all three methods the PICP is equal to 1 and the MR is equal to 0. This means that the algorithm was perfectly able to classify all inputs in the test set. In this regard, it is interesting to see which methods scores best on informativeness as the correctness of the PIs is equal. The average MPIW is narrowest for the SBC method (0.0490), followed by the delta method (0.0728) and the Bayesian method (0.1021). Hence, the SBC method would produce the highest quality PIs in such a scenario. In addition, the mean uncertainty interval construction execution time is 0.0800 for the delta method, 0.1580 for the Bayesian method and 0.0707 for the SBC method.



(a) Delta **(b)** Bayesian

(c) SBC

Figure 6-9: Sorted predictions (middle full line) and corresponding prediction intervals (dots below and above middle line) for run number 1 of the Banknote experiment, the classifier casts doubt with significance level of 5% on the labels of prediction lying between the two big dots (in red) and is confident with significance level of 5% on the labels of the predictions outside the big dots (in green).

Raisin For the final classification experiment, the Bayesian method provides the most competitive $\mathrm{CWC}_{\mathrm{Best}}$ and the SBCmethod has the best $\mathrm{CWC}_{\mathrm{Med}}$ and $\mathrm{CWC}_{\mathrm{SD}}$. Furthermore, the delta method and SBC method are well able to approximate the nominal confidence level as the average CICP of the delta method (0.9542) and the SBC method (0.9698) just above the nominal confidence level of 95%. The Bayesian method also performs well on correctness but the average CICP (0.9361) is slightly lower than the nominal confidence level. In terms of informativeness, the Bayesian method and the SBC method produce CIs that are on average about the same width since the average MCIW is 0.9361 and 0.9698 for the Bayesian and SBC method respectively. The average MCIW of the delta method is considerably wider (2.8071). Furthermore, When observing the MCIW for each method, it is noticeable that the measure across the different runs varies considerably. This is especially noticeable for the delta method. In addition, the mean uncertainty interval construction execution time is 0.0616 for the delta method, 0.3305 for the Bayesian method and 0.0572 for the SBC method.

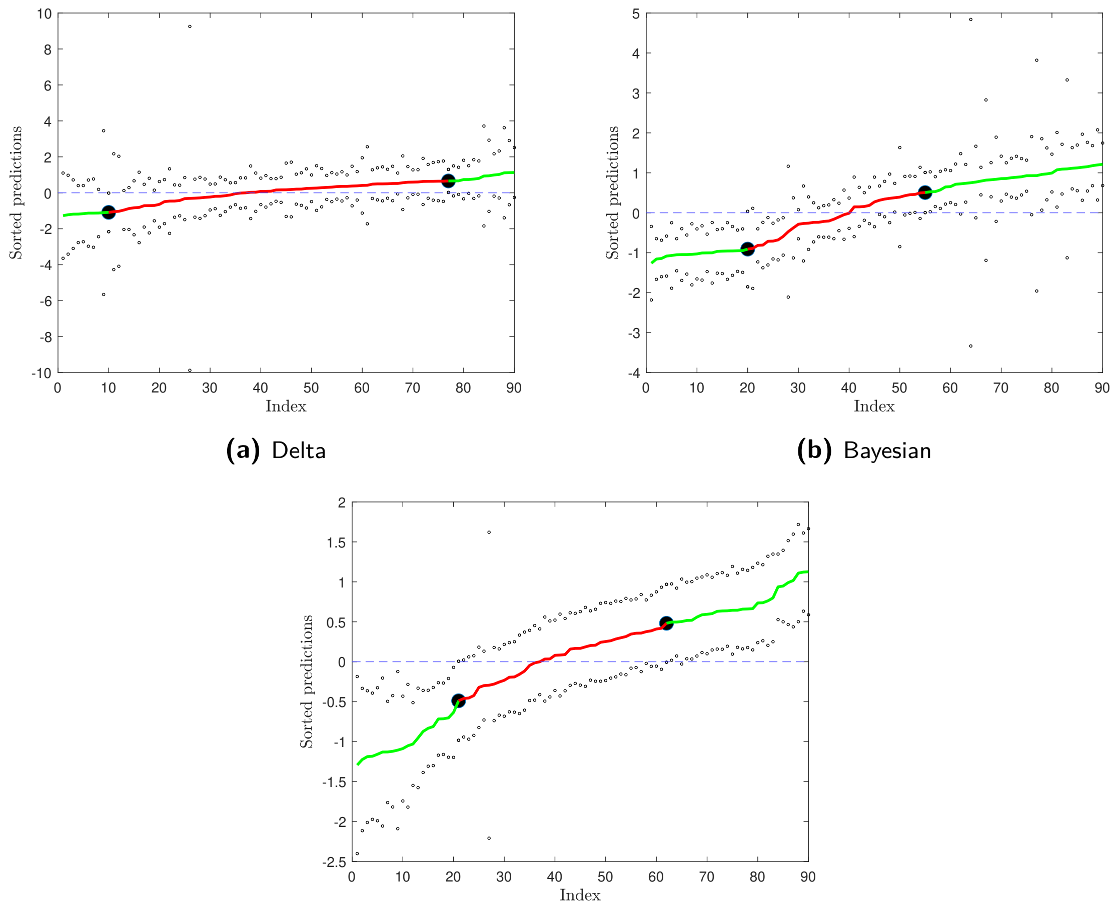


Figure 6-10: Sorted predictions (middle full line) and corresponding prediction intervals (dots below and above middle line) for run number 1 of the Raisin experiment, the classifier casts doubt with significance level of 5% on the labels of prediction lying between the two big dots (in red) and is confident with significance level of 5% on the labels of the predictions outside the big dots (in green).

Based on the observed results, the following conclusions on the quality of the uncertainty quantification methods can be distilled:

1. **The delta, Bayesian and SBC method provide reasonably accurate uncertainty quantification in terms of correctness and informativeness. The delta method**

and SBC method provide the highest quality uncertainty intervals which are remarkably similar.

2. **Between the delta end SBC method, the delta method is slightly more informative in regression tasks compared to the SBC method. On the other hand, the SBC method is slightly more informative in classification tasks.**

3. **The time it takes to construct uncertainty intervals is generally the highest for the Bayesian method, followed by the delt and SBC method. This difference is amplified when the number of features increases.**

4. **The Bayesian uncertainty intervals are in general relatively conservative.**

Chapter 7

Conclusion and Discussion

In this section the conclusions on the results will be summarized and a discussion will be provided with respect to the utilized methods and the obtained results. Finally, recommendations for future research in will be provided.

A caveat in the proposed delta method is the approximation of the Hessian of the regularization term. The Fisher information matrix is used to estimate the uncertainties in the parameter estimates expressed as the parameter covariance matrix. The Hessian matrix measures how sensitive the model's likelihood function is to changes in the model parameters. Besides uncertainties in the parameters by the data-fitting term, the regularization term also influences the uncertainties of parameter estimates. Therefore, inclusion of the Hessian of the regularization term is essential for a more accurate estimation of the parameter uncertainties in the presence of regularization. Not doing so may lead to overly optimistic parameter uncertainties, as it would not account for the regularization's influence on the curvature of the cost function. Ensuring that the parameter covariance matrix properly reflects the combined influence of the data-fitting term and regularization leads to more reliable parameter estimates. In this research, as the analytical derivation of the Hessian of the regularization term can become a cumbersome task, an approximation is used by considering only the block diagonal terms. It is assumed in this research that the effect of this approximation is insignificant. Moreover, the cross-validation procedure can determine the influence of the regularization term by the scaling regularization hyperparameter λ_{reg}, therefore, the precise magnitude of the Hessian of the regularization term is inconsequential.

Furthermore, a major drawback drawback can be identified for the Bayesian method. This method adopts the unscented transform which produces an approximate mean and covariance of the full parameter tensor. Thereby, this methods scales exponentially in the number of features of the data, rendering a storage complexity of $\mathcal{O}(\hat{M}^{2D})$. In this way, the Bayesian method, which adopts the Bayesian Canonical Polyadic Decomposition Alternating Linear Scheme (CP-ALS) algorithm, is inapplicable for datasets with more features. This issue can be circumvented by modelling the unscented mean and covariance as a tensor network. This is done in [34] for the Tensor Train where a rounding procedure takes place when the propagated sigma points are added in Tensor Train format which increases the ranks of the Tensor Train.

However, such a procedure, performed by the TT-rounding algorithm, does not exists for the CPD, thereby rendering this method infeasible for higher-dimensional data.

This research serves as a comparison between uncertainty quantification methods for tensor network constrained kernel machine, focusing on the CPD constrained kernel machine. The validation on both actual datasets and synthetic data, where the experiments are set up in a way that assumptions governing the uncertainty quantification methods hold, has been important. In this way, the uncertainty quantification methods are assessed on their quality performance in problems where data may violate assumptions, in the case of real datasets, or certainly condemn assumptions in the case of synthetic experiments.

7-1 Discussion on the uncertainty quantification performance on synthetic experiments

The synthetic experiments show favorable results. These experiments were constructed such that the assumptions governing the different methods were guaranteed to hold. For the parameter covariance, where the assumptions are listed in 4-2-1 and 4-2-2, it is known that when such assumptions do not hold, this can generate low-quality uncertainty quantification results. For example, heteroscedasticity of the noise variance can generate low-quality confidence intervals (CIs) [22].

In terms of correctness, which is connected to the coverage probability of the CI (CICP), the delta and Single Bayesian Core (SBC) method are well able to approximate the nominal confidence level very closely. On the other hand, the Bayesian method slightly overestimates the confidence interval coverage probability (CICP) by covering more test points. This does not necessarily mean that the constructed CIs are faulty, but the overestimation generally leads to wider uncertainty intervals, which can be classified as conservative. With regards to informativeness, the delta and SBC method have significantly narrower CIs than the CIs constructed by the Bayesian method.

Deviations of the Bayesian CIs compared to the delta and SBC constructed CIs mainly result from two aspects. The first aspect is the way that the factor matrices are computed. The Bayesian method takes into account a prior mean and covariance on each factor factor matrix, in addition, the noise variance of the aleatory error term is taken into account as a hyperparameter. Bayesian priors encapsulate beliefs or uncertainty about parameters, whereas in the conventional CP-ALS that is applied in the frequentist delta and SBC method a regularization terms is incorporated that is conceptually similar to a prior in Bayesian statistics. The concept of a prior in Bayesian statistics has a broader meaning, encompassing beliefs or information about parameters beyond regularization.

The second aspect is the way that the prediction error variance is computed, which is different for all three methods. In the delta method, the prediction error variance is constructed based on an approximation of the parameter covariance of its parameter estimates which is computed with the inverse of the Fisher information matrix. This is because a nonlinear least squares problem is solved where the objective is to minimize the sum of squared residuals, which is analogous to maximizing the likelihood function. The Fisher information quantifies how much information the observed data contains about the parameters. The Fisher Information matrix is directly related to the second derivative of the log-likelihood function with respect to the parameters, which is essentially the Hessian matrix of the log-likelihood. On the other hand, the Bayesian method computes the unscented covariance with the conditional parameter covariance of each factor matrix and makes predictions by averaging over all possible parameter values, weighted by the unscented covariance. The SBC only treats a single factor matrix as having a probability distribution and makes predictions by first applying a projection to the posterior distribution of the stochastic factor matrix.

In the 8-D synthetic data experiment where only the delta and SBC methods are compared. The mean confidence Interval width (MCIW) is slightly narrower for the delta method compared to the SBC method, rendering the delta method most informative as both methods perform well on correctness by approximating the nominal confidence level closely. Overall, the different methods to construct uncertainty intervals are based on different philosophical foundations: frequentist, Bayesian, or a mix of both. However, the methods yield relatively similar intervals with the Bayesian method being slightly more conservative compared to the delta and SBC method.

7-2 Discussion on the uncertainty quantification performance on datasets

The results of experiments on actual datasets are somewhat less straightforward to draw conclusions upon. A first remark is that the chosen hyperparameters by the cross-validation procedure can differ a lot across different runs for some experiments. Especially the scaling hyperparameter for the prior covariance in the Bayesian and SBC method deviates across the different runs. Hence, both relatively certain and uncertain prior covariance distributions can lead to similar results in terms of prediction interval coverage probability (PICP) and mean prediction Interval width (MPIW). Another remark is concerned with the mean squared error (MSE) and misclassification rate (MR). The MSE and MR are equal for the delta and SBC method as both methods adopt the conventional CP-ALS algorithm. On the other hand, the Bayesian method adopts the Bayesian CP-ALS algorithm which produces a mean and covariance for a factor matrix as opposed to fixed point estimates for the conventional CP-ALS algorithm. The MSE and MR incurred with predictions from the delta and SBC compared to the Bayesian method exhibit no significant differences.

On the three regression task experiments, by some small margin of error for experiment 2 and 3, the delta method and SBC method are able to very closely match with the nominal confidence level. The Bayesian method is also able to cover at least 95% of the test inputs on average. From an informativeness perspective, it can be concluded that the Bayesian method often produces wider prediction intervals (PIs), but does not necessarily obtain a much better PICP while doing so. The delta method and SBC would be preferred over the Bayesian method in regression tasks do to their competitive performance mainly from an informativeness point of view. When comparing the constructed uncertainty intervals of the delta and SBC method, the SBC method provides slightly more informative intervals in the regression experiments.

On the three classification experiments, all three methods perform well from a correctness point of view. This is because the average CICP closely matches the nominal confidence level across all experiments. In the Banknote experiment the CICP is effectively equal to 1 as the misclassification rate is zero. From an informative perspective, the Bayesian method provides (except for experiment 6) the least informative CIs as the average CICP is in general highest for the Bayesian method. This can also be visually observed from the 2-D classification plots on the Banana and Ripley dataset. However, a user might still use the Bayesian intervals when relatively conservative intervals are important for a specific application. Similar to the regression experiments, the uncertainty intervals for the delta and SBC method are remarkably similar but differ in that the delta method provides slightly more informative intervals in the classification experiments compared to the SBC method.

The most important aspect in this research is the correctness and the informativeness of the constructed uncertainty measures. However, an additional time for the construction of the uncertainty measure could also be of importance for a user the proposed methods. Moreover, the computation time of uncertainty intervals could be decisive factor when the performance of two methods is relatively similar. The time for CI and PI construction takes longest in almost all experiments for the Bayesian method, followed by the delta method and the SBC method is generally the fastest. These observation confirm the theoretically stated computational complexity of the uncertainty interval construction execution time. Namely, the computational complexity for construction of the covariance, when the derivatives of the model with respect to each factor matrix are already obtained, is $\mathcal{O}((D\hat{M}R)^2N + (D\hat{M}R)^3)$ for the delta method and $\mathcal{O}((\hat{M}R)^2N + (\hat{M}R)^3)$ for the SBC method. In addition, computing the prediction error variance after obtaining the parameter covariance has a computational complexity of $\mathcal{O}((D\hat{M}R)^2N_{\text{test}})$ for the delta method ans $\mathcal{O}((\hat{M}R)^2N_{\text{test}})$ for the SBC method. As the difference in the complexities remains to be the linear D term, the difference in the computational time between the methods is little. However, the Bayesian method has computational complexity of at least $\mathcal{O}(DR\hat{M}^{2D+1})$ for the construction of the unscented covariance and $\mathcal{O}(\hat{M}^DN_{\text{test}})$ for making predictions.

In practice, the computation time of the construction of the uncertainty intervals should not be the only factor with regards to the overall computation time. This is because, the K-fold cross-validation procedure is not taken into account in the PI construction time as the amount of time for this procedure is dependent on the amount of folds and the number of hyperparameters that participates in the procedure. From this perspective, the Bayesian method ans SBC method both incorporate two hyperparameters in the K-fold cross-validation procedure for regression tasks, whereas the delta method uses only one. In addition, the number different values for a hyperparameter that participate in the K-fold cross-validation procedure has a significant effect.

An exact comparison between the uncertainty quantification methods for real datasets would be more explanatory and sophisticated when more information about the particular dataset is gathered. As such, the proposed methods could be tailored in order to meet the specific assumptions brought by the dataset. For example, normally distributed and homoscedastic observation noise are characteristics of the noise incorporated in the data that are assumed by all three proposed methods. Violating these assumptions by the data could lead to inaccurate results for the derived uncertainty of the predictions. For example, it could be the case that assumptions the observation noise inherent to data are violated for the delta method in experiment 6 as the width of the intervals varies considerably across different runs. In such a situation, the uncertainty quantification method does not serve as a rigorous experiment in terms of an accurate comparative analysis with the other methods.

Furthermore, it is unknown to what degree there are structural model errors in the dataset experiments. Structural model errors can, for example, arise when there is significant model misspecification, in the way that the model is unable to fully explain the data. This can be the case when the true underlying function that describes the data is not part of the model structure. It has been explained in this research that when there is significant structural model error, the constructed confidence intervals and prediction intervals are inherently inaccurate and there is no way to circumvent this issue. It is not appropriate to validate on the performance of the obtained uncertainty quantification results on actual datasets as it is unknown to what degree the errors in the prediction for real datasets possess structural model error.

7-3 Conclusion

In this research, uncertainty quantification for a Canonical Polyadic Decomposition (CPD) constrained kernel machine model is examined. The particular CPD constrained kernel machine model studied in this research employs a product feature map with polynomial features. Uncertainty quantification in this research is concerned with the uncertainty in the predictions which is encapsulated by the prediction error variance. The prediction error variance is treated as estimation error which is an epistemic source of error as opposed to the observation noise variance which is a source of aleatory error. Three different uncertainty quantification methods are proposed and assessed on their ability to provide qualitative uncertainty quantification measures. The main research in this research is:

> ***How to provide uncertainty quantification for tensor network constrained kernel machine, focusing on the CPD constrained kernel machine, and which of the investigated methods is most accurate in terms of correctness and informativeness?***

The three methods that have been extensively studied in this research are: the delta method, the Bayesian method and the Single Bayesian Core (SBC) method. The delta method is an inherently frequentist method that constructs the inverse of the Fisher information in order to provide a covariance matrix for the estimated model parameters which is done by computing the Jacobian of the model with respect to each factor matrix evaluated at the estimated factor matrices by the CP-ALS algorithm. In turn, the gradient vector of the test model with respect to each factor matrix is computed and evaluated at the estimated factor matrices in order to provide a way of quantifying the uncertainty in the prediction of the model by propagating the parameter covariance to the uncertainty in the output of the model. On the other hand, the Bayesian model is based on a different philosophy where parameters are viewed as random variables with probability distributions. The Bayesian CP-ALS algorithm incorporates a prior mean and covariance distribution for each factor matrix and computes the posterior mean and covariance of each factor matrix. The SBC method combines aspects from the delta and Bayesian method, as the conventional CP-ALS algorithm is adopted and uncertainty is quantified by treating a single factor matrix as a Bayesian factor matrix.

The quality of the three uncertainty quantification methods is assessed on two aspects: correctness and informativeness. Correctness is assessed by evaluating the ability of constructed confidence intervals and prediction intervals to cover the underlying true function and the test inputs respectively. This coverage should approximate the chosen nominal confidence level. From an informativeness perspective, the constructed uncertainty intervals should be as narrow as possible, while maintaining accurate coverage.

Experiments have been performed on both synthetically generated data and real datasets. It is assumed that the structural model error is insignificant, as structural model error would inherently produce inaccurate uncertainty quantification results. In general, all three methods provide accurate uncertainty intervals in terms of correctness. The Bayesian intervals often provide higher coverage than the nominal confidence level which directly reflects on the informative aspect as the intervals are generally wider than the delta and SBC intervals. A major drawback of the Bayesian method is its lack of scalability as the size of the unscented mean and covariance in the Bayesian method scales exponentially. Furthermore, uncertainty intervals of the delta method are remarkably similar to uncertainty intervals constructed with the SBC method. Comparing the two, the SBC intervals are on average slightly wider in regression tasks, whereas the delta intervals are slightly wider in classification tasks.

All three methods could readily be applied in order to provide valuable uncertainty quantification for the CPD constrained kernel machine. The Bayesian method is in general more conservative than the delta and SBC method. Moreover, the delta and SBC method quantify the error in the predictions relatively similar in terms of correctness and informativeness.

7-4 Future research

This research has a particular scope on the goal of providing valuable uncertainty quantification for tensor network constrained kernel machines. That is, particular choices have been made in in order to narrow down the problem. For example, the tensor network that is considered is the CPD, however, it would be interesting to examine how the proposed uncertainty quantification methods perform when adopting other tensor networks. This would especially be interesting for tensor networks for which a rounding procedure exists when adding tensor networks, for example the TT-rounding algorithm for Tensor Trains. This is because, a rounding procedure can circumvent the exponential increase in the unscented mean and covariance constructed in the the Unscented Transform (UT). In this way, a more comprehensive comparative analysis can be provided for the performance of the Bayesian method. Besides the Bayesian method, it would be interesting to examine the performance of the delta method and SBC method when adopting other tensor networks.

In this research, the Kernel Ridge Regression (KKR) is studied and polynomial features are considered. Various other choices choices for the loss function, the regularization term could be studied to provide a more generalized description of the performance in terms of correctness and informativeness of the of the proposed uncertainty quantification methods.

Appendix A

Algorithms

A-1 K-fold cross validation algorithm

Algorithm 2 Hyperparameter Tuning using K-fold Cross Validation

```
 1: procedure HyperparameterTuning($\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}}, K, \text{threshold}$)
 2:     Prior hyperparameters $\boldsymbol{\rho} \leftarrow [\rho_1, \rho_2, \ldots, \rho_M]$
 3:     Noise hyperparameters $\bar{\boldsymbol{\sigma}}^2 \leftarrow [\bar{\sigma}_1^2, \bar{\sigma}_2^2, \ldots, \bar{\sigma}_P^2]$ Split $\mathbf{X}_{\text{train}}$ into $K$ folds
 4:     for $\rho_m$ in $\boldsymbol{\rho}$ do
 5:         for $\bar{\sigma}_p^2$ in $\bar{\boldsymbol{\sigma}}^2$ do
 6:             for $i = 1$ to $K$ do
 7:                 Use fold $i$ for test data $\mathbf{X}_{\text{test}}^i$, $\mathbf{y}_{\text{test}}^i$ and the rest for
 8:                 training data $\mathbf{X}_{\text{train}}^{K-i}$, $\mathbf{y}_{\text{train}}^{K-i}$
 9:                 Train model and obtain prediction intervals
10:                 Obtain error$_i$ from (6-1) in regression and (6-2) in classification
11:                 Obtain CWC$_i$ from (6-3), (6-4), (6-5), (6-6) in regression and (6-7), (6-4),
    (6-5), (6-6)
12:                 in classification
13:             end for
14:             Mean error$_{mp} \leftarrow [\text{error}_1, \text{error}_i, \ldots, \text{error}_K]$
15:             Mean error$_{mp} \leftarrow [\text{CWC}_1, \text{CWC}_i, \ldots, \text{CWC}_K]$
16:         end for
17:     end for
18:     if Mean error$_{mp} < (1 + \text{threshold}) \min(\text{Mean error})$ then
19:         The $m$ and $p$ pair are a candidate corresponding to CWC value $\text{CWC}_{\text{candidate}}^{mp}$
20:     end if
21:     Best $\rho$ and best $\bar{\sigma}^2 \leftarrow \min(\text{CWC}_{\text{candidate}})$
22:     return bestAlpha, bestSigma
23: end procedure
```

A-2 Bayesian CP-ALS algorithm

Algorithm 3 MATLAB Function: `CP-ALS-Bayesian`

```
procedure CP ALS($\mathbf{X}, \mathbf{y}, M, R, \alpha, \sigma^2$, *number of sweeps*)
    $D \leftarrow size(\mathbf{X}, 2)$                                   ▷ Extract number of features

    $\mathbf{ZW} \leftarrow 1$                                           ▷ Initialization of algorithm
    $\mathbf{\Gamma} \leftarrow 1$
    for $d \leftarrow D$ to 1 do
        $\mathbf{W}^{(d)}_{\text{mean}} \leftarrow randn(M, R)$
        $\mathbf{W}^{(d)}_{\text{mean}} \leftarrow \mathbf{W}^{(d)}_{\text{mean}} / ||\mathbf{W}^{(d)}||$
        $\mathbf{ZW} \leftarrow \mathbf{ZW} * \mathbf{Z}(\mathbf{X}^{(d)})^T \mathbf{W}^{(d)}_{\text{mean}}$
        $\mathbf{m}^0_d \leftarrow \text{vec}(\mathbf{W}^{(d)}_{\text{mean}})$
        $(\mathbf{P}^0_d)^{-1} \leftarrow (\alpha \mathbf{I})^{-1}$
    end for

    for $i \leftarrow 1$ to *number of sweeps* do                       ▷ Sweeping over the factor matrices
        for $d \leftarrow 1$ to $D$ do
            $\mathbf{ZW_d} \leftarrow \mathbf{ZW} / (\mathbf{Z}(\mathbf{X}^{(d)})^T \mathbf{W}^{(d)}_{\text{mean}})$
            $\mathbf{C} \leftarrow \left( \mathbf{Z}(\mathbf{X}^{(d)}) \odot \left( *^{D}_{p=1, p \neq d} \mathbf{Z}(\mathbf{X}^{(p)})^T \mathbf{W}^{(p)}_{\text{mean}} \right)^T \right)^T$
            $\mathbf{W}^{(d)}_{\text{cov}} \leftarrow \frac{\mathbf{C}^T \mathbf{C}}{\sigma^2} + (\mathbf{P}^d_0)^{-1}$
            $\mathbf{m}^0_d \leftarrow \text{vec}(\mathbf{W}^{(d)}_{\text{mean}})$
            $\mathbf{b} \leftarrow \frac{\mathbf{C}^T \mathbf{y}}{\sigma^2} + (\mathbf{P}^d_0)^{-1} \mathbf{m}^0_d$
            $\mathbf{m}^+_d \leftarrow solve(\mathbf{W}^{(d)}_{\text{cov}}, \mathbf{b})$
            $\mathbf{W}_{\text{mean}} \leftarrow reshape(\mathbf{m}^+_d, (M, R)))$
            $\mathbf{ZW} \leftarrow \mathbf{ZW_d} * (\mathbf{Z}(\mathbf{X}^{(d)})^T \mathbf{W})$
            $\mathbf{W}^{(d)} \leftarrow \mathbf{W}$
        end for
    end for
    return $\mathbf{W}^{(1)}_{\text{mean}}, \mathbf{W}^{(2)}_{\text{mean}}, \ldots, \mathbf{W}^{(D)}_{\text{mean}}$ ▷ Return mean and covariance of factor matrices
    and $\mathbf{W}^{(1)}_{\text{cov}}, \mathbf{W}^{(2)}_{\text{cov}}, \ldots, \mathbf{W}^{(D)}_{\text{cov}}$
end procedure
```

A-3 Unscented Transform algorithm

Algorithm 4 MATLAB Function: `Unscented Transform (UT)`

```
1: procedure UT($\mathbf{W}^{(1)}_{\text{mean}}, \mathbf{W}^{(2)}_{\text{mean}}, \ldots, \mathbf{W}^{(D)}_{\text{mean}}, \mathbf{W}^{(1)}_{\text{cov}}, \mathbf{W}^{(2)}_{\text{cov}}, \ldots, \mathbf{W}^{(D)}_{\text{cov}}$)
2:     Compute known distribution from factor matrices with 5-9
3:     Compute Sigma points with 5-10
4:     Propagate sigma points through nonlinear function using 5-11
5:     Compute weighting factors with 5-13
6:     Estimate the unscented mean and covariance using 5-14
7:     return $\mathbf{m}_{\text{UT}}$ and $\mathbf{P}_{\text{UT}}$
8: end procedure
```

Appendix B

Results experiments

B-1 Experiment 1

Table B-1: Results case study 1 for the delta method

Run	**Delta**						
	CICP	**MPIW**	**CWC**	**MSE**	**Time (s)**	λ_{reg}	$\hat{\sigma}_e^2$
1	0.9645	0.9296	0.9296	0.0535	1.0676	1^{-4}	0.0537
2	0.9707	0.9279	0.9279	0.0567	1.0415	1^{-4}	0.0534
3	0.9687	0.9398	0.9398	0.0566	1.1730	1^{-3}	0.0549
4	0.9634	0.9276	0.9276	0.0594	1.1870	1^{-4}	0.0534
5	0.9592	0.9296	0.9296	0.0629	1.2860	1^{-3}	0.0537
6	0.9603	0.9344	0.9344	0.0615	1.2153	1^{-3}	0.0543
7	0.9551	0.9386	0.9386	0.0597	1.2138	1^{-3}	0.0547
8	0.9519	0.9340	0.9340	0.0623	1.2945	1^{-3}	0.0542
9	0.9645	0.9291	0.9291	0.0558	1.1616	1^{-4}	0.0535
10	0.9728	0.9312	0.9312	0.0546	1.1913	1^{-4}	0.0538
Mean	0.9631	0.9322	0.9322	0.0583	1.1832	N/A	0.0540

Table B-2: Results case study 1 for the Bayesian method

Run	Bayesian						
	CICP	MPIW	CWC	MSE	Time (s)	ρ	$\bar{\sigma}_e^2$
1	0.9864	1.2665	1.2665	0.0590	1.9006	1^{-3}	0.1
2	0.9655	0.9098	0.9098	0.0561	2.2064	1	0.05
3	0.9666	0.9103	0.9103	0.0542	1.8652	1	0.05
4	0.9896	1.2665	1.2665	0.0637	2.0929	1^{-3}	0.1
5	0.9530	0.9042	0.9042	0.0619	2.0049	0.1	0.05
6	0.9875	1.2664	1.2664	0.0678	2.2623	1^{-3}	0.1
7	0.9498	0.9113	1.9192	0.0578	2.1045	1	0.05
8	0.9446	0.9058	2.2145	0.0620	2.0374	0.1	0.05
9	0.9948	1.2667	1.2667	0.0595	2.3699	1^{-3}	0.1
10	0.9937	1.2667	1.2667	0.0594	2.1210	1^{-3}	0.1
Mean	0.9731	1.0874	1.3191	0.0602	2.0965	N/A	N/A

Table B-3: Results case study 1 for the SBC method

Run	SBC							
	CICP	MPIW	CWC	MSE	Time (s)	λ_{reg}	ρ	$\hat{\sigma}_e^2$
1	0.9666	0.9425	0.9425	0.0535	0.9080	1^{-3}	50	0.0549
2	0.9707	0.9491	0.9491	0.0567	0.9513	1^{-2}	1	0.0558
3	0.9687	0.9518	0.9518	0.0566	0.8884	1^{-2}	5	0.0560
4	0.9624	0.9494	0.9494	0.0594	0.9110	1^{-2}	50	0.0557
5	0.9592	0.9385	0.9385	0.0629	1.0362	1^{-2}	0.001	0.0548
6	0.9572	0.9467	0.9467	0.0615	0.8661	1^{-2}	5	0.0554
7	0.9572	0.9503	0.9503	0.0597	0.9685	1^{-2}	1	0.0558
8	0.9561	0.9476	0.9476	0.0623	0.9902	1^{-2}	50	0.0554
9	0.9655	0.9498	0.9498	0.0558	0.9538	1^{-2}	1	0.0559
10	0.9760	0.9542	0.9542	0.0546	0.9228	1^{-2}	5	0.0563
Mean	0.9639	0.9480	0.9480	0.0583	0.9396	N/A	N/A	N/A

B-2 Experiment 2

Table B-4: Results case study 2 for the delta method

Run	Delta						
	CICP	**MPIW**	**CWC**	**MSE**	**Time (s)**	λ_{reg}	$\hat{\sigma}_e^2$
1	0.9404	1.8580	3.4743	0.2362	0.1836	1^{-2}	0.2080
2	0.9338	1.6605	3.9112	0.1969	0.3217	1^{-3}	0.1677
3	0.9470	1.8535	3.0142	0.2336	0.3282	1^{-2}	0.2092
4	0.9536	1.8396	1.8396	0.2267	0.2446	1^{-2}	0.2055
5	0.9404	1.8458	3.4621	0.2651	0.2464	1^{-2}	0.2070
6	0.9205	1.8274	6.1920	0.2486	0.2499	1^{-2}	0.2043
7	0.9536	1.8376	1.8376	0.2180	0.2517	1^{-2}	0.2055
8	0.9603	1.6513	1.6513	0.2196	0.3483	1^{-3}	0.1632
9	0.9470	1.8337	2.9944	0.2488	0.3481	1^{-2}	0.2054
10	0.9073	1.8355	10.2990	0.3606	0.2493	1^{-2}	0.2017
Mean	0.9404	1.8043	3.8676	0.2454	0.2772	N/A	0.1978

Table B-5: Results case study 2 for the Bayesian method

Run	Bayesian						
	CICP	**MPIW**	**CWC**	**MSE**	**Time (s)**	ρ	$\bar{\sigma}_e^2$
1	0.9801	2.8808	2.8808	0.3311	1.2937	0.1	0.5
2	0.9801	2.8720	2.8720	0.2481	0.9480	0.1	0.5
3	0.9801	2.8734	2.8734	0.2728	0.9176	0.1	0.5
4	1	2.9663	2.9663	0.2053	0.9157	0.1	0.5
5	0.9735	2.8711	2.8711	0.2985	1.2233	0.1	0.5
6	0.9801	2.8759	2.8759	0.2618	0.8870	0.1	0.5
7	0.9934	3.0196	3.0196	0.1566	1.2898	0.1	0.5
8	0.9868	2.8672	2.8672	0.2744	0.8887	0.1	0.5
9	0.9801	2.8639	2.8639	0.3123	0.9274	0.1	0.5
10	0.9801	3.0768	3.0768	0.4105	0.9494	0.1	0.5
Mean	0.9834	2.9167	2.9167	0.2771	1.0241	N/A	N/A

Table B-6: Results case study 2 for the SBC method

Run	**SBC**							
	CICP	**MPIW**	**CWC**	**MSE**	**Time (s)**	λ_{reg}	ρ	$\hat{\sigma}_e^2$
1	0.9404	1.8949	3.5112	0.2362	0.2763	1^{-2}	50	0.2085
2	0.9404	1.8656	3.4819	0.1969	0.2425	1^{-2}	50	0.2083
3	0.9470	1.8818	3.0425	0.2336	0.2134	1^{-2}	100	0.2082
4	0.9603	1.8613	1.8613	0.2267	0.1842	1^{-2}	5	0.2068
5	0.9404	1.8838	3.5001	0.2651	0.1855	1^{-2}	10	0.2069
6	0.9272	1.8510	4.9852	0.2486	0.1727	1^{-2}	10	0.2047
7	0.9470	1.8623	3.0230	0.2180	0.2377	1^{-2}	100	0.2043
8	0.9669	1.8751	1.8751	0.2196	0.1721	1^{-2}	100	0.2057
9	0.9470	1.8628	3.0234	0.2488	0.1800	1^{-2}	100	0.2068
10	0.9139	1.8572	7.9349	0.3606	0.1834	1^{-2}	10	0.2011
Mean	0.9430	1.8696	3.6239	0.2454	0.1834	N/A	N/A	N/A

B-3 Experiment 3

Table B-7: Results case study 3 for the delta method

Run	**Delta**						
	CICP	**MPIW**	**CWC**	**MSE**	**Time (s)**	λ_{reg}	$\hat{\sigma}_e^2$
1	0.9126	1.4496	7.9310	0.1683	0.0742	1^{-2}	0.1277
2	0.8835	1.3109	29.1166	0.1576	0.0761	1^{-4}	0.1027
3	0.9709	1.4648	1.4648	0.1720	0.0953	1^{-2}	0.1304
4	0.9223	1.4906	5.4794	0.1171	0.1086	1^{-2}	0.1342
5	0.9612	1.4041	1.4041	0.1779	0.0874	1^{-3}	0.1176
6	0.9612	1.3983	1.3983	0.1382	0.0927	1^{-3}	0.1167
7	0.9126	1.3729	7.8543	0.3088	0.1034	1^{-3}	0.1077
8	0.9320	1.4545	3.9093	0.1866	0.0947	1^{-2}	0.1269
9	0.9320	1.2987	3.7536	0.1570	0.0823	1^{-5}	0.0995
10	0.9223	1.4255	5.4143	0.1749	0.0812	1^{-2}	0.1236
Mean	0.9311	1.4070	6.7726	0.1758	0.0896	N/A	0.1187

Table B-8: Results case study 3 for the Bayesian method

Run	**Bayesian**						
	CICP	**MPIW**	**CWC**	**MSE**	**Time (s)**	ρ	$\bar{\sigma}_e^2$
1	1	2.9291	2.9291	0.2054	1.3807	5	0.5
2	1	2.9014	2.9014	0.1794	1.4374	1	0.5
3	0.9903	2.8852	2.8852	0.2095	1.3134	1	0.5
4	0.9806	2.9237	2.9237	0.2604	2.2564	1	0.5
5	1	2.9335	2.9335	0.1508	1.2548	1	0.5
6	1	2.9305	2.9305	0.1606	1.4160	1	0.5
7	0.9806	2.8934	2.8934	0.2748	1.2702	1	0.5
8	1	2.9830	2.9830	0.2308	1.2492	5	0.5
9	0.9515	1.3967	1.3967	0.1402	1.2923	50	0.1
10	1	2.8870	2.8870	0.1628	1.2595	1	0.5
Mean	0.9903	2.7664	2.7664	0.1975	1.3130	N/A	N/A

Table B-9: Results case study 3 for the SBC method

Run	**SBC**							
	CICP	**MPIW**	**CWC**	**MSE**	**Time (s)**	λ_{reg}	ρ	$\hat{\sigma}_e^2$
1	0.9126	1.4709	7.9523	0.1683	0.0774	1^{-2}	10	0.1332
2	0.9223	1.4529	5.4417	0.1576	0.0658	1^{-2}	1	0.1281
3	0.9515	1.4717	1.4717	0.1720	0.0731	1^{-2}	5	0.1314
4	0.9612	1.3896	1.3896	0.1171	0.0645	1^{-3}	50	0.1161
5	0.9612	1.4847	1.4847	0.1779	0.0664	1^{-2}	50	0.1328
6	0.9515	1.4796	1.4796	0.1382	0.0754	1^{-2}	50	0.1323
7	0.9320	1.5063	3.9611	0.2088	0.0649	1^{-2}	10	0.1245
8	0.9320	1.4510	3.9058	0.1866	0.0770	1^{-2}	100	0.1263
9	0.9417	1.4770	2.9878	0.1570	0.0777	1^{-2}	0.01	0.1333
10	0.9417	1.4695	2.9802	0.1749	0.0673	1^{-2}	50	0.1307
Mean	0.9408	1.4653	3.3055	0.1758	0.0710	N/A	N/A	0.1289

B-4 Experiment 4

Table B-10: Results case study 4 for the delta method with 95% nominal confidence level

Run	Delta						
	CICP	**MCIW**	**CWC**	**MR**	**Time (s)**	λ_{reg}	$\bar{\sigma}_e^2$
1	0.9901	1.4022	1.4022	0.0480	0.0654	1^{-7}	10
2	0.9714	1.0607	1.0607	0.0960	0.0967	1^{-4}	10
3	0.9881	1.3448	1.3448	0.0560	0.1242	1^{-5}	10
4	0.9623	1.1670	1.1670	0.1200	0.0865	1^{-5}	10
5	0.9643	1.6171	1.6171	0.0960	0.0775	1^{-6}	10
6	0.9867	1.0937	1.0937	0.1040	0.0727	1^{-5}	10
7	0.9464	1.0301	2.2256	0.1520	0.1022	1^{-4}	10
8	1	1.3409	1.3409	0.1200	0.1022	1^{-5}	10
9	0.9406	1.0912	2.6917	0.0800	0.0713	1^{-4}	10
10	0.9787	2.2564	2.2564	0.0960	0.1136	1^{-7}	10
Mean	0.9729	1.3404	1.6200	0.0936	0.0831	N/A	N/A

Table B-11: Results case study 4 for the Bayesian method with 95% nominal confidence level

Run	Bayesian						
	CICP	**MCIW**	**CWC**	**MR**	**Time (s)**	ρ	$\bar{\sigma}_e^2$
1	0.9737	0.6575	0.6575	0.0480	0.0622	5	1
2	0.9450	0.6835	1.9705	0.0960	0.0945	5	1
3	0.9839	1.8041	1.8041	0.0560	0.1408	10	10
4	0.9848	2.4241	2.4241	0.1120	0.0622	50	0.1
5	0.9626	0.9228	0.9448	0.0800	0.0658	10	1
6	0.9691	0.9170	0.9170	0.1120	0.0686	10	0.5
7	0.9182	0.6923	5.6005	0.1200	0.0719	50	1
8	0.9231	0.7280	4.5706	0.1280	0.1005	5	1
9	0.9478	0.8100	1.9249	0.0640	0.0773	50	1
10	0.9579	8.7952	8.7952	0.1200	0.0871	50	0.1
Mean	0.9566	1.8457	2.9609	0.0936	0.0831	N/A	N/A

Table B-12: Results case study 4 for the SBC method with 95% nominal confidence level

Run	SBC							
	CICP	MCIW	CWC	MR	Time (s)	λ_{reg}	ρ	$\bar{\sigma}_e^2$
1	0.9873	1.5205	1.5205	0.0480	0.1159	1^{-7}	0.001	10
2	0.9865	1.4143	1.4143	0.0960	0.0879	1^{-4}	0.0001	120
3	0.9732	0.9464	0.9464	0.0560	0.0988	1^{-5}	0.01	10
4	0.9412	0.9596	2.5141	0.1200	0.0649	1^{-5}	0.01	10
5	0.9545	0.6316	0.6316	0.0960	0.0657	1^{-6}	0.0001	10
6	0.9155	0.9550	6.5695	0.1040	0.0812	1^{-5}	0.001	10
7	0.9355	1.2998	3.3662	0.1520	0.0704	1^{-4}	0.0001	40
8	0.9773	1.3528	1.3528	0.1200	0.0913	1^{-5}	0.1	10
9	0.9571	0.9300	0.9300	0.0800	0.0879	1^{-4}	0.0001	80
10	0.9412	1.2522	2.8067	0.0960	0.0695	1^{-7}	0.001	10
Mean	0.9569	1.1262	2.2052	0.0936	0.0833	N/A	N/A	N/A

B-5 Experiment 5

Table B-13: Results case study 5 for the delta method with 95% nominal confidence level

Run	Delta						
	CICP	MCIW	CWC	MR	Time (s)	λ_{reg}	$\bar{\sigma}_e^2$
1	1	0.0703	0.0703	0	0.1135	1^{-5}	0.01
2	1	0.0722	0.0722	0	0.0750	1^{-5}	0.01
3	1	0.0717	0.0717	0	0.0721	1^{-5}	0.01
4	1	0.0714	0.0714	0	0.0779	1^{-5}	0.01
5	1	0.0703	0.0703	0	0.0808	1^{-5}	0.01
6	1	0.0733	0.0733	0	0.0725	1^{-5}	0.01
7	1	0.0807	0.0807	0	0.0702	1^{-5}	0.01
8	1	0.0676	0.0676	0	0.0739	1^{-5}	0.01
9	1	0.0773	0.0773	0	0.0849	1^{-6}	0.01
10	1	0.0732	0.0732	0	0.0797	1^{-5}	0.01
Mean	1	0.0728	0.0728	0	0.0800	N/A	N/A

Table B-14: Results case study 5 for the Bayesian method with 95% nominal confidence level

Run	Bayesian						
	CICP	MCIW	CWC	MR	Time (s)	ρ	$\bar{\sigma}_e^2$
1	1	0.0963	0.0963	0	0.1582	0.01	0.01
2	1	0.0971	0.0971	0	0.1425	0.01	0.01
3	1	0.0931	0.0931	0	0.1706	0.01	0.01
4	1	0.0986	0.0986	0	0.1453	0.01	0.01
5	1	0.0925	0.0925	0	0.2198	0.01	0.01
6	1	0.1085	0.1085	0	0.1433	0.01	0.01
7	1	0.1131	0.1131	0	0.1475	0.01	0.01
8	1	0.0937	0.0937	0	0.1486	0.01	0.01
9	1	0.1177	0.1177	0	0.1375	0.1	0.01
10	1	0.1135	0.1135	0	0.1664	0.01	0.01
Mean	1	0.1021	0.1021	0	0.1580	N/A	N/A

Table B-15: Results case study 5 for the SBC method with 95% nominal confidence level

Run	SBC							
	CICP	MCIW	CWC	MR	Time (s)	λ_{reg}	ρ	$\bar{\sigma}_e^2$
1	1	0.0484	0.0484	0	0.1324	1^{-5}	0.0001	0.01
2	1	0.0446	0.0446	0	0.0639	1^{-5}	0.0001	0.01
3	1	0.0408	0.0408	0	0.0554	1^{-5}	0.0001	0.01
4	1	0.0464	0.0464	0	0.0662	1^{-5}	0.0001	0.01
5	1	0.0479	0.0479	0	0.0612	1^{-5}	0.0001	0.01
6	1	0.0464	0.0464	0	0.0610	1^{-5}	0.0001	0.01
7	1	0.0584	0.0584	0	0.0624	1^{-5}	0.001	0.01
8	1	0.0433	0.0433	0	0.0598	1^{-5}	0.0001	0.01
9	1	0.0643	0.0643	0	0.0802	1^{-6}	0.1	0.01
10	1	0.0491	0.0491	0	0.0641	1^{-5}	0.0001	0.01
Mean	1	0.0490	0.0490	0	0.0707	N/A	N/A	N/A

B-6 Experiment 6

Table B-16: Results case study 6 for the delta method with 95% nominal confidence level

Run	Delta						
	CICP	MCIW	CWC	MR	Time (s)	λ_{reg}	$\bar{\sigma}_e^2$
1	1	2.6586	2.6586	0.1000	0.0528	1^{-2}	40
2	0.9767	2.9042	2.9042	0.1444	0.0568	1^{-2}	40
3	0.9583	3.9003	3.9003	0.1556	0.0584	1^{-2}	40
4	0.9615	1.7480	1.7480	0.1333	0.0671	1^{-3}	10
5	0.9412	3.1635	4.7181	0.0889	0.0619	1^{-3}	40
6	0.9714	5.6433	5.6433	0.1333	0.0449	1^{-5}	10
7	0.9600	2.5816	2.5816	0.1556	0.0620	1^{-2}	40
8	0.9091	1.7226	9.4552	0.1444	0.0742	1^{-4}	10
9	0.9636	2.1288	2.1288	0.1333	0.0555	1^{-4}	10
10	0.9000	1.6202	13.8027	0.1556	0.0825	1^{-3}	10
Mean	0.9542	2.8071	4.9541	0.1344	0.0616	N/A	N/A

Table B-17: Results case study 6 for the Bayesian method with 95% nominal confidence level

Run	Bayesian						
	CICP	MCIW	CWC	MR	Time (s)	ρ	$\bar{\sigma}_e^2$
1	0.9643	1.3804	1.3804	0.0889	0.3385	100	1
2	0.9241	1.1930	4.8530	0.1333	0.2867	10	1
3	0.9365	1.3818	3.3451	0.1556	0.4029	10	1
4	0.9688	1.2178	1.2178	0.1222	0.2953	50	1
5	0.9655	1.1585	1.1585	0.0667	0.2865	10	1
6	0.9194	2.0329	6.6616	0.1333	0.2861	100	1
7	0.9474	2.2495	3.3901	0.1111	0.3736	5	10
8	0.9167	1.2871	6.5815	0.1556	0.3914	50	1
9	0.9130	2.2432	8.5892	0.1333	0.3608	5	10
10	0.9054	1.1157	10.4130	0.1556	0.2837	10	1
Mean	0.9361	1.5260	4.7590	0.1256	0.3305	N/A	N/A

Table B-18: Results case study 6 for the SBC method with 95% nominal confidence level

Run	**SBC**							
	CICP	**MCIW**	**CWC**	**MR**	**Time (s)**	λ_{reg}	ρ	$\bar{\sigma}_e^2$
1	0.9756	1.2740	1.2740	0.1000	0.0556	1^{-2}	0.01	40
2	1	1.3237	1.3237	0.1444	0.0477	1^{-2}	0.01	40
3	0.9545	1.5517	1.5517	0.1556	0.0473	1^{-2}	10	10
4	1	1.6911	1.6911	0.1333	0.0581	1^{-3}	0.01	80
5	0.9808	1.6834	1.6834	0.0889	0.0490	1^{-3}	0.1	40
6	0.9600	2.2128	2.2128	0.1333	0.0494	1^{-5}	0.001	120
7	1	1.8774	1.8774	0.1556	0.0496	1^{-2}	0.1	80
8	0.9474	0.9394	2.0800	0.1444	0.0722	1^{-4}	0.0001	120
9	0.9111	1.4542	8.4439	0.1333	0.0742	1^{-4}	10	10
10	0.9688	1.6528	1.6528	0.1556	0.0690	1^{-3}	0.001	120
Mean	0.9698	1.5661	2.3791	0.1344	0.0572	N/A	N/A	N/A

Bibliography

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

[2] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[3] Cong Chen, Kim Batselier, Wenjian Yu, and Ngai Wong. Kernelized support tensor train machines. *Pattern Recognition*, 122:108337, 2022.

[4] Zhongming Chen, Kim Batselier, Johan AK Suykens, and Ngai Wong. Parallelized tensor train learning of polynomial classifiers. *IEEE transactions on neural networks and learning systems*, 29(10):4621–4632, 2017.

[5] Andrzej Cichocki, Namgil Lee, Ivan V Oseledets, A-H Phan, Qibin Zhao, and D Mandic. Low-rank tensor networks for dimensionality reduction and large-scale optimization problems: Perspectives and challenges part 1. *arXiv preprint arXiv:1609.00893*, 2016.

[6] Pierre Comon, Xavier Luciani, and André LF De Almeida. Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 23(7-8):393–405, 2009.

[7] Kris De Brabanter, Peter Karsmakers, Jos De Brabanter, Johan AK Suykens, and Bart De Moor. Confidence bands for least squares support vector machine classifiers: A regression approach. *Pattern Recognition*, 45(6):2280–2287, 2012.

[8] Louis De Branges. The stone-weierstrass theorem. *Proceedings of the American Mathematical Society*, 10(5):822–824, 1959.

[9] Richard D De Vleaux, Jennifer Schumi, Jason Schweinsberg, and Lyle H Ungar. Prediction intervals for neural networks via nonlinear regression. *Technometrics*, 40(4):273–282, 1998.

[10] Richard Dybowski and Stephen J Roberts. Confidence intervals and prediction intervals for feed-forward neural networks. In *Clinical Applications of Artificial Neural Networks*. Cambridge University Press, 2001.

[11] Carl Elkin and Sims Witherspoon. Machine learning can boost the value of wind energy. *deep mind field study*, 2019.

[12] Claudio Gambella, Bissan Ghaddar, and Joe Naoum-Sawaya. Optimization problems for machine learning: A survey. *European Journal of Operational Research*, 290(3):807–828, 2021.

[13] Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[14] Johan Håstad. Tensor rank is np-complete. In *Automata, Languages and Programming: 16th International Colloquium Stresa, Italy, July 11–15, 1989 Proceedings 16*, pages 451–460. Springer, 1989.

[15] Trevor Hastie, Robert Tibshirani, Jerome Friedman, Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Overview of supervised learning. *The elements of statistical learning: Data mining, inference, and prediction*, pages 9–41, 2009.

[16] Simon Haykin. *Kalman filtering and neural networks*. John Wiley & Sons, 2004.

[17] Håkan Hjalmarsson and Lennart Ljung. Estimating model variance in the case of undermodeling. *IEEE Transactions on Automatic Control*, 37(7):1004–1008, 1992.

[18] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713, 2012.

[19] Prateek Jain, Purushottam Kar, et al. Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–363, 2017.

[20] Jos De Brabanter Johan A. K. Suykens, Tony Van Gestel. *Least Squares Support Vector Machines*. River Edge, NJ: World Scientific, 2002.

[21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[22] Abbas Khosravi, Saeid Nahavandi, Doug Creighton, and Amir F Atiya. Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on neural networks*, 22(9):1341–1356, 2011.

[23] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[24] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

[25] Erik G Learned-Miller. Introduction to supervised learning. *I: Department of Computer Science, University of Massachusetts*, page 3, 2014.

[26] Subhash R Lele. How should we quantify uncertainty in statistical inference? *Frontiers in Ecology and Evolution*, 8:35, 2020.

[27] Ljung Lennart. System identification: theory for the user. *PTR Prentice Hall, Upper Saddle River, NJ*, 28:540, 1999.

[28] Ljung Lennart and Peter E Caines. Asymptotic normality of prediction error estimators for approximate system models. *Stochastics*, 3(1-4):29–46, 1980.

[29] Bing Liu and Bing Liu. *Supervised learning.* Springer, 2011.

[30] L Ljung. System identification-theory for the user 2nd edition ptr prentice-hall. *Upper Saddle River, NJ*, 1999.

[31] Magnus Malmström. Uncertainties in neural networks: A system identification approach. Master's thesis, Linköping University Electronic Press, 2021.

[32] Magnus Malmström, Isaac Skog, Daniel Axehill, and Fredrik Gustafsson. Asymptotic prediction error variance for feedforward neural networks. *IFAC-PapersOnLine*, 53(2):1108–1113, 2020.

[33] Eva Memmel, Clara Menzen, Jetze Schuurmans, Frederiek Wesel, and Kim Batselier. Towards green ai with tensor networks–sustainability and innovation enabled by efficient algorithms. *arXiv preprint arXiv:2205.12961*, 2022.

[34] Clara Menzen, Manon Kok, and Kim Batselier. Alternating linear scheme in a bayesian framework for low-rank tensor approximation. *SIAM Journal on Scientific Computing*, 44(3):A1116–A1144, 2022.

[35] Clara Menzen, Eva Memmel, Kim Batselier, and Manon Kok. Projecting basis functions with tensor networks for gaussian process regression. *Delft Center for Systems and Control, TU Delft, Netherlands*, 2023.

[36] Clara Menzen, Eva Memmel, Manon Kok, and Kim Batselier. Projecting basis functions with tensor networks for gaussian process regression. 2023.

[37] NOAA National Centers for Environmental Information. Climate at a Glance: Global Time Series, August 2023. Published.

[38] HB Pacejka and IJM Besselink. Magic formula tyre model with transient properties. *Vehicle system dynamics*, 27(S1):234–249, 1997.

[39] Georgios Papadopoulos, Peter J Edwards, and Alan F Murray. Confidence estimation methods for neural networks: A practical comparison. *IEEE transactions on neural networks*, 12(6):1278–1287, 2001.

[40] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

[41] Apostolos F Psaros, Xuhui Meng, Zongren Zou, Ling Guo, and George Em Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, page 111902, 2023.

[42] Simo Särkkä and Lennart Svensson. *Bayesian filtering and smoothing*, volume 17. Cambridge university press, 2023.

[43] G Scheithauer. Jorge nocedal and stephen j. wright: Numerical optimization.

[44] Johan Schoukens and Lennart Ljung. Nonlinear system identification: A user-oriented road map. *IEEE Control Systems Magazine*, 39(6):28–99, 2019.

[45] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.

[46] Knvul Sheikh. A growing presence on the farm: robots. *The New York Times*, 13, 2020.

[47] Talha Siddique, Md Shaad Mahmud, Amy M Keesee, Chigomezyo M Ngwira, and Hyunju Connor. A survey of uncertainty quantification in machine learning for space weather prediction. *Geosciences*, 12(1):27, 2022.

[48] E Miles Stoudenmire. Learning relevant features of data with multi-scale tensor networks. *Quantum Science and Technology*, 3(3):034003, 2018.

[49] Edwin Stoudenmire and David J Schwab. Supervised learning with tensor networks. *Advances in Neural Information Processing Systems*, 29, 2016.

[50] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[51] Guglielmo Tamburrini. The ai carbon footprint and responsibilities of ai scientists. *Philosophies*, 7(1):4, 2022.

[52] Yingjie Tian and Yuqi Zhang. A comprehensive survey on regularization strategies in machine learning. *Information Fusion*, 80:146–166, 2022.

[53] E.A. van Mourik. All-at-once optimization for kernel machines with canonical polyadic decompositions: Enabling large scale learning for kernel machines. Master's thesis, Delft University of Technology, 2022.

[54] Warren E Walker, Poul Harremoës, Jan Rotmans, Jeroen P Van Der Sluijs, Marjolein BA Van Asselt, Peter Janssen, and Martin P Krayer von Krauss. Defining uncertainty: a conceptual basis for uncertainty management in model-based decision support. *Integrated assessment*, 4(1):5–17, 2003.

[55] Frederiek Wesel and Kim Batselier. Large-scale learning with fourier features and tensor decompositions. *Advances in Neural Information Processing Systems*, 34:17543–17554, 2021.

[56] Frederiek Wesel and Kim Batselier. Quantized fourier and polynomial features for more expressive tensor network models. *arXiv preprint arXiv:2309.05436*, 2023.

[57] Stephen J Wright. Coordinate descent algorithms. *Mathematical programming*, 151(1):3–34, 2015.

[58] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4:795–813, 2022.

[59] Changjiang Yang, Ramani Duraiswami, and Larry S Davis. Efficient kernel machines using the improved fast gauss transform. *Advances in neural information processing systems*, 17, 2004.

Glossary

List of Acronyms

AI	Artificial intelligence
CP-ALS	Canonical Polyadic Decomposition Alternating Linear Scheme
CPD	Canonical Polyadic Decomposition
MSE	mean squared error
MR	misclassification rate
ALS	Alternating Linear Scheme
SBC	Single Bayesian Core
SNR	signal-to-noise ratio
KKR	Kernel Ridge Regression
PI	prediction interval
CI	confidence interval
PICP	prediction interval coverage probability
MPIW	mean prediction Interval width
CWC	coverage width-based criterion
CICP	confidence interval coverage probability
MCIW	mean confidence Interval width
UT	Unscented Transform
WR	wrongly classified labels outside the confidence interval
RI	rightly classified labels outside the confidence interval

List of Symbols

Notation	Definition
a	Scalar
$\mathbf{A}$	Matrix
$\boldsymbol{\mathcal{A}}$	Tensor
i_d	Index in dimension d of a tensor
I_d	Size of dimension d of a tensor
$\boldsymbol{\mathcal{A}}(i_1, i_2, \ldots, i_D) = a_{i_1,i_2,\ldots,i_D}$	Element of a tensor
$\text{vec}(\boldsymbol{\mathcal{A}})$	Vectorization of tensor $\boldsymbol{\mathcal{A}}$
$\otimes$	Kronecker product
$*$	Hadamard product
$\odot$	Khatri-Rao product
$\bigotimes_{d=1}^{D}$	Sequence of Kronecker products
$\bigodot_{d=D}^{1}$	Sequence of Khatri-Rao products
$\langle \cdot, \cdot \rangle_F$	Frobenius inner product
$\|\| \cdot \|\|$	Frobenius norm
$\mathcal{O}(\cdot)$	Big O notation
$\mathcal{X}$	Original input space
$\mathcal{H}$	Reproducing Kernel Hilbert Space
$\mathcal{L}$	Loss function
$\boldsymbol{\lambda}$	Norm vector of the CPD
λ_{reg}	Regularization hyperparameter
R	CP-rank
M	Dimension of total feature map
$\hat{M}$	Dimension of each feature map
$R(\cdot)$	Regularization term
N	Number of samples
D	Number of dimensions
I	Size of a dimension
$R, \quad (R_1, \ldots, R_N)$	tensor rank R and multilinear rank
$\mathbf{x}$	Input vector
$\mathbf{X}$	Input matrix
$\mathbf{x}_n$	Row of input matrix
$x_n^{(d)}$	Element of input matrix
$\mathbf{z}_d(x_n^{(d)})$	Nonlinear mapping of element $x_n^{(d)}$
$\mathcal{Z}(\cdot)$	Tensorized multidimensional feature mapping
$\mathbf{z}_* = \mathbf{z}(\mathbf{x}_{n*})$	test input in feature space
y_n	Target value
$\mathbf{y}$	Vector of target values
e_n	Random error variable
$\mathbf{e}$	Vector of random error variables
σ_e^2	Noise variance
$\hat{\sigma}_e^2$	Estimated noise variance
$\bar{\sigma}_e^2$	Hyperparameter noise variance
$\mathbb{V}(\hat{f})$	Prediction error variance
$\mathbb{V}(f_*)$	Estimated test input variance
$\mathbf{w}$	Parameter vector

$\boldsymbol{\phi}(\cdot)$	Multidimensional nonlinear feature map
$\mathbf{\Phi}$	Matrix of nonlinear feature mappings
∇	Gradient
V_N	Cost function
$p(\cdot)$	Probability distribution
$\mathcal{N}$	Normal distribution
τ	Unscented transform scaling parameter
κ	Unscented transform scaling parameter
α_{UT}	Unscented transform scaling parameter
∇	Gradient
η	Learning-rate parameter
$\mathbf{J}$	Jacobian for training samples
$\mathbf{g}$	Jacobian for test samples
$\mathbf{C}^{(d)}$	Derivative of the model with respect to the *d*-th factor
$\mathbf{H}$	Hessian
$\mathbf{y}_{\text{scale}}$	Scaled outputs
ϵ_n	Error between observation and prediction for sample n
σ_f^2	Output variance hyperparameter
$\mathbf{x}_*$	Vector of test inputs
$\hat{\mathbf{w}}$	Estimated parameters
$\mathbf{w}^*$	Asymptotic value of the parameter estimate
$\mathbf{w}^0$	Model parameters for the true system
$\mathbf{w}^b$	Bias in the parameters
$f^*(\mathbf{x}_n)$	Function that describes the true system
$\mathbb{E}(\hat{f})$	Mean vector of predictive distribution
$\mathbb{E}(\hat{f}_n)$	Mean prediction for the n-th sample
f^{ref}	Reference model
$f(\mathbf{x}_n, \hat{\mathbf{w}})$	Vector of predictions for the n-th sample
$\epsilon_n(\mathbf{x}_n, \hat{\mathbf{w}})$	Prediction error
$\overline{\mathbb{E}}[\cdot]$	Asymptotic mean
$\mathbf{P}$	Covariance matrix
$\mathcal{M}$	Model structure
$\mathcal{I}_{\mathbf{w}}$	Fisher information matrix
$\mathbf{m}_0$, $\mathbf{m}_N$	Prior and posterior mean
$\mathbf{P}_0$, $\mathbf{P}_N$	Prior and posterior covariance
$\mathbf{x}^{(\cdot)}$	Sigma points
$\mathbf{s}^{(\cdot)}$	Transformed sigma points
$\hat{f}$	Taylor series approximation of a nonlinear function
$\mathbf{w}_{LS}$	Least squares estimate
$\mathbf{w}_{MAP}$	Maximum posterior weight vector
$\propto$	Proportional to
$p(\mathbf{g}_d)$	Prior distribution of TN component
$p(\mathbf{y}\|\{\mathbf{g}_d\})$	Likelihood
$p(\{\mathbf{g}_d\}\|\mathbf{y})$	Posterior joint distribution of TN components
α	Significance level

p	Proportion in K-fold cross-validation procedure
μ	Confidence level (1-α)
γ	First CWC hyperparameter
η	Second CWC hyperparameter
ρ	Prior covariance scaling hyperparameter