Uncertainties in Neural Networks

A System Identification Approach

Magnus Malmström

Uncertainties in Neural Networks

A System Identification Approach

Magnus Malmström

This is a Swedish Licentiate's Thesis.

Swedish postgraduate education leads to a Doctor's degree and/or a Licentiate's degree.

A Doctor's Degree comprises 240 ECTS credits (4 years of full-time studies).

A Licentiate's degree comprises 120 ECTS credits,
of which at least 60 ECTS credits constitute a Licentiate's thesis.

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

<https://creativecommons.org/licenses/by-nc/4.0/>

Linköping studies in science and technology. Licentiate Thesis
No. 1902

Uncertainties in Neural Networks: A System Identification Approach

Magnus Malmström

magnus.malmstrom@liu.se

www.control.isy.liu.se

Department of Electrical Engineering

Linköping University

SE-581 83 Linköping

Sweden

ISBN 978-91-7929-680-3

ISSN 0280-7971

Copyright © 2021 Magnus Malmström

Printed by LiU-Tryck, Linköping, Sweden 2021

To my family and friends!

“The best material model of a cat is another, or preferably the same, cat.”
Rosenblueth and Wiener [1945]

Abstract

In science, technology, and engineering, creating models of the environment to predict future events has always been a key component. The models could be everything from how the friction of a tire depends on the wheels slip to how a pathogen is spread throughout society. As more data becomes available, the use of data-driven black-box models becomes more attractive. In many areas they have shown promising results, but for them to be used widespread in safety-critical applications such as autonomous driving some notion of uncertainty in the prediction is required.

An example of such a black-box model is neural networks (NNs). This thesis aims to increase the usefulness of NNs by presenting a method where uncertainty in the prediction is obtained by linearization of the model. In system identification and sensor fusion, under the condition that the model structure is identifiable, this is a commonly used approach to get uncertainty in the prediction from a nonlinear model. If the model structure is not identifiable, such as for NNs, the ambiguities that cause this have to be taken care of in order to make the approach applicable. This is handled in the first part of the thesis where NNs are analyzed from a system identification perspective, and sources of uncertainty are discussed.

Another problem with data-driven black-box models is that it is difficult to know how flexible the model needs to be in order to correctly model the true system. One solution to this problem is to use a model that is more flexible than necessary to make sure that the model is flexible enough. But how would that extra flexibility affect the uncertainty in the prediction? This is handled in the later part of the thesis where it is shown that the uncertainty in the prediction is bounded from below by the uncertainty in the prediction of the model with lowest flexibility required for representing true system accurately.

In the literature, many other approaches to handle the uncertainty in predictions by NNs have been suggested, of which some are summarized in this work. Furthermore, a simulation and an experimental studies inspired by autonomous driving are conducted. In the simulation study, different sources of uncertainty are investigated, as well as how large the uncertainty in the predictions by NNs are in areas without training data. In the experimental study, the uncertainty in predictions done by different models are investigated. The results show that, compared to existing methods, the linearization method produces similar results for the uncertainty in predictions by NNs.

Populärvetenskaplig sammanfattning

Inom forskning och utveckling har det alltid varit centralt att skapa modeller av verkligheten. Dessa modeller har bland annat använts till att förutspå framtida händelser eller för att styra ett system till att bete sig som man önskar. Modellerna kan beskriva allt från hur friktionen hos ett bildäck påverkas av hur mycket hjulen glider till hur ett virus kan sprida sig i ett samhälle. I takt med att mer och mer data blir tillgänglig ökar potentialen för datadrivna *black-box* modeller. Dessa modeller är universella approximationer vilka ska kunna representera vilken godtycklig funktion som helst. Användningen av dessa modeller har haft stor framgång inom många områden men för att verkligen kunna etablera sig inom säkerhetskritiska områden såsom självkörande farkoster behövs en förståelse för osäkerhet i prediktionen från modellen.

Neuronnät är ett exempel på en sådan *black-box* modell. I denna avhandling kommer olika sätt att tillförskaffa sig kunskap om osäkerhet i prediktionen av neuronnät undersökas. En metod som bygger på linjärisering av modellen för att tillförskaffa sig osäkerhet i prediktionen av neuronnätet kommer att presenteras. Denna metod är välbeprövad inom systemidentifiering och sensorfusion under antagandet att modellen är identifierbar. För modeller såsom neuronnät, vilka inte är identifierbara behövs det att det tas hänsyn till tvetydigheterna i modellen.

En annan utmaning med datadrivna *black-box* modeller, är att veta om den valda modellmängden är tillräckligt generell för att kunna modellera det sanna systemet. En lösning på detta problem är att använda modeller som har mer flexibilitet än vad som behövs, det vill säga en överparameteriserad modell. Men hur påverkas osäkerheten i prediktionen av detta? Detta är något som undersöks i denna avhandling, vilken visar att osäkerheten i den överparameteriserade modellen kommer att vara begränsad underifrån av modellen med minst flexibilitet som ändå är tillräckligt generell för att modellera det sanna systemet. Som avslutning kommer dessa resultat att demonstreras i både en simuleringsstudie och en experimentstudie inspirerad av självkörande farkoster. Fokuset i simuleringsstudien är hur osäkerheten hos modellen är i områden med och utan tillgång till träningsdata medan experimentstudien fokuserar på jämförelsen mellan osäkerheten i olika typer av modeller. Resultaten från dessa studier visar att metoden som bygger på linjärisering ger liknande resultat för skattningen av osäkerheten i prediktionen av neuronnät, jämfört med existerande metoder.

Acknowledgments

First and foremost, I would like to thank my supervisor Prof. Fredrik Gustafsson and my co-supervisors Asst. Prof. Isaac Skog and Assoc. Prof. Daniel Axehill for the continuous support and encouragement during these three years. I am very grateful for your guidance on my journey to become a researcher, all the interesting discussions during our meetings, and help in improving my scientific language and communication skills.

I would also like to thank Assoc. Prof. Martin Enqvist, head of division at Automatic Control for giving me the opportunity to start work at Automatic Control group, as well as maintaining a friendly and professional work environment and Ninna Stensgård for taking care of all the practicalities. Furthermore, I would like to thank Assoc. Prof. Gustaf Hendeby for providing the template used to write this thesis.

This thesis has also been greatly improved by the proof reading done by Lic. Kritoffer Bergman, Lic. Per Boström-Rost, Lic. Robin Forsling, M.Sc. Anjs Hellander, M.Sc. Anton Kullberg, Lic. Fredrik Ljungberg, M.Sc. Kristin Nielsen, M.Sc. Alberto Zenere. I am very grateful for all your feedback, without it, this thesis would not have been stabilized to an equilibrium. Thank you!

I would like to thank all my current and past colleagues at the Automatic Control group for contributing to such a great and friendly working environment. For all interesting discussions, both around the tables in the fika room but also during the online fika breaks, and for the after-work and other activities.

This work is supported by Sweden's innovation agency, Vinnova, through project iQDeep (project number 2018-02700). Their funding is gratefully acknowledged. The project is a collaboration between LiU and Scania, and I would like to thank everyone involved in the project, especially the AI Technologies group at Scania (EARA) for interesting discussions and input to my work, both at the annual meetings and during the visits in Södertälje.

Last but not least I would like to thank my family and friends for their love and support during this journey. Without your support, this would not have been possible.

*Linköping, March 2021
Magnus Malmström*

Contents

Notation	xiii
1 Introduction	1
1.1 Learning in autonomous vehicles	1
1.2 Previous work	4
1.3 Contributions	5
1.4 Thesis outline	6
2 System identification using neural networks	9
2.1 Problem formulation	9
2.2 Model sets and model structure	10
2.2.1 Linear models	11
2.2.2 Neural networks	13
2.2.3 Model order selection	16
2.3 Parameter optimization	16
2.3.1 Newton method	17
2.3.2 Gauss-Newton method	18
2.4 Parameter optimization for neural networks	19
2.4.1 RMS-prop and ADAM optimizer	21
2.4.2 Initialization of neural networks	21
2.4.3 Backpropagation	22
2.4.4 Regularization	22
2.5 Hessian and parameter covariance	24
2.6 Uniqueness	25
2.6.1 Canonical representation of a model	25
2.6.2 Symmetries in neural networks	26
2.7 Regularity conditions and identifiability	31
2.7.1 Identifiability and informativity	31
2.7.2 Implications for neural networks	32
2.8 Different sources of uncertainty	32
3 Related approaches to quantifying uncertainties in neural networks	37
3.1 Linearization	37

3.2	Ensemble techniques	38
3.2.1	Monte Carlo dropout	38
3.2.2	Monte Carlo batchnorm	39
3.3	Learning prediction uncertainties in neural networks	39
3.3.1	Gaussian model	39
3.3.2	Energy-based model	40
3.4	Gaussian processes	41
3.5	Bayesian neural networks	42
4	Parameter and prediction uncertainty for canonical neural networks	45
4.1	Parameter covariance	45
4.1.1	True system not in model set	46
4.1.2	True system in model set	48
4.1.3	Cramér-Rao lower bound	48
4.1.4	Approximating the variance expressions	49
4.1.5	Parameter covariance for irreducible neural networks	49
4.2	Prediction error variance	52
4.3	Summary	54
5	Prediction error variance for overparameterized models	55
5.1	Overparameterization in neural networks	55
5.2	Models with redundant parameters	56
5.2.1	Linear regression	59
5.2.2	Nonlinear regression	65
5.3	Models with more flexibility	69
5.3.1	Linear regression	71
5.3.2	Nonlinear regression	74
5.4	Summary	76
6	Experimental and simulation study	77
6.1	Tire-road friction	77
6.1.1	Data processing	78
6.1.2	Magic tire formula with bias	78
6.2	Simulation study	79
6.2.1	Simulation setup	79
6.2.2	Relation to previous work	80
6.2.3	Model and estimation error	82
6.2.4	Prediction error uncertainty	82
6.2.5	Asymptotic results	84
6.3	Experimental study	86
6.3.1	Model order selection	87
6.3.2	Results	87
6.4	Summary	94
7	Conclusion	95
	Bibliography	97

Notation

ABBREVIATIONS

Abbreviation	Meaning
ABS	Anit-lock braking system
ACC	Adaptive cruise controller
ADAS	Advanced driver-assistance systems
AIC	Akaike’s information criterion
BIC	Bayesian information criterion
BNN	Bayesian neural network
CRLB	Cramér-Rao lower bound
GP	Gaussian process
LLSS	Linear least squares solution
LM	Levenberg-Marquardt algorithm
MC	Monte Carlo
MLE	Maximum likelihood estimate
NN	Neural network
NTF	Normalized traction force
SNR	Signal to noise ratio
SVD	Singular value decomposition

NEURAL NETWORK SPECIFIC NOTATION

Notation	Meaning
$\mathbf{a}^{(l)}$	Linear combination of the activation function of the $(l - 1)$ th layer to the l th layer
$\mathbf{h}^{(l)}$	Activation of the nodes in the l th hidden layer
$W^{(l)}$	Parameters that connect the l th and the $(l - 1)$ th hidden layers
$\sigma(\cdot)$	Activation function

PARAMETER ESTIMATION AND SYSTEM IDENTIFICATION

Notation	Meaning
$\mathbf{x}_m$	Vector at time index m
$\mathbf{y}_N$	Vector consisting of N observations
$\hat{s}_m$	Estimate of the signal s at time m
$f^*(\mathbf{x}_m)$	Function of $\mathbf{x}_m$ that describes the true system
$f(\mathbf{x}_m, \boldsymbol{\theta})$	Parametric model with parameter $\boldsymbol{\theta}$
$\varepsilon(\mathbf{x}_m, \boldsymbol{\theta})$	Prediction error
$f_c(\mathbf{x}_m, \boldsymbol{\theta}^c)$	Model on canonical form
$\varepsilon_c(\mathbf{x}_m, \boldsymbol{\theta}^c)$	Prediction error of canonical model
$\psi(\mathbf{x}_m, \boldsymbol{\theta}^c)$	Derivative of the canonical model with respect to the canonical parameters
$V_N(\boldsymbol{\theta})$	Cost function depending of $\boldsymbol{\theta}$ given N observations
$\boldsymbol{\theta}^0$	Model parameters for the true system
$\hat{\boldsymbol{\theta}}_N$	Parameter estimate that minimizes $V_N(\boldsymbol{\theta})$
$\boldsymbol{\theta}^*$	Asymptotic value of the parameter estimate
$\boldsymbol{\theta}^c$	Parameter for the canonical model
$\boldsymbol{\theta}^i$	Non-canonical parameters
$\boldsymbol{\theta}^{(j)}$	The j iteration of the parameter estimate, or the j th sample of the parameter
$T_i(\boldsymbol{\theta}^i)$	Transformation taking the non-canonical parameters to the canonical
$T'_{i,\boldsymbol{\theta}}(\boldsymbol{\theta}^{i0})$	Derivative of transformation taking the non-canonical parameters to the canonical with respect to the non-canonical parameters
$\boldsymbol{\varphi}_m$	Regressor vector at time index m
Φ_N	Matrix composed of a collection of N regression vectors
λ_0	True variance of the prediction error
$\hat{\lambda}_N$	Estimated variance of the prediction error
$\mathcal{I}_{\boldsymbol{\theta}}$	Fisher information matrix for the parameter vector $\boldsymbol{\theta}$
$\epsilon(\mathbf{x}_m, y_m)$	Energy function

SOME SETS

Notation	Meaning
$\mathbb{R}$	Set of real numbers
$\mathcal{M}^*$	Model set
$\mathcal{M}$	Model structure
$\mathcal{S}$	True system
Θ	Set of parameters θ
$\mathcal{D}_{\hat{\theta}}$	Set of parameters that minimizes a given cost function
$\mathcal{D}_{\theta^o}$	Set of parameters that describes the input output relationship perfectly
$\mathcal{D}_{\theta^*}$	Set of parameters that gives the best fit in the least squares sense

MATHEMATICS

Notation	Meaning
x	Scalar
$\mathbf{x}$	Vector
X	Matrix
$[\mathbf{x}]_i$	The i th element of a vector
$\mathbf{x}^\top$	Transpose
$[X]_{j,i}$	The element on the j th column and i th the row of a matrix
X^{-1}	Matrix inverse
X^+	Moore-Penrose matrix inverse
$E[\mathbf{x}]$	Expectation of $\mathbf{x}$
$\text{Var}(x)$	Variance of x
$\text{Cov}(\mathbf{x})$	Covariance of $\mathbf{x}$
$\bar{V}$	Asymptotic mean of V

1

Introduction

In science, a central component has always been to create models of the reality around us that can be used to predict future behaviour of a system and draw conclusions from them. This is especially true in the control community where mathematical models play a crucial role in making sure that a system behaves as intended when a control signal is applied. Traditionally, these models have been the result of hours of modeling of reality using physical insight, where the models are validated through numerous experiments. As the amount of available data in society increases, it opens up for a new sort of model which rather than using physical insights, is built using data. These models are often referred to as *black-box models*, and neural networks (NNs) are one example of such model.

As Box [1976] famously said “*All models are wrong, but some are useful*”, so how can one make sure that the black-box models intended to be used are useful, and that one doesn’t just create models for the sake of it? The ability to express some sort of uncertainty in the prediction of the model is one essential component that drastically increases the usefulness of a given model. This is especially true if the models are aimed to be used in safety-critical applications such as autonomous vehicles. Hence, this thesis aims to provide some insight into how the uncertainty in the prediction of one such model, namely NN, can be expressed. It is also demonstrated how this uncertainty in the prediction is compared to other black-box models where the uncertainty in the model comes more naturally. The results will be demonstrated both using simulated data, as well as data collected from experiments.

1.1 Learning in autonomous vehicles

Autonomous vehicles is one application where black-box models, such as NNs, are a very tempting alternative to use. To create a decision tree or another model

that covers all corner-cases that can happen in the traffic is extremely time consuming and infeasible in practice. On the other hand, one could fairly easily collect data from driving tests from which a black-box model could be built. With a lot of data available, NNs have had huge success in solving very hard regression and classification problems, and have become the standard approach in disciplines such as computer vision, [Krizhevsky et al., 2012], speech recognition, [Hinton et al., 2012a], and natural language processing [Mikolov et al., 2013]. They have also showed promising result in applications such as 5G positioning, [Malmström et al., 2019], modelling chemical properties of the atom, [Gilmer et al., 2017], and creating images from noise, [Goodfellow et al., 2014]. Despite this huge success, their use in safety-critical applications, such as autonomous vehicles, is limited, [Bagloee et al., 2016, Grigorescu et al., 2020], even though successful implementations of self-driving cars, e.g., [Codevilla et al., 2018] in simulation with programs such as CARLA, developed by Dosovitskiy et al. [2017].

Modern cars and early demonstrated autonomous cars have an abundance of sensor information, which are used to implement applications such as autonomous line following. In the future, NN based algorithms might also be used in applications such as platooning, overtaking of other cars, and advanced driver-assistance systems (ADAS). One major task of ADAS is to estimate the friction level of the road. The friction level can influence the safety distance in an adaptive cruise controller (ACC), which avoidance maneuvers that are most effective, and vehicular dynamics control systems such as the anti-lock braking system (ABS).

Example 1.1

The estimation of the normalized traction force (NTF) on a road based on the wheel slip is a famously difficult problem. Extensive modeling by, e.g., Pacejka and Besselink [1997], has lead to a parametric model called the *magic tire formula* that describes how the NTF, μ , depends on the wheel slip, s , i.e.,

$$\mu = D \sin \left(C \arctan \left((1 - E)s + E/B \arctan(Bs) \right) \right), \quad (1.1)$$

where parameters C, E, D, E depend on the surface. For further description of the magic tire formula, see Section 6.1.

This relationship can also be modeled using a black-box. e.g., consider

$$f(s, \theta) = \theta_5 \sigma(\theta_1 s + \theta_3) + \theta_6 \sigma(\theta_2 s + \theta_4) + \theta_7, \quad (1.2)$$

i.e., an NN with fairly low complexity. This is shown in Figure 1.1, where the NN models capture the relationship between the NTF and wheel slip.

But how certain is the algorithm that the prediction is correct when it comes to uncommon critical situations such as how hard the autonomous car has to push the brake to avoid colliding with a moose that is crossing the road? The setup in Example 1.1 is used throughout the thesis, where the question of how certainty of the NN is answered by providing confidence bounds on the prediction.

Figure 1.1: Estimation of friction level by a two-layer NN.

Lately, some companies have gone even further than simulation, and implemented fully autonomous vehicles, e.g., *Tesla*, *Waymo*, *Embark Trucks Inc.*, *Scania* and *Einride* to mention a few. One example is the self-driving bus currently tested at Scania seen in Figure 1.2. There are already some fully autonomous vehicles on our roads, but to use of black-box models such as NN in fully autonomous vehicles comes with a huge risk. This is due to their lack of ability to assess the *uncertainty* of the computed predictions. It has already been a couple of accidents with fatal outcome, e.g. [BBC News, 31 March 2018, NTSB, 2018] and [Reuters, 21 Aug 2017]. If the NN would include uncertainty in these predictions, these kinds of accidents could have been prevented, or at least they might not have led to a fatal outcome. There have also been demonstrated how vulnerable NNs are to adversarial attacks. For example, Eykholt et al. [2018] show that with small adjustments to the input image, an NN misclassified a stop-sign as a speed limit sign.

To separate the effect on the uncertainty in the training data¹ and the uncertainty in the model, the notions of aleatoric and epistemic uncertainty are often used. The epistemic uncertainty can e.g., be seen in uncertainty in the parameter in the NN and later propagated to uncertainty in the output of the model. It can also be used to indicate if the algorithm operates in an area where training data exists (interpolates) or where it does not (extrapolates).

¹ Generally, when referring to uncertainties in the training data only uncertainties in the dependent variables are considered, and the independent variables are assumed to be perfectly known.

Figure 1.2: One of the early autonomous buses developed by Scania.

1.2 Previous work

Since including concepts of certainty in black-box models such as NN could make them much more useful in applications such as autonomous vehicles the subject of assessing uncertainty in the prediction of NNs have recently received increased attention [Garnelo et al., 2018, Ghahramani, 2015, Kendall and Cipolla, 2016, Kendall and Gal, 2017]. Bayesian NNs (BNNs), which by design provide uncertainty measures, were proposed already in the 1990's [MacKay, 1992, Neal, 1996]. However, they were known to be computationally intensive and did not scale well with network size. Recently, some more efficient implementations have been proposed [Blundell et al., 2015, Gustafsson et al., 2019b, Kendall and Cipolla, 2016, Osband et al., 2016].

Most implementations of the BNN rely on creating ensembles of NNs, from which the network parameters can be sampled. A recent trend is to use dropouts or batchnorm to create those ensembles, e.g., [Gal and Ghahramani, 2016, Teye et al., 2018]. In simple terms, dropouts involve randomly disabling nodes in an NN, which provides an ensemble of models. In this way, outputs can be obtained in a Monte Carlo (MC) like fashion, from which the uncertainty in the predictions can be assessed. The advantage of these methods is that they rely on existing NN structures and are easy to implement. The downside is that multiple forward passes are required to create the MC-like simulations needed to obtain the uncertainty in the prediction. Another suggested approach is to include the variance as a component in the loss function for the NN, and hence, the NN learns its uncertainty, see e.g., [Eldesokey et al., 2018, Gustafsson et al., 2019a, Kendall and Gal, 2017]. However, this increases the size of the NN and thus the computational complexity. The inclusion of uncertainty, is done either by assuming a given probability distribution for the output and finding the values of the distribution parameters, or by learning the full probability density function (PDF). However, this also leads to the question of how certain it is that the selected distribution is correct and what the uncertainty of the learned PDF is, i.e., what is the *uncertainty of the learned uncertainty*?

Yet another approach is to estimate the uncertainty in the parameters of the NNs from the training phase, and then let this uncertainty be propagated to uncertainty in the output in the prediction step. Several publications, [He and Li, 2011, Hwang and Ding, 1997, Papadopoulos et al., 2001] have proposed this as a mean to get a variance expression for the prediction. Recently, similar ideas have been used to investigate what impact a single training example might have on the outcome in the testing phase, e.g., [Koh and Liang, 2017].

System identification, where sensor data is used to estimate and model different physical systems, is an established and well-developed field where the uncertainty of the computed predictions is considered, [Ljung, 1999]. Even if one sensor modality is uncertain about what it observes, there are often alternative ones to support the decisions or to close the feedback loops. This holds, given that the system knows that the currently used modality is uncertain about what it observes. The knowledge of the relative uncertainty of different sensor modalities is also the cornerstone of sensor fusion theory, [Gustafsson, 2018]. Since modern cars and early demonstrated autonomous cars have an abundance of sensor information, this is an example where the knowledge of the relative uncertainty can be of specific interest.

Recently, the connection between NN and nonlinear system identification of black-box model has been highlighted, e.g., [Andersson et al., 2019, 11-19 Dec, Ljung et al., 2020, Schoukens and Ljung, 2019]. In Ljung et al. [2020] and Andersson et al. [2019, 11-19 Dec] feedforward NNs are presented as a special case of the nonlinear autoregressive models (NARX) where multiple NARX models are stacked on top of each other. Ljung et al. [2020] also point out that recurrent NNs (RNNs) are a special case of nonlinear state space models. The authors also provide simulated results of how the number of breakpoints in an NN model increases exponentially as the number of layers increases in the NN. This strengthens the statement that deeper NN models are more flexible compared to shallow NNs with an equal number of nodes. Nonlinear system identification is thoroughly investigated in [Schoukens and Ljung, 2019] where the authors also point out the difficulties with uncertainty bounds for nonlinear models, namely, that if there is a structural model error, no theoretical bound of the uncertainty is available. One must rather rely on variability obtained from repeating the experiment with varying excitation, i.e., ensembles.

1.3 Contributions

This thesis aims to increase the usefulness of the nonlinear model NN in safety-critical applications by complementing the prediction of the model with some prediction error variance, also known as uncertainty in the prediction. This is done by linearizing the model to calculate uncertainties in the parameters which later can be propagated to uncertainties in the prediction. These techniques are fairly standard in the system identification field, but not in the machine learning field. By complementing one of the most common models used in the machine learning community, namely NN, with an uncertainty calculated using tech-

niques from system identification this thesis also aims to strengthen the connection between the different fields.

The analysis forces some restrictions on the NN, such as the uniqueness of parameterization. It is shown that if one would consider the prediction error variance, our analysis is also true for parameterizations with redundant parameters. There will also be shown how the prediction error variance is affected by using an overparameterized model. In Chapter 3, a survey of how uncertainty in prediction by NNs has been handled in the literature will be provided, and where the suggested approach is placed in relation to work previously done.

Using the suggested approach, the contribution of different sources of uncertainty will be demonstrated, as well as a comparison of the calculated uncertainty and the uncertainty obtained from MC simulations. Since the theoretical result is asymptotic in the number of observations, an analysis of how many observations that are needed for the results to be valid will also be provided.

As a final contribution, the uncertainty of the model set including NNs will be compared to other model structures, that might either be linear in the parameters, incorporate the concepts of uncertainty more naturally, or be obtained from extensive physical modeling of the system. To do this comparison, data collected from experiments inspired by autonomous driving is used.

Some of the results in this thesis have been published before, e.g., some of the results in Chapter 4 and the simulations in Chapter 6 regarding the different sorts of uncertainty and the prediction error variance for different structures of NNs have also been published in

Magnus Malmström, Isaac Skog, Daniel Axehill, and Fredrik Gustafsson. Asymptotic Prediction Error Variance for Feedforward Neural Networks. In *Proc. of 21th IFAC World Congress, (IFAC)*, Online (Berlin, Germany), 2020. 11-17 Jul.

1.4 Thesis outline

In Chapter 2 the problem setup will be described. This includes a description of NN models and the estimation problem of finding the best such model. Some regularity conditions on the data and model sets will be presented which are needed for our analysis later in the thesis. Some of the challenges using the model sets consisting of NNs are also covered. This chapter will be concluded by discussing what different sources of uncertainty that exist and how they contribute to the model uncertainty.

In Chapter 3, a survey of different ways uncertainty in black-box models are handled in the literature is presented. Chapter 4 provides the main result regarding how uncertainty from model sets, where NN are included, can be obtained by linearization with respect to the parameters of the NN. These results are under the assumption of uniqueness in the model, which might not always be true for NN.

Then later in Chapter 5, these assumptions are relaxed and it is shown that in order to estimate the prediction error variance, it is not necessary to consider

whether the model is uniquely parameterized or not. This chapter also provides some insight into how the uncertainty is affected when a model structure that is more flexible than necessary for the data it is intended to model is used, i.e., an overparameterized model.

In Chapter 6, the suggested approach to obtain prediction error variance is compared to commonly used approaches in the literature. This is using both simulation results and results on data collected from experiments. The thesis will be concluded with some conclusions and future work in Chapter 7.

2

System identification using neural networks

In this chapter, the task of training an NN is formulated as a problem of estimating parametric models from noisy observations. This will make it possible to use tools from estimation theory to obtain the uncertainty of the prediction made by the NN. Furthermore, a review of the concepts of model structure, model sets, the notion of a true system, uniqueness, and identifiability is given. The chapter is concluded with a discussion of different sources of uncertainties in the predictions made by NNS. Even though the concepts presented in this chapter are valid for general models, the special case of model based on NN is emphasized.

2.1 Problem formulation

Consider the following static nonlinear signal model

$$s_m = f^*(\mathbf{x}_m), \quad (2.1a)$$

$$y_m = s_m + e_m, \quad (2.1b)$$

where the function $f^*(\mathbf{x}_m)$ describes the relation between the input (independent variable) $\mathbf{x}_m \in \mathbb{R}^{d_x}$ and output (dependent variable) $s_m \in \mathbb{R}_{d_s}$ of the *true system* $\mathcal{S}$ under consideration, at time index m . The superscript $*$ is used to indicate that the function f describes the true system, i.e., $f^*(\mathbf{x}_m) = \mathcal{S}$. Furthermore, e_m is the observation noise which is identically and independently distributed according to some distribution with variance λ_0 and mean $E[e_m] = 0$, and $y_m \in \mathbb{R}_{d_s}$ is the observation of the output¹.

Given a set of data consisting of the observations $y_{1:N} \triangleq (y_n)_{n=1}^N$ and inputs

¹In this thesis, it is assumed that $d_s = 1$, hence the output does not have a vector notation (bold-face).

$\mathbf{x}_{1:N} \triangleq (\mathbf{x}_n)_{n=1}^N$, a parametric model on the form

$$\hat{s}_m = f(\mathbf{x}_m, \boldsymbol{\theta}), \quad (2.2)$$

with parameter vector $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^d$ containing d parameters is fitted to the data by for example minimizing a least-squares cost function. This is also referred to as learning the parameters, where the data used to learn the parameters is called training data. Hence, the parameters $\hat{\boldsymbol{\theta}}_N$ are estimated as

$$\hat{\boldsymbol{\theta}}_N = \arg \min_{\boldsymbol{\theta}} V_N(\boldsymbol{\theta}), \quad (2.3)$$

where

$$V_N(\boldsymbol{\theta}) \triangleq \frac{1}{N} \sum_{n=1}^N \|y_n - f(\mathbf{x}_n, \boldsymbol{\theta})\|^2. \quad (2.4)$$

The parametric model together with the estimated parameters are then used to predict new output $\hat{s}_m = f(\mathbf{x}_m, \hat{\boldsymbol{\theta}}_N)$ of the system given some input $\mathbf{x}_m$. In this thesis, the aim is to analyze how the prediction error

$$\varepsilon(\mathbf{x}_m, \hat{\boldsymbol{\theta}}_N) \triangleq y_m - \hat{s}_m = f^*(\mathbf{x}_m) + e_m - f(\mathbf{x}_m, \hat{\boldsymbol{\theta}}_N), \quad (2.5)$$

depends on the uncertainty in the training data (aleatoric uncertainties) and in the model (epistemic uncertainties), as well as to provide an analytic expression that can be used to calculate a lower bound on the prediction error uncertainty. NN model.

2.2 Model sets and model structure

To carry out the analysis of the prediction error variance and put the estimation of the parametric model $f(\mathbf{x}_m, \boldsymbol{\theta})$ in a system identification framework, the concept of models, model structure, model sets and true system used in e.g. Ljung [1999] need to be introduced and formalized. Thus, a *prediction model* is defined as:

Definition 2.1 (Prediction model). A prediction model $f(\mathbf{x}_{1:m}, y_{1:m-1})$ predicts the current output of a system given information about current and past inputs as well as past outputs.

The identification problem is to find a suitable model that can describe the measured data as well as possible. To find this model, a search over a set of candidate models is conducted, which leads us to the natural definition of *model set* as:

Definition 2.2 (Model set). A model set $\mathcal{M}^*$ is a collection of candidate models.

Example 2.1

One example of a model set is, e.g., the set of all linear models,

$$\mathcal{M}^* = \{\text{all linear models}\}. \quad (2.6a)$$

Another example is the set of all autoregressive models in which the regressor vector consists of output observations that are delayed at most n time steps, i.e., with lag of at most n ,

$$\mathcal{M}^* = \{f(y_{m-n-1:m-1}) = a_1 y_{m-1} + \cdots + a_n y_{m-n-1}, \quad a_i \in \mathbb{R}\}. \quad (2.6b)$$

A final example is a finite set of specific models

$$\mathcal{M}^* = \{f_1(\mathbf{x}_{1:m}, y_{1:m-1}), \quad f_2(\mathbf{x}_{1:m}, y_{1:m-1}), \quad f_3(\mathbf{x}_{1:m}, y_{1:m-1})\}. \quad (2.6c)$$

Most of the model sets of interest contain an infinite number of models, hence an exhaustive search over all those model sets is intractable. Instead, the search is usually done over a subset of these models. This subset is often constructed by parameterizing the set “*smoothly*” over some “*nice*” areas, which refers to that the model is differential with respect to the parameters, and that the parameters come from an open set. This could be done by restricting the parametric model $f(\mathbf{x}_m, \theta)$ to have a limited number of parameters d and that the gradient of the model with respect to the parameters should be well defined, i.e., the model should be differentiable with respect to the parameters [Ljung, 1999]. This gives the formal definition of *model structure* as:

Definition 2.3 (Model structure). A model structure $\mathcal{M}$ is a differential mapping from a connected open subset Θ of $\mathbb{R}$ to a model set $\mathcal{M}^*$, such that the gradient of the predictor function is smooth. That is

$$\mathcal{M} : \theta \in \Theta \rightarrow \mathcal{M}(\theta) = f(\mathbf{x}, \theta) \in \mathcal{M}^*, \quad (2.7)$$

where $\mathcal{M}(\theta)$ is one model in the model structure.

It is often interesting to know if the model set is able to include $\mathcal{S}$. That is, if there exists an θ^0 such that $\mathcal{M}(\theta^0) = \mathcal{S}$, which is denoted by $\mathcal{S} \in \mathcal{M}^*$. In practice, this assumption is hard to fulfill, but it gives useful insights into the estimated models [Ljung, 1999].

2.2.1 Linear models

When estimating a model that describes the relationship between the input $\mathbf{x}_m$ and s_m , it is a good practice to start by restricting the model set to simple models, then complexity can be added to the model. This is of particular importance when the model set consists of black-box models, i.e., universal approximators of functions. For example, models that are linear in the parameters are a good choice for simpler models to start with. The result from more complex models

can then later be compared against these simpler models, this to validate the use of the more complexed models. To initially use model linear in the parameters is good practice, since the analysis for them are more straightforward.

Given a parametric model that is linear in the parameters θ , i.e.,

$$f(\mathbf{x}_m, \theta) = \varphi_m^\top \theta, \quad \varphi_m^\top = g(\mathbf{x}_m), \quad (2.8)$$

where $\varphi_m^\top$ is denoting a basis function (sometimes also called regressors or pseudoregressors). The basis function is some transformation of the input $\mathbf{x}_m$ expressed by the function g , which might potentially be nonlinear. Then, it is possible to find the parameters that minimize the squared error loss function, i.e., the linear least squares solution (LLSS) as

$$\hat{\theta}_N = (\Phi_N \Phi_N^\top)^{-1} \Phi_N \mathbf{y}_N, \quad (2.9)$$

where

$$\mathbf{y}_N = \begin{pmatrix} y_1^\top & \cdots & y_N^\top \end{pmatrix}^\top, \quad (2.10)$$

is the observations of the output collected in a matrix with N number of rows, and

$$\Phi_N^\top = \begin{pmatrix} \varphi_1^\top & \cdots & \varphi_N^\top \end{pmatrix}^\top, \quad (2.11)$$

is an $N \times d$ matrix consisting of the basis functions for the N observations. In this thesis, three different choices of basis functions for models linear in the parameters are investigated: polynomials, Chebyshev polynomials, and Laplacian polynomials. Given a choice of basis functions, the model structure for these models is the number of basis functions used.

Polynomials

An initial approach to create basis functions is to use powers of the input, i.e.,

$$[\Phi_N(\mathbf{x}_m)]_{j,m} = \mathbf{x}_m^{j-1}. \quad (2.12)$$

where $j = 1, \dots, d$, and $[\Phi_N(\mathbf{x}_m)]_{j,m}$ denotes the j th element of the basis function for the m th sample. Given indefinitely many polynomial basis functions, it is possible to represent any given function arbitrarily well. One such example is Taylor series expansion of a function.

Chebyshev polynomials

To limit the number of basis function, one approach is to use basis functions that are orthogonal to each other [Gasquet and Witomski, 2013, Kreyszig, 1978]. Chebyshev polynomials are an example of such a basis function. They are defined by the recursion

$$[\Phi_N(\mathbf{x}_m)]_{1,m} = 1, \quad (2.13a)$$

$$[\Phi_N(\mathbf{x}_m)]_{2,m} = \mathbf{x}_m, \quad (2.13b)$$

$$[\Phi_N(\mathbf{x}_m)]_{j+1,m} = 2\mathbf{x}_m[\Phi_N(\mathbf{x}_m)]_{j,m} - [\Phi_N(\mathbf{x}_m)]_{j-1,m}. \quad (2.13c)$$

Denoting $\langle \cdot, \cdot \rangle$ as an inner product, Chebyshev polynomials are orthogonal on the interval $[-1, 1]$ using the inner product defined as

$$\langle f(\mathbf{x}_m), g(\mathbf{x}_m) \rangle = \int_{-1}^1 \frac{f(\mathbf{x}_m)g(\mathbf{x}_m)}{\sqrt{1 - \mathbf{x}_m^\top \mathbf{x}_m}} d\mathbf{x}_m. \quad (2.14)$$

Laplacian polynomials

Another choice of orthogonal basis functions is the Laplacian polynomials [Solin and Särkkä, 2019]. They are derived as the eigenfunctions of the Laplacian operator with Dirichlet boundary conditions

$$\begin{cases} [-\nabla \boldsymbol{\varphi}^\top(\mathbf{x}_m)]_j = \lambda_j [\boldsymbol{\varphi}^\top(\mathbf{x}_m)]_j, & \mathbf{x}_m \in \Omega \\ [\boldsymbol{\varphi}^\top(\mathbf{x}_m)]_j = 0, & \mathbf{x}_m \in \partial\Omega, \end{cases} \quad (2.15)$$

where some compact set $\Omega \subset \mathbb{R}^{d_x}$ with some sufficiently smooth boundary $\partial\Omega$ (but other boundary conditions could be used as well). The functions $[\boldsymbol{\varphi}(\mathbf{x}_m)]_j$ are eigenfunctions to the Laplacian operator with eigenvalues λ_j . Since $[\boldsymbol{\varphi}(\mathbf{x}_m)]_j$ are eigenfunctions, they are also orthogonal given an inner product defined as

$$\langle f(\mathbf{x}_m), g(\mathbf{x}_m) \rangle = \int_{\Omega} f(\mathbf{x}_m)g(\mathbf{x}_m)d\mathbf{x}_m. \quad (2.16)$$

The resulting eigenfunctions are defined as

$$[\Phi_N(\mathbf{x}_m)]_{j,m} = [\boldsymbol{\varphi}^\top(\mathbf{x}_m)]_j = \frac{1}{\sqrt{L}} \sin\left(\frac{\pi j(\mathbf{x}_m + L)}{2L}\right), \quad (2.17)$$

where L is a hyperparameter set by the user.

2.2.2 Neural networks

In many cases, models that are linear in the parameters are not flexible enough to represent the given data. This leads to that a model set consisting of more complex models needs to be used, [Goodfellow et al., 2016]. An example of model set with more complex models is the set containing NNs. One model in this set can be described by the recursions

$$\mathbf{h}^{(0)} = \mathbf{x}_m, \quad (2.18a)$$

$$\mathbf{a}^{(l+1)} = (\mathbf{h}^{(l)} \quad 1)^\top W^{(l)}, \quad l = 0, \dots, L, \quad (2.18b)$$

$$\mathbf{h}^{(l)} = \sigma(\mathbf{a}^{(l)}), \quad l = 1, \dots, L, \quad (2.18c)$$

$$s_m = \mathbf{a}^{(L)}. \quad (2.18d)$$

The number of layers in the network is denoted L , $W^{(l)}$ are the weights of the l th layer, $\mathbf{a}^{(l)}$ is the contribution from layer $(l - 1)$ to layer l and $\mathbf{h}^{(l)}$ is the activation

Figure 2.1: Schematic illustration of a fully-connected NN with d_x inputs, d_s outputs, and L -layers where every layer has d_l nodes $l = 1, \dots, L$. The bias term in hidden layers (nodes marked with the number 1), as well as the input in the NN are indicated with green nodes. The white nodes apply the nonlinear activation function $\sigma(\cdot)$ to the sum of all inputs. The blue nodes are the output nodes which sums all the inputs to them.

of the contribution from the $(l - 1)$ th layer at the l th layer, referred to as the *hidden node*. In Figure 2.1, a schematic illustration of a fully-connected NN with d_x inputs, d_s outputs, and d_l nodes in the l th hidden layer. This gives that the dimension of $W^{(l)}$ is $(d_{l-1} + 1) \times d_l$.

Furthermore $\sigma(\cdot)$ is a so-called activation function operating element-wise. Some of the most commonly used activation functions are the sigmoid function $\sigma(u) = 1/(1 + e^{-u})$, the hyperbolic tangent $\sigma(u) = \tanh(u)$, the rectified linear unit (ReLU) $\sigma(u) = \max\{u, 0\}$, the leaky ReLU $\sigma(u) = \max\{u, 0.1u\}$, and the exponential linear unit (ELU)

$$\sigma(u) = \begin{cases} u, & u \geq 0, \\ \alpha(e^u - 1), & u < 0, \end{cases} \quad (2.19)$$

where α is a hyperparameter that can be chosen by the user. In Figure 2.2 these activation functions and their derivatives are shown.

Collecting the biases and the weights of the NN into a parameter vector, i.e.

$$\theta = \left(\text{vec}(W^{(0)}) \quad \dots \quad \text{vec}(W^{(L)}) \right)^\top, \quad (2.20)$$

an NN can be written as a parametric model, $\hat{s}_m = f(\mathbf{x}_m, \theta)$. Hence, for an NN with d_x inputs, d_s outputs, and L hidden layers results in that the number of

(a) Activation functions

(b) Derivative of the activation functions

Figure 2.2: Some of the most commonly used activation functions together with their derivatives.

model parameters is

$$(d_x + 1)d_1 + \sum_{l=1}^{L-1} (d_l + 1)d_{l+1} + (d_L + d_L)d_s. \quad (2.21)$$

Given a nonlinear activation function, the model structure for NNs is given by the number of hidden layers L , and the number of nodes in each of these hidden layers d_l .

2.2.3 Model order selection

In system identification, an important aspect is to select the size of the model set under consideration, i.e., choosing the number of model parameters based on the trade-off between flexibility and overparameterization. Here, overparameterization refers to that a model where unnecessarily many parameters are used, i.e., a too flexible model. This is referred to as model order selection or the order selection problem, [Ljung, 1999]. For models that are linear in the parameters, such as polynomials, selecting the correct model order is equivalent to select the number of basis functions. For NNs, this would be the number of hidden layers as well as a number of nodes in those hidden layers.

A commonly used approach to select the model order is to study the cost function $V_N(\theta)$ for different choices of number of parameters d . Evaluated on training data, it is natural that the cost function decreases as the number of parameters increases since the model is more flexible, but this might not be desirable since models with a larger number of parameters may suffer from overparameterization and higher variance in its prediction. Instead, it is possible to study a version of the cost function, which have a penalty factor for using models with higher order. Two classical penalty functions are Akaike's information criterion (AIC) suggested by Akaike [1974] and Bayesian information criterion (BIC) suggested by Schwarz [1978]. In the case of Gaussian observation noise, AIC and BIC are given by

$$V_N(\hat{\theta}_N) \left(1 + \frac{2d}{N} \right), \quad (2.22a)$$

and

$$V_N(\hat{\theta}_N) \left(1 + \frac{d \log N}{N} \right), \quad (2.22b)$$

respectively.

2.3 Parameter optimization

When estimating a model from noisy data, the squared error between the measured data and the estimate is a commonly used measure of how accurate the

model is. The problem specified in (2.3) is to find the model with the least squares error, if the model is nonlinear in the parameters this is called the *nonlinear least squares problem*. To find the parameters of the nonlinear model that minimize the squared error, i.e., the solution the problem specified in (2.3) and (2.4), some kind of optimization algorithm needs to be used. A commonly used approach is to use gradient-descent based algorithms. In such approaches, the parameter θ is updated sequentially by taking a step in the direction of the negative gradient of the cost function. Using the notation $V'_N(\hat{\theta}_N) = \frac{\partial}{\partial \theta} V_N(\theta)|_{\theta=\hat{\theta}_N}$ for the gradient of the cost function with respect to the parameters in the model, the general form of an iteration performed by a gradient-descent based algorithm is

$$\hat{\theta}_N^{(j+1)} = \hat{\theta}_N^{(j)} - (R_N^{(j)})^{-1} V'_N(\hat{\theta}_N^{(j)}), \quad (2.23)$$

where superscript (j) denotes the estimate of the parameters at iteration j in the optimization algorithm [Goodfellow et al., 2016, Ljung, 1999]. Here, the matrix $R_N^{(j)}$ is used to decide how much one should move in every direction of the Gradient. The simplest choice for the step length is to use a scaled version of the identity matrix for $R_N^{(j)}$, this is sometimes referred to as the *vanilla gradient-descent*, but it is possible to make a much smarter choices of $R_N^{(j)}$, [Goodfellow et al., 2016].

2.3.1 Newton method

Now return to the estimation problem specified in (2.3) and (2.4). Assume that the estimated parameter $\hat{\theta}_N$ is close to the true one, i.e., $\theta^0 = \hat{\theta}_N + \Delta\theta$ where $\Delta\theta$ is small. Then a quadratic approximate model of the objective function around $\hat{\theta}_N$ can be obtained using Taylor series expansion as

$$V_N(\theta^0) \approx V_N(\hat{\theta}_N + \Delta\theta) = V_N(\hat{\theta}_N) + \Delta\theta^\top V'_N(\hat{\theta}_N) + \frac{1}{2} \Delta\theta^\top V''_N(\hat{\theta}_N) \Delta\theta, \quad (2.24)$$

where the notation

$$V''_N(\hat{\theta}_N) = \frac{\partial^2}{\partial \theta^2} V_N(\theta)|_{\theta=\hat{\theta}_N}, \quad (2.25)$$

is used. The Gradient and Hessian of the cost function can be calculated as

$$V'_N(\hat{\theta}_N) = \frac{1}{N} \sum_{n=1}^N \varepsilon'(\mathbf{x}_n, \hat{\theta}_N) \varepsilon(\mathbf{x}_n, \hat{\theta}_N) = \frac{1}{N} (\varepsilon'(\mathbf{x}_{1:N}, \hat{\theta}_N))^\top \varepsilon(\mathbf{x}_{1:N}, \hat{\theta}_N), \quad (2.26a)$$

$$\begin{aligned} V''_N(\hat{\theta}_N) &= \frac{1}{N} \sum_{n=1}^N \varepsilon'_\theta(\mathbf{x}_n, \hat{\theta}_N)^\top \varepsilon'_\theta(\mathbf{x}_n, \hat{\theta}_N) + \frac{1}{N} \sum_{n=1}^N \varepsilon(\mathbf{x}_n, \hat{\theta}_N) \varepsilon''_\theta(\mathbf{x}_n, \hat{\theta}_N) \\ &= \frac{1}{N} (\varepsilon'_\theta(\mathbf{x}_{1:N}, \hat{\theta}_N))^\top \varepsilon'_\theta(\mathbf{x}_{1:N}, \hat{\theta}_N) + \frac{1}{N} \sum_{n=1}^N \varepsilon(\mathbf{x}_n, \hat{\theta}_N) \varepsilon''_\theta(\mathbf{x}_n, \hat{\theta}_N), \end{aligned} \quad (2.26b)$$

using the notation

$$\varepsilon'_{\theta}(\mathbf{x}_m, \hat{\theta}_N) = \frac{\partial}{\partial \theta} \varepsilon(\mathbf{x}_m, \theta) \Big|_{\theta=\hat{\theta}_N}, \quad (2.27a)$$

and

$$\varepsilon''_{\theta}(\mathbf{x}_m, \hat{\theta}_N) = \frac{\partial^2}{\partial \theta^2} \varepsilon(\mathbf{x}_m, \theta) \Big|_{\theta=\hat{\theta}_N}, \quad (2.27b)$$

as well as $\varepsilon(\mathbf{x}_{1:N}, \hat{\theta}_N)$ and $\varepsilon'_{\theta}(\mathbf{x}_{1:N}, \hat{\theta}_N)$ which denote the vector with N values of the prediction error and the matrix with all the derivatives of the prediction error, respectively.

Returning to the quadratic model, the error in the estimated parameters $\Delta\theta$ calculated as the solution to the following system of equations

$$V_N''(\hat{\theta}_N)\Delta\theta = -V_N'(\hat{\theta}_N). \quad (2.28)$$

i.e., when the derivative of (2.24) with respect to $\Delta\theta$ is equal to zero. In particular, this describes approximately the optimal estimate $\hat{\theta}_N$ as a function of the observations around the optimal solution θ^0 , valid for small values of $\Delta\theta$.

Given that $V_N''(\hat{\theta}_N)$ is invertible, i.e., non-singular, this also gives us a good idea of how to find a better approximation of the true parameters, because $\hat{\theta}_N$ can be updated as

$$\hat{\theta}_N^{(j+1)} = \hat{\theta}_N^{(j)} - \left(V_N''(\hat{\theta}_N^{(j)}) \right)^{-1} V_N'(\hat{\theta}_N^{(j)}), \quad (2.29)$$

which is known as Newton's method [Ljung, 1999, Nocedal and Wright, 2006].

2.3.2 Gauss-Newton method

For many problems, it can be very costly to calculate the full Hessian $V_N''(\hat{\theta}_N)$. If the estimation of $\hat{\theta}_N$ is near optimum, the cost function is close to linear. Hence the second derivative of the cost function is small. This results in that the Hessian can approximate as

$$V_N''(\hat{\theta}_N) \approx H_N(\hat{\theta}_N), \quad (2.30)$$

where

$$H_N(\theta) \triangleq \frac{1}{N} \sum_{n=1}^N \varepsilon'_{\theta}(\mathbf{x}_n, \hat{\theta}_N)^{\top} \varepsilon'_{\theta}(\mathbf{x}_n, \hat{\theta}_N). \quad (2.31)$$

Hence, this approximation of the Hessian is constructed by only consider the first term in (2.26b). This would result in the Gauss-Newton method seen in Algorithm 2.1, see e.g., [Nocedal and Wright, 2006]. The approximation of the Hessian is especially good if the prediction errors are independent, which would be the case if the model set includes the true system, see [Ljung, 1999] and Chapter 4.

Algorithm 2.1: Gauss-Newton method

Input: Sequence of stepsizes $\mu_N^{(j)}$
 $V_N(\theta)$: Cost function

- 1: **Initialization:** $\hat{\theta}_N^{(0)}$: Initialize the parameter vector of dimension n_θ
- 2: **while** $\hat{\theta}_N^{(j)}$ not converged **do**
- 3: $\mathbf{g}_N^{(j)} = V'_N(\hat{\theta}_N^{(j)})$
 $R_N^{(j)} = H_N(\hat{\theta}_N^{(j)})$, where $H_N(\theta)$ is given in (2.31).
 $\hat{\theta}_N^{(j+1)} = \hat{\theta}_N^{(j)} - \mu_N^{(j)} [R_N^{(j)}]^{-1} \mathbf{g}_N^{(j)}$
- 4: **end while**
- 5: **return** Resulting parameters $\hat{\theta}_N^{(j+1)}$

If (2.31) is singular or close to singular, a popular adaptation of the Gauss-Newton method is the Levenberg-Marquardt (LM) algorithm. Then, the approximation of the Hessian is replaced by

$$R_N^{(j)}(\lambda) = \frac{1}{N} \sum_{n=1}^N \varepsilon'_\theta(\mathbf{x}_n, \hat{\theta}_N)^{\top} \varepsilon'_\theta(\mathbf{x}_n, \hat{\theta}_N) + \lambda I, \quad (2.32)$$

where here λ is a positive scalar to ensure that the estimate of the Hessian is a positive definite matrix. For overparameterized models, this approach is especially useful since the overparameterization could result in that (2.31) loses rank, [Ljung, 1999, Nocedal and Wright, 2006].

2.4 Parameter optimization for neural networks

For training NNs, gradient-descent based algorithms are commonly used as optimization algorithms. In practice when training NN, there are often a lot of training data available and it is not computationally efficient to use all the data in every iteration in the optimization algorithm. Instead, the data is randomly split into different batches where one batch is used in each iteration. When all batches have been used, and all training data has been processed by the optimization algorithm once, the process repeats, and new batches are created. To iterate over all the batches is called an *epoch*. To divide the data into different batches, where one batch is used at a time in the optimization algorithm, is often referred to as *stochastic gradient-descent*, [Bishop, 2006, Goodfellow et al., 2016].

The convergence of stochastic gradient-descent is often slow, but to calculate the Hessian, as done in the Gauss-Newton method, is often not tractable due to NNs being highly flexible models with many parameters. Also, since in each iteration only a subset of the training data is used, it is useful to include information from the earlier iterations. This can be done by using the moving average of the gradient (and Hessian) when calculating the step used to adjust the parameters in the NN, [Goodfellow et al., 2016].

Algorithm 2.2: RMS-prop**Input:** α Stepsize $\rho \in [0, 1)$: Decay rates for the moving average $V_N(\theta)$: Cost function

- 1: **Initialization:** $\hat{\theta}_N^{(0)}$: Initialize the parameter vector of dimension n_θ

$$\tilde{H}_N^{(0)} = 0$$

- 2: **while** $\hat{\theta}_N^{(j)}$ not converged **do**

- 3: $\mathbf{g}_N^{(j)} = V'_N(\hat{\theta}_N^{(j)})$
 $[H_N^{(j)}]_{k,k} = [\mathbf{g}_N^{(j)}]_k^2 \quad \forall k = 1, \dots, n_\theta$
 $\tilde{H}_N^{(j)} = \rho \tilde{H}_N^{(j+1)} + (1 - \rho) H_N^{(j)}$
 $R_N^{(j)} = (\tilde{H}_N^{(j)})^{1/2} + \epsilon I$
 $\hat{\theta}_N^{(j+1)} = \hat{\theta}_N^{(j)} - \alpha [R_N^{(j)}]^{-1} \mathbf{g}_N^{(j)}$

- 4: **end while**

- 5: **return** Resulting parameters $\hat{\theta}_N^{(j+1)}$

Algorithm 2.3: ADAM optimizer**Input:** α Stepsize $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moments estimates $V_N(\theta)$: Cost function

- 1: **Initialization:** $\hat{\theta}_N^{(0)}$: Initialize the parameter vector of dimension n_θ

$$\bar{\mathbf{g}}_N^{(0)} = \mathbf{0}$$

$$\tilde{H}_N^{(0)} = 0$$

- 2: **while** $\hat{\theta}_N^{(j)}$ not converged **do**

- 3: $\mathbf{g}_N^{(j)} = V'_N(\hat{\theta}_N^{(j)})$
 $\bar{\mathbf{g}}_N^{(j)} = \beta_1 \bar{\mathbf{g}}_N^{(j-1)} + (1 - \beta_1) \mathbf{g}_N^{(j)}$
 $\hat{\mathbf{g}}_N^{(j)} = \bar{\mathbf{g}}_N^{(j)} / (1 - \beta_1^j)$
 $[H_N^{(j)}]_{k,k} = [\hat{\mathbf{g}}_N^{(j)}]_k^2 \quad \forall k = 1, \dots, n_\theta$
 $\tilde{H}_N^{(j)} = \beta_2 \tilde{H}_N^{(j+1)} + (1 - \beta_2) H_N^{(j)}$
 $\hat{H}_N^{(j)} = \tilde{H}_N^{(j)} / (1 - \beta_2^j)$
 $R_N^{(j)} = (\hat{H}_N^{(j)})^{1/2} + \epsilon I$
 $\hat{\theta}_N^{(j+1)} = \hat{\theta}_N^{(j)} - \alpha [R_N^{(j)}]^{-1} \hat{\mathbf{g}}_N^{(j)}$

- 4: **end while**

- 5: **return** Resulting parameters $\hat{\theta}_N^{(j+1)}$

2.4.1 RMS-prop and ADAM optimizer

Two of the most commonly used optimization algorithms for training NNs are RMS-prop suggested by Tieleman and Hinton [2012] and the ADAM optimizer suggested by Kingma and Ba [2015]. Both of the algorithms are extensions of the standard stochastic gradient-descent which are utilizing a moving average of the gradient (and Hessian) when updating the parameters. These algorithms² are outlined in Algorithm 2.2 and Algorithm 2.3, respectively. In the algorithms, $A^{1/2}$ is the square root of the matrix defined using the singular value decomposition (SVD). Compared to the Gauss-Newton method presented in Algorithm 2.1, where the Hessian is approximated by (2.31), the RMS-prop and ADAM optimizer approximate the Hessian with diagonal elements in (2.31). Then, as step size used by the optimization algorithm in the update of the parameters, the square root of a moving average of the Hessian is used.

The main difference between RMS-prop and ADAM optimizer is that ADAM optimizer also uses a moving average of the gradient of the cost function with respect to the parameters. This can be motivated similarly to the use of moving average for the Hessian.

In both algorithms, there are a couple of hyperparameters connected to the decay in the accumulated moving average of the moments, i.e., ρ for RMS-prop and β_1 and β_2 for ADAM. In general, the default values $\rho = 0.9$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ are used when training NNs [Goodfellow et al., 2016, Kingma and Ba, 2015, Tieleman and Hinton, 2012]. Intuitively, this means that current estimates of the Hessian are used to a larger extent compared to the moving average estimate of the Hessian. In some sense, this is similar to have a high leakage factor in a leaky recursive least mean square algorithm [Gustafsson et al., 2011]. The value of ϵ is often set to 10^{-8} to regularize the approximation of the Hessian and make sure it is not singular.

2.4.2 Initialization of neural networks

How to choose the initial guess of the parameters $\hat{\theta}_N^{(0)}$ is a challenging problem and crucial for the performance of the optimization algorithms. The standard way used for NN is the so-called Xavier initialization suggested by Glorot and Bengio [2010], which states that for a fully-connected NN, the weights at the j th layer should be initialized by sampling from

$$W^{(j)} \sim \mathcal{N}\left(0, \sqrt{\frac{2}{d_j + d_{j+1}}}\right). \quad (2.33)$$

²These algorithms are presented with a slight different notation compared to the original formulations, this is in order to make comparison with Algorithm 2.1 easier.

If ReLU is used as an activation function, He et al. [2015] has shown that it is better to initialize the weights as

$$W^{(j)} \sim \mathcal{N}\left(0, \sqrt{\frac{2}{d_j}}\right). \quad (2.34)$$

2.4.3 Backpropagation

To use any of these optimization algorithms in Section 2.3.2 and Section 2.4.1, one needs to calculate the derivate of the cost function with respect to to the network parameters. Using the recurrent structure of the NN, this process can be automated using automatic differentiation in reverse mode. For NNs, this process is referred to as *backpropagation*, [Rumelhart et al., 1986]. Consider an NN with $\sigma(x)$ as activation function and $\sigma'(x) = \frac{\partial \sigma(x)}{\partial x}$ as the derivative of the activation function. Then, by first evaluating the network at every node in every layer

$$\mathbf{a}^{(k)} = \mathbf{h}^{(k-1)}[W^{(k-1)}]_{1:d_{k-1},1:d_k} + [W^{(k-1)}]_{0,1:d_k}, \quad (2.35a)$$

$$\mathbf{h}^{(k)} = \sigma(\mathbf{a}^{(k)}), \quad (2.35b)$$

where $W_{i_1:i_2,j_1:j_2}^{(l)}$ denotes the i_1 to i_2 row and j_1 to j_2 column in the matrix of weights connecting the $(l-1)$ th layer to the l th layer, the derivative of the cost function with respect to the parameters can be computed as

$$\delta^{(k)} = \frac{\partial V_N(\boldsymbol{\theta})}{\partial \mathbf{a}^{(k)}} = \frac{\partial V_N(\boldsymbol{\theta})}{\partial \mathbf{h}^{(k)}} \odot \sigma'(\mathbf{a}^{(k)}), \quad (2.35c)$$

$$\delta^{(k-1)} = \sigma'(\mathbf{a}^{(k-1)})\delta^{(k)} \odot [W^{(k-1)}]_{1:d_{k-1},1:d_k}, \quad (2.35d)$$

$$\frac{\partial V_N(\boldsymbol{\theta})}{\partial [W^{(k)}]_{1:d_{k-1},1:d_k}} = \frac{\partial V_N(\boldsymbol{\theta})}{\partial \mathbf{a}^{(k)}} \frac{\partial \mathbf{a}^{(k)}}{\partial [W^{(k-1)}]_{1:d_{k-1},1:d_k}} = \delta^{(k)} \mathbf{h}^{(k-1)}, \quad (2.35e)$$

$$\frac{\partial V_N(\boldsymbol{\theta})}{\partial [W^{(k-1)}]_{0,1:d_{k-1}}} = \frac{\partial V_N(\boldsymbol{\theta})}{\partial \mathbf{a}^{(k)}} \frac{\partial \mathbf{a}^{(k)}}{\partial [W^{(k-1)}]_{0,1:d_{k-1}}} = \sum_{i=1} [\delta^{(k)}]_{i,\cdot}. \quad (2.35f)$$

Here $\odot$ denotes element-wise multiplication. In a regression setting where a least squares criterion is used as cost function, the first derivative of the cost function with respect to the activation $\mathbf{a}^{(L)}$, i.e., $\delta^{(L)}$, is given by

$$\delta^{(L)} = -2 \sum_{n=1}^N (y_n - f(\mathbf{x}_n, \boldsymbol{\theta})). \quad (2.36)$$

2.4.4 Regularization

When optimizing flexible models such as NNs, one might end up overfitting to the training data. To prevent this one can use different regularization techniques. One of the most commonly used regularization is L^2 -regularization, where the

square of the parameters is penalized, [Goodfellow et al., 2016, Ljung, 1999, Nocedal and Wright, 2006]. That is, the cost function in (2.4) is modified such that

$$V_N(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N \|y_n - f(\mathbf{x}_n, \boldsymbol{\theta})\|^2 + \lambda \|\boldsymbol{\theta}\|^2, \quad (2.37)$$

where λ is a scalar value.

For NNs, there has been a substantial work developing different regularization techniques. Two of the most popular techniques are *dropout* and *batchnorm*, [Hinton et al., 2012b, Ioffe and Szegedy, 2015, Srivastava et al., 2014].

Dropout

Dropout was popularized by Hinton et al. [2012b] and Srivastava et al. [2014] and has become a standard regularization technique to prevent overfitting in NNs. The idea is that combining multiple models would lead to better generalization performance of NNs. Training NN with dropout is equivalent to training an ensemble of thinner NNs and then combining these to a single “unthinned” NN at prediction. An NN with n nodes can be seen as an ensemble of 2^n thinned NNs, where the thinned NNs are created by dropping nodes in the network with a given dropout probability. During the training of the NN, one of these thinned NNs are sampled in every iteration, i.e., not all weights in the NN are updated in every step of the training of the network. Hence, training an NN with dropout is equivalent to training an ensemble of 2^n thin NNs with extensive weight sharing, where every thin NN gets trained very rarely, or not at all.

In the original formulation of dropout by [Hinton et al., 2012b, Srivastava et al., 2014], the combination of all the thin NNs are done by scaling the weights with the dropout probability, to create a single NN for prediction. That is, given the parameters $\hat{\boldsymbol{\theta}}_N$ of the “unthinned” NN, the parameters of the thin NNs are given by $\mathbf{z}_j \odot \hat{\boldsymbol{\theta}}_N$, where $\odot$ denotes the element-wise product. The variable $\mathbf{z}_j \sim \text{Bernoulli}(\mathbf{p})$ indicates if a parameter is dropped or not in the $j = 1, \dots, J$ thinned NN, and $\mathbf{p}$ is a vector with all the dropout probabilities for every parameter $\hat{\boldsymbol{\theta}}_N$. Then the prediction of the “unthinned” NN is the average over the J realizations with³

$$E[\hat{s}_m] = \frac{1}{J} \sum_{j=1}^J f(\mathbf{x}_m, \mathbf{z}_j \odot \hat{\boldsymbol{\theta}}_N) \approx f(\mathbf{x}_m, \mathbf{p} \odot \hat{\boldsymbol{\theta}}_N). \quad (2.38)$$

Batchnorm

In recent years, Batchnorm introduced by Ioffe and Szegedy [2015] has become one of the most popular regularization techniques. The idea of Batchnorm comes from the fact that NNs work best for normalized input and output, since the commonly used activation functions usually are most expressive in an area around

³To simplify the notation, assume that the observations noise e_m has zero mean.

the origin. This results in that optimization of the NN is more likely to avoid vanishing and exploding gradients around origin.

Instead of only normalizing the input and output to the NN, when Batchnorm is used to train an NN the input and output are also normalized between the different layers. In this sense, the network learns a bias and a variance of the output from every layer for every batch in the training data.

2.5 Hessian and parameter covariance

There is a close connection between the Hessian of the cost function and the Fisher information matrix, and hence to the parameter covariance which is the inverse of the Fisher information matrix [Gustafsson, 2018, Kay, 1993, Liero and Zwanzig, 2011, Ljung, 1999]. The Fisher information matrix for the parameters θ is given by the second derivative of the log-likelihood with respect to the parameters. In the case of Gaussian noise this would be the Hessian of the least squares cost function.

This connection can also be seen in the optimization algorithms presented Section 2.4. For example, $\tilde{H}_N^{(j)}$ in RMS-prop and ADAM-optimizer is referred to as an estimate of the variance of the gradient of the cost function, i.e., the Hessian, [Kingma and Ba, 2015, Tieleman and Hinton, 2012]. The connection is even more clear if one would consider a model that is linear in the parameters. Then the second term in (2.26b) is zero, i.e.,

$$V_N''(\hat{\theta}_N) = \frac{1}{N}(\varepsilon'_\theta(\mathbf{x}_{1:N}, \hat{\theta}_N))^\top \varepsilon'_\theta(\mathbf{x}_{1:N}, \hat{\theta}_N), \quad (2.39)$$

which is a constant matrix. That is, since the observation noise does not depend on the parameters, $\varepsilon'_\theta(\mathbf{x}_{1:N}, \hat{\theta}_N)$ is a constant matrix. Assuming that the prediction error has zero mean and variance λ_0 as well as the Hessian has full rank, the covariance of the estimated parameters is given by

$$\text{Cov}(\hat{\theta}_N - \theta^0) = \text{Cov}(\hat{\theta}_N) = [\text{solution to (2.28)}] \quad (2.40a)$$

$$= \text{Cov}\left((V_N''(\hat{\theta}_N))^{-1} V_N'(\hat{\theta}_N)\right) \quad (2.40b)$$

$$= \text{Cov}\left((V_N''(\hat{\theta}_N))^{-1} \frac{1}{N}(\varepsilon'_\theta(\mathbf{x}_{1:N}, \hat{\theta}_N))^\top \varepsilon(\mathbf{x}, \hat{\theta}_N)\right) \quad (2.40c)$$

$$= \lambda_0(V_N''(\hat{\theta}_N))^{-1} \frac{1}{N}(\varepsilon'_\theta(\mathbf{x}_{1:N}, \hat{\theta}_N))^\top \varepsilon'_\theta(\mathbf{x}_{1:N}, \hat{\theta}_N)(V_N''(\hat{\theta}_N))^{-1} \quad (2.40d)$$

$$= \lambda_0(V_N''(\hat{\theta}_N))^{-1} V_N''(\hat{\theta}_N)(V_N''(\hat{\theta}_N))^{-1} \quad (2.40e)$$

$$= \lambda_0(V_N''(\hat{\theta}_N))^{-1}. \quad (2.40f)$$

Hence, the covariance of the parameter for a model which is linear in the parameters is proportional to the inverse of the Hessian of the cost function.

For nonlinear models, the cost function is close to quadratic near the optimum and hence the second derivative of the cost function is small, i.e., a linear approximation of the cost function is viable in that region. This also motivates that the

first term of the Hessian in (2.26b) is more important than the second term, see [Nocedal and Wright, 2006]. The same arguments are used when finding the step direction in the Gauss-Newton method, see Algorithm 2.1. In Chapter 4, this is thoroughly examined for a nonlinear model, highlighting the result that if the true model is in the model set, the second term in the Hessian is small and can be neglected. In Chapter 5 the analysis is extended to the case when the Hessian is ill-conditioned or even singular.

2.6 Uniqueness

Due to the ambiguities in the model structure, the choice of parameters for NN is not unique. This ambiguity comes from symmetries in the NN model structure and the activation function, as well as possible overparameterizations in terms of nodes needed to describe the true input-output relationship; see Hwang and Ding [1997] and D'Amour et al. [2020] for details. Hence, the parameter estimate $\hat{\theta}_N$ is non-unique and will depend on factors such as the training data realization, as well as the initial condition and choice of optimization algorithm and its implementation. Let the set of parameter vectors that minimize the cost function (2.4) be defined as

$$\mathcal{D}_{\hat{\theta}} \triangleq \{\hat{\theta}_N \in \Theta : \hat{\theta}_N = \arg \min_{\theta} V_N(\theta)\}. \quad (2.41)$$

Depending on if the true model belongs to the chosen model set $\mathcal{M}^*$ or not, several different sources contributing to the prediction uncertainty will be considered. If $f^* = \mathcal{S} \in \mathcal{M}^*$ there exists a non-empty parameter set

$$\mathcal{D}_{\theta^0} \triangleq \{\theta^0 \in \Theta : f(\mathbf{x}_m, \theta^0) = f^*(\mathbf{x}_m) \quad \forall \mathbf{x}_m\}, \quad (2.42)$$

of parameter vectors where $\theta^{i0} \in \mathcal{D}_{\theta^0}$ is the i th parameter vector such that the NN model describes the input-output relationship perfectly. Otherwise, if $\mathcal{S} \notin \mathcal{M}$, from Ljung [1999] it is known that the parameter estimate converges to a vector θ^{i*} that gives the best fit of the training data in the least squares sense. That is, $\theta^{i*} \in \mathcal{D}_{\theta^*}$, where

$$\mathcal{D}_{\theta^*} \triangleq \{\theta^* \in \Theta : \theta^* = \lim_{N \rightarrow \infty} \arg \min_{\theta} V_N(\theta)\}. \quad (2.43)$$

If $\mathcal{S} \in \mathcal{M}^*$ then $\mathcal{D}_{\theta^*} \equiv \mathcal{D}_{\theta^0}$ and the parameter estimate converges to one vector in this set $\mathcal{D}_{\theta^0}$.

2.6.1 Canonical representation of a model

A canonical model is an irreducible, or minimal, representation of the model if there is no model with fewer parameters that can represent the given data as well as the canonical model. A related concept is redundant parameters, i.e., parameters that are not adding flexibility to the model. If a model has redundant parameters, it is also reducible.

To handle the fact that the choice of parameters in the NN in general is non-unique and hence not globally identifiable, see Definition 2.5, a canonical representation of a parametric model $f_c : \mathbb{R} \times \mathbb{R}^{d_c} \rightarrow \mathbb{R}$ with a corresponding canonical, i.e., unique and irreducible, parameter vector $\theta^c \in \mathbb{R}^{d_c}$ is introduced. Here, the number of parameters in the canonical representation d_c is less or equal to the number of parameters in the non-canonical representation of the model d , $d_c \leq d$. For the canonical representation of a parametric model, it holds that given any θ there exists a unique θ^c such that

$$f(\mathbf{x}, \theta) = f_c(\mathbf{x}, \theta^c), \quad \forall \mathbf{x}. \quad (2.44)$$

Hence, $f_c \in \mathcal{M}^*$ can represent any input-output relation that $f \in \mathcal{M}^*$ can, but the parameterization is assumed unique and potentially of lower dimension. Furthermore, assume that there exist k differentiable mappings T_i , $i = 1, \dots, k$, relating any parameter vector θ^i in the original representation of the model and the corresponding one θ^c in the canonical representation of the model such that $\theta^c = T_i(\theta^i)$.

There could be the case that multiple choices of θ^i resulting in irreducible models, i.e., models with the same number of parameters that give the same prediction. In that case, any of those models could be considered the canonical representation of the model. Therefore, it is a choice of which of these irreducible models should be the canonical representation of the model, and which parameters θ^i should be considered to be θ^c . After a choice of which realization of the model that should be considered canonical has been made, one could consider θ^c unique and relate them to the other irreducible parameters by a mapping T_i .

2.6.2 Symmetries in neural networks

In Hwang and Ding [1997] it is shown that for a two-layer NN with sigmoid activation functions there exist only two kinds of transformations on θ^c such that $\theta^i \in \mathbb{R}^{d_c}$, namely:

- (i) Interchange of nodes, i.e., interchange $\mathbf{h}_i^{(l)}$ and $\mathbf{h}_j^{(l)}$ with each other.
- (ii) Change of sign of the parameters due to symmetries in the activation function, i.e., $W_{j,k}^{i(l)} = -W_{j,k}^{c(l)}$ and $W_{\cdot,j}^{i(l-1)} = -W_{\cdot,j}^{c(l-1)}$. For sigmoid activation functions, which are symmetric around 0.5, the change of sign will also result in a move of information to the bias term. This added information could be described as $W_{0,k}^{i(l)} = W_{0,k}^{c(l)} + W_{j,k}^{c(l)}$. For activation functions symmetric around zero, e.g., tanh, it is not necessary to move any information to the bias term. See Figure 2.2a for illustrations of the activation functions.

These two transformations for a two-layer NN with a sigmoid as activation function will generate a family with $2^{d_1} d_1!$ elements. See Hwang and Ding [1997], Sussmann [1992] for more details of how such a mapping can be constructed in an irreducible two-layer NN with sigmoid activation functions.

Remark 2.1. One can notice that these transformations are linear, i.e., $\theta^c = T_i \theta^i$. Hence the parameters for all irreducible NNs, i.e., NNs that have the number of parameters as the canonical representation, can be written as a linear combination of the parameters of the canonical representation.

Example 2.2

Consider an example where a signal is generated by

$$f^*(\mathbf{x}_m) = D \sin \left(C \arctan \left((1 - E)s + E/B \arctan(Bs) \right) \right), \quad (2.45)$$

which is a model that describes how the NTF depends on the wheel slip, for further description of this model see Section 6.1. This signal is aimed to be estimated by a black-box model, i.e., a universal approximator of the data.

Assume a NN with two layers as the black-box model, i.e.,

$$f(\mathbf{x}_m, \theta) = \theta_5 \sigma(\theta_1 \mathbf{x}_m + \theta_3) + \theta_6 \sigma(\theta_2 \mathbf{x}_m + \theta_4) + \theta_7. \quad (2.46)$$

A schematic illustration of such model is seen in Figure 2.3. Denote the canonical representation of the model by

$$f_c(\mathbf{x}_m, \theta^c) = 0.6 \sigma(-6.8 \mathbf{x}_m + 0.0064) - 1.4 \sigma(-40 \mathbf{x}_m + 0.0039) + 0.42. \quad (2.47)$$

The parameters for the canonical representation of the model is one of parameter vectors, that are minimizing the squared error between observation generated (2.45) and a two layer NN with two nodes in the hidden layer. The parameters are found by training the NN using the ADAM optimizer, [Kingma and Ba, 2015], with the standard settings for the hyperparameters, where the parameters are initialized using the Xavier initialization, [Glorot and Bengio, 2010].

The process of training the NNs to find parameters that minimize the squared error is repeated 200 times. For the 200 realization, different initial values of the parameters is used, and the separation of training data into batches are done differently. The batches are as subset of training data, where one batch at a time is used in the optimization algorithm. These models are referred to as the non-canonical representation of the model.⁴

In Figure 2.4a the two linear parameters connecting the hidden layer with the output layer, i.e., $\theta_5 = W_{11}^{(1)}$ and $\theta_6 = W_{12}^{(1)}$, are plotted for the different realizations. One can see that the parameters converge to some finite sets of different values.

Due to the high dimensionality of the parameter vector, it is hard to illustrate how the cost function depends on those parameters in the NN, as well as how the parameter is changing in the different iterations in the solver. In Figure 2.5 the cost function for the canonical representation of the NN is shown, where the parameters connecting the hidden layer to the output layer are changed, while

⁴Notice that since the canonical representation of the model and non-canonical representation of the model have the same size, there are no redundant parameters in the non-canonical model, i.e., all are irreducible.

Figure 2.3: Schematic illustration of a fully-connected NN with one inputs, one outputs, and one hidden layer with two nodes in the hidden layer. The same color-coding as Figure 2.1 is used.

all other parameters are fixed. The cost function is steeper in one direction compared to the other one. A projection of the eigenvector corresponding to the largest eigenvalue of the Hessian matrix can be seen as the steeper direction in the cost function, while the projection of the eigenvector that corresponds to the smallest eigenvalue of the Hessian matrix can be seen as a flat direction in the cost function. As demonstrated in Section 2.5, there is a connection between the Hessian of the cost function and the parameter covariance. The smaller eigenvalues of the Hessian would represent larger parameter covariance and a more uncertain prediction, and vice versa. Using this interpretation of the Hessian, one can see that directions with larger curvature in the cost function correspond to directions where the parameter estimate is more certain and directions that have a smaller curvature correspond to directions where the parameter estimate is more uncertain. One can also see that the solver moves slower in the direction corresponding to the uncertain combination of the parameters compared to the certain one.

From Figure 2.4a one can see that the parameters end up in eight different clusters. This coincides with the predicted number of the clusters for a two-layer NN with $d_1 = 2$ presented earlier in this section. But the path defined by the intermediate iteration from the initialization of the parameters to the optimal parameters looks far from direct. This might be a consequence of that the parameter space which has seven dimensions is projected to a two-dimensional plane. This behavior can also be seen in Figure 2.4b where the cluster of the optimal parameters and their initial values are color-coded. There one can see that even though a parameter is initialized close to a specific cluster, it can converge to another cluster. One such example is the initialization near the red cluster converges to the blue cluster.

For all the clusters the prediction is equally good, so a natural next question is if the uncertainty in the parameters for the different clusters is different? This

(a) Parameters for different realizations. For a couple of realizations, the path of intermediate iterations from the initial parameter values to the optimum are visualised.

(b) Clusters of the parameters for different realizations. The final value and the initialization of the parameters are color coded such that realizations that converge to the same cluster have the same color.

Figure 2.4: Illustration of how symmetry in the NN can lead to different optimal parameters for different realizations. The parameters are clustered in eight clusters. Here the parameters connecting the hidden layer to the output layer are visualized, i.e., θ_5 and θ_6 , where the numbering of the parameters can be seen in Figure 2.3 and (2.46).

Figure 2.5: Cost function as a function of the parameters connecting the output layer and the last hidden layer, i.e., θ_5 and θ_6 , where the numbering of the parameters can be seen in Figure 2.3 and (2.46). The projection of the eigenvectors that corresponds to the smallest and largest eigenvalue is also included in the figure.

will be further examined in Chapter 5. Next, it is shown how a mapping between the different clusters can be constructed.

Example 2.3

Returning to the setup in Example 2.2 one can compare the values of the canonical realization of the parameters (in the red cluster) given in (2.47) and a realization from the light blue cluster given by

$$f(\mathbf{x}_m, \theta^i) = -1.4\sigma(-40\mathbf{x}_m + 0.0061) + 0.60\sigma(-6.8\mathbf{x}_m + 0.0036) + 0.42. \quad (2.48)$$

It is possible to see that the parameters are related by the transformation given by

$$T_i = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad (2.49)$$

i.e., interchanging the nodes in the same layer corresponding to similar operations (bias or linear combination) with each other, which is the first sort of transformations, described earlier in this section.

For the analysis in this thesis, the restriction to irreducible NNs is relaxed to NNs that are equally expressive, i.e., the canonical and non-canonical model can express equally complex systems. The main difference is that the transformation T_i also can be applied to models with redundant parameters, i.e., non-canonical models having parameters that are linear combinations of parameters in the canonical model. This will be handled in more detail in Chapter 5. Noteworthy is that the somewhat abstract function T_i is only needed for the forthcoming analysis in Chapter 4 and not, as shown in Chapter 5, for the application of the discussed methods.

2.7 Regularity conditions and identifiability

To pursue the analysis of the prediction error variance, some regularity conditions regarding both the observed data generated by the true system, as well as the smoothness of the models in the model structure and the loss function need to be imposed.

2.7.1 Identifiability and informativity

Given a training data set, the user chooses a model set from which a model should be estimated. However, the estimator might not converge, or the model might not be reasonable. This can be a consequence both of the choice of model structure for which the search is conducted, but also a consequence of that the data set is not informative enough. Using the definition for informativity used in Ljung [1999] and Gevers et al. [2009], but modified for static system.

Definition 2.4 (Informativity). A data set, $\mathbf{x}_{1:N}$ is informative enough with respect to a model set $\mathcal{M}^*$ if for any two models $f(\mathbf{x}, \theta_1)$ and $f(\mathbf{x}, \theta_2)$ in the model set

$$f(\mathbf{x}, \theta_1) - f(\mathbf{x}, \theta_2) = 0, \quad \forall \mathbf{x} \implies \theta_1 = \theta_2. \quad (2.50)$$

Example 2.4

Consider the model set of polynomials of degree n , which is an example of a nonlinear static model. For the data to be informative enough, at least $n + 1$ unique data points are needed.

For further discussions of informativity see [Gevers et al., 2009]. Throughout this thesis it will be assumed that generated training data is informative enough such that all models in the true system have been persistently excited.

Considering the chosen model structure, one has to make sure that the model structure is globally identifiable, i.e., two choices of parameters θ will not result in the same output from the model. First, let us define the concept of (local) identifiability of a model using the uniqueness-oriented definition suggested in [Ljung, 1999].

Definition 2.5 (Identifiability). A parametric model structure is locally identifiable at value θ^* if $\exists \delta > 0$ such that for all $\|\theta - \theta^*\| < \delta$

$$f(\cdot, \theta^*) = f(\cdot, \theta) \implies \theta = \theta^*. \quad (2.51)$$

For a model structure to be globally identifiable at θ^* , the same must hold as $\delta \rightarrow \infty$, i.e., it has to be locally identifiable at all θ^* .

Taking the smoothness of the models in the model sets into consideration, the following assumption is needed.

Assumption A1. Let Θ be compact, and assume that the model is three times continuously differentiable. Assume as well that both the model and its derivatives, component-wise, are bounded. For the loss function, it will be assumed that it is three times differentiable with respect to the parameters and the prediction error as well as the norm of derivatives and cross-derivatives are bounded, as described in [Ljung and Caines, 1980].

These assumptions are needed to show that the discussed estimators are consistent, which is used in Chapter 4 to show asymptotic results of the estimator.

2.7.2 Implications for neural networks

As a first remark, to fulfill the presented regularity conditions, the analysis has to be restricted to NN with continuous activation functions, e.g., sigmoids and tanh. Due to the symmetries in NNs, the model structure will not be globally identifiable with a global minimizer to (2.3), but rather to a set of local minimizers that are locally identifiable. This is handled by the transformation T_i between the local minima which was introduced in Section 2.6.2.

In Hwang and Ding [1997] it is also shown that apart from the transformations generated by T_i given in Section 2.6.2, the parameter θ is globally identifiable for an irreducible two-layer NN. This holds for an NN where the activation function $\sigma(\cdot)$ is such as the class of functions $\{\sigma(ax + b), a > 0\} \cup \{\sigma \equiv 1\}$ is linearly independent. This condition can be interpreted as that the contribution from all the hidden nodes in $\mathbf{h}^{(1)}$ is different and not constant.

Next, it will be discussed which sources that contribute to the uncertainties in the estimated parameters and how they affect the prediction accuracy.

2.8 Different sources of uncertainty

The purpose of this section is to quantify different sources of uncertainty and to get an explicit expression for the contribution of each source to the total uncertainty. Assume that the estimated (learned) model parameter vector is $\hat{\theta}_N^i \in \mathcal{D}_{\hat{\theta}}$ and the corresponding asymptotic (in the number of training data samples N) parameter estimate is $\theta^{i*} \in \mathcal{D}_{\theta^*}$. The prediction error in (2.5) can then be decom-

posed as

$$\varepsilon(\mathbf{x}_m, \hat{\theta}_N^i) = e_m + \underbrace{f^*(\mathbf{x}_m) - f(\mathbf{x}_m, \theta^{i*})}_{\text{Model error}} + \underbrace{f(\mathbf{x}_m, \theta^{i*}) - f(\mathbf{x}_m, \hat{\theta}_N^i)}_{\text{Estimation error}}. \quad (2.52)$$

Next, let $\hat{\theta}_N^c$ and θ^{c*} denote the canonical representation of $\hat{\theta}_N^i$ and θ^{i*} , respectively. Then the parameter estimate can further be decomposed as

$$\hat{\theta}_N^c = \theta^{c*} + \underbrace{\hat{\theta}_N^c - \theta^{c*} - \theta^{cb}}_{\text{Stochastic error}} + \underbrace{\theta^{cb}}_{\text{Bias error}}, \quad (2.53)$$

where θ^{cb} is the bias in the parameters due to the model error. Thus, the total uncertainty in the prediction has several components with different origins, and the following observations can be made:

- If $\mathcal{S} \in \mathcal{M}^*$, even if one was given one of the optimal parameter vectors $\theta^{i0} \in \mathcal{D}_{\theta^0}$ of the true system by an oracle one would still have an estimation error $\varepsilon(\mathbf{x}_m, \theta^{i0}) = e_m$. This since the observation noise e_m can not be predicted.
- If the parameter estimation error is sufficiently small, the prediction uncertainty that originates from the error in the parameter estimate can be found via the first order Taylor expansion

$$\begin{aligned} \hat{s}_m &= f(\mathbf{x}_m, \hat{\theta}_N^i) = f(\mathbf{x}_m, T_i(\hat{\theta}_N^c)) \\ &\approx f(\mathbf{x}_m, \theta^{i0}) + f'_\theta(\mathbf{x}_m, \theta^{i0})(\hat{\theta}_N^i - \theta^{i0}), \end{aligned} \quad (2.54)$$

from which it follows that

$$\text{Var}(\hat{s}_m) \approx f'_\theta(\mathbf{x}_m, \theta^{i0}) \text{Cov}(\hat{\theta}_N^i) f'_\theta(\mathbf{x}_m, \theta^{i0})^\top. \quad (2.55)$$

Here $f'_\theta(\mathbf{x}_m, \theta^{i0}) \triangleq \frac{\partial}{\partial \theta} f(\mathbf{x}_m, \theta)|_{\theta=\theta^{i0}}$ is the Gradient of the parametric model with respect to the parameter vector θ . Furthermore, $\text{Cov}(\hat{\theta}_N^i)$ is the covariance matrix of the parameter estimate.

- If $\mathcal{S} \notin \mathcal{M}^*$, which is the typical case in practice, one has to replace θ^{c0} in (2.54) and (2.55) by θ^{c*} . This will introduce a systematic deterministic uncertainty in the estimate. In contrast to the estimation error, that decays with the size of the training data set, this model error is independent of the training data size.

Using an example inspired by the estimation of NTF from wheel slip by an NN, these different sources of uncertainty and error are now demonstrated. One could consider an NN as a collection of nonlinear basis functions that are linearly combined in the output layer [Sjöberg et al., 1994]. Using this view one could write the NN in (2.18) as

$$f(\mathbf{x}_m, \theta) = \varphi_m^\top \theta, \quad (2.56a)$$

where

$$\boldsymbol{\varphi}_m^\top = \left(\mathbf{h}^{(L)} \quad 1 \right)^\top, \quad (2.56b)$$

and

$$\boldsymbol{\theta} = W^{(L)}. \quad (2.56c)$$

Example 2.5

Consider the same nonlinear system as in Example 2.2, i.e., (2.45) that describes how the NTF depends on the wheel slip, which is further described in Section 6.1. Assume that this can be described exactly by the model given by

$$f^0(x_m, \boldsymbol{\theta}_1) = \boldsymbol{\varphi}_{1,m}^\top \boldsymbol{\theta}_1, \quad (2.57a)$$

$$\boldsymbol{\varphi}_{1,m} = \left(\sigma(6x_m + 6) \quad \sigma(-40x_m + 0.0061) \quad \sigma(-6.8x_m + 0.0036) \quad 1 \right)^\top, \quad (2.57b)$$

for the parameters $\boldsymbol{\theta}_1 = (0.40 \quad 1.44 \quad -0.60 \quad -0.42)^\top$. Hence the true system is in this model set, $\mathcal{S} \in \mathcal{M}^*$. Then assume that one also has the model set described by a model structure given by

$$f^*(x_m, \boldsymbol{\theta}_2) = \boldsymbol{\varphi}_{2,m}^\top \boldsymbol{\theta}_2, \quad (2.58a)$$

$$\boldsymbol{\varphi}_{2,m} = \left(\sigma(-40x_m + 0.0061) \quad \sigma(-6.8x_m + 0.0036) \quad 1 \right)^\top, \quad (2.58b)$$

The true system is not here included in the model set parametrized by the model structure in (2.58), i.e., $\mathcal{S} \notin \mathcal{M}^*$. Here the problem of estimating a model from a signal generated by (2.45) distorted with Gaussian noise is considered. This can be done using both models from a model set that contains the true system as well as models from a model set that does not contain the true system. That is, using models on the form given in (2.57) and (2.58), respectively.

Both models are linear in the parameters, hence the parameters can be estimated by the linear least squares solution (LLSS). The noise variance for the simulation is set to 10^{-3} , which gives a signal to noise ratio (SNR) of 30 dB, and the number of available observations used to estimate the models varies between 10 and 1000.

The different contributions of uncertainty are best visualized using the estimation of the parameters. In Figure 2.6a the estimate of the parameter corresponding to the third nonlinear basis function in (2.57) (second in (2.58)), is plotted as more samples are used to estimate the models. These two parameters should converge to the same value, but due to the systematic deterministic uncertainty there is a gap between the the parameter of the model from the model set which contain the true system and the parameter the model which model set does not contain the true system. The stochastic error due to the observation noise can be seen as oscillations of the estimated parameter value around the true value. When studying the estimated function it is hard to distinguish the systematic deterministic uncertainty from the bias in the model. In Figure 2.6b the two models

are plotted together with the true signal and the observations. Without knowing the signal from the true system it is nearly impossible to distinguish the predictions done by the different models, even though one includes the true system while the other does not.

In Chapter 4 a method to obtain prediction error variance for the canonical model is provided. In Chapter 5, it is proven that the expression obtained by the non-canonical model is identical to the expression obtained by the canonical model. Hence, one can use the practically available non-canonical model to obtain the prediction error variance.

(a) Estimation of the parameter, corresponding to one of the shared nonlinear basis functions, for different number of samples.

(b) Estimation of the NTF.

Figure 2.6: Illustration of the stochastic and systematic deterministic uncertainty, both using a model with and without the true system in the model set, see (2.57) and (2.58).

3

Related approaches to quantifying uncertainties in neural networks

Despite the huge success of NNs in solving very hard regression and classification problems, their applications in safety-critical applications, such as autonomous vehicles, are limited. This is due to the lack of ability to assess the *uncertainty* associated with the computed predictions, see e.g., [BBC News, 31 March 2018, Eykholt et al., 2018, NTSB, 2018, Reuters, 21 Aug 2017]. This has led to a recent increase in interest in uncertainty for predictions done by NNs, [Eldesokey et al., 2018, Gal and Ghahramani, 2016, Garnelo et al., 2018, Gustafsson et al., 2019a, Ilg et al., 2018, Kendall and Gal, 2017, Postels et al., 2019, Teye et al., 2018, Wilson et al., 2016b]. Many approaches to include uncertainty in the NN predictions have been suggested. This chapter aims to present a survey of some of the most commonly used approaches, as well as put the suggested approach in context to previous work.

3.1 Linearization

Uncertainty propagation through linearization of the model parameters, as described in Chapter 4, is a standard tool within estimation theory, system identification, and statistical theory subjected to analyze the uncertainty of a statistical nonlinear model disturbed with some noise [Gustafsson, 2018, Liero and Zwanzig, 2011, Ljung, 1999]. One reason for the popularity of the linearization approach is the close connection to the information matrix and the Cramér-Rao lower bound (CRLB), i.e., a lower bound on the variance of the estimated model. These models can for example be NNs, but in the literature of NNs this analysis is not standard.

The approach proposed in this thesis in Chapter 4 and Chapter 5, is an extension of this linearization approach. The resulting expression for the uncertainty in the prediction has been presented in related publications [Chrysosolouris et al.,

1996, He and Li, 2011, Hwang and Ding, 1997, Papadopoulos et al., 2001, Rivals and Personnaz, 2000], but with slight difference in the derivations. Using the notion of model sets, the suggested approach can handle cases when NN are not enabling to exactly represent the true system. This allows us to extend the linearization approach to handle the case where an overparameterized model is used. This is done by both including the transformation T_i to cover the case with redundant parameters, as well as to cover the case when more flexible models are used. In practice, for black-box models such as NNs, models that are too flexible are commonly used. Hence it is essential to have the tools to handle overparameterized models.

3.2 Ensemble techniques

A straightforward approach to quantify the uncertainty in a model is to generate multiple realizations and calculate the variance of the ensemble of realizations. To create these realizations, either the data used to create them should be perturbed by, e.g., adding noise to the observations or changing the split between training data and validation data, or the parameter initialization should be different between the different realizations.

The idea of combining models has also been shown to improve the prediction performance, see e.g. [Carney et al., 1999, Ilg et al., 2018]. This approach is however not feasible for NNs, since already training a network using a single realization can be very time-consuming. Therefore, clever ways to create these ensembles have been developed, e.g., using regularization techniques that are already used in training of NNs to prevent overfitting, see [Gal, 2017, Teye et al., 2018]. Two such approaches are MC dropout and MC batchnorm, which are presented in more detail.

3.2.1 Monte Carlo dropout

Inspired by the interpretation that an NN trained with dropout trains an ensemble of 2^n thinned networks, Gal [2017] suggested using these ensembles at prediction time to obtain the uncertainty of the NN prediction. The idea is to save some of the sampled thinned NNs that were used during training and reuse them during prediction, calculating the variance and mean of this ensemble as an estimate of the prediction and uncertainty in the prediction done by the NN. This approach to obtaining uncertainty is referred to as MC dropout, where the variance of the model can be calculated as the sample variance of the ensemble.

Gal and Ghahramani [2016] has shown that an NN trained with dropout is an approximation of the probabilistic deep Gaussian process (GP), presented by Damianou and Lawrence [2013]. This is proven by showing that the cost function minimized in an NN with dropout is equivalent to minimizing the log evidence lower bound in deep GP, for details see [Gal and Ghahramani, 2015].

There has been some criticism against MC dropout. Rather than obtaining the uncertainty in the prediction of the NN, it models the risk with the decision taken

by the NN, as well as the method not providing satisfactory confidence bounds, see [Osband, 2016, Osband et al., 2016].

3.2.2 Monte Carlo batchnorm

In similar manners as dropout can be used to create an ensemble, the regularization technique batchnorm can be used to create ensembles of NNs. Teye et al. [2018], has introduced MC batchnorm, where batchnorm is used during the prediction of the network to calculate the variance of the predicted output of the NN. The authors also prove that, much like dropout, training with batchnorm can be viewed as approximate inference in a Bayesian model.

3.3 Learning prediction uncertainties in neural networks

Instead of linearizing the NN or creating ensembles of NNs to obtain the variance of the prediction, one could include the variance as an output of the NN and let it learn its own uncertainty on the prediction. This implies also that the cost function of the NN has to be changed to take into consideration this new output. Many different approaches has been suggested including Eldesokey et al. [2018], Gustafsson et al. [2019a], Kendall and Gal [2017].

3.3.1 Gaussian model

Assuming that the likelihood of the data is Gaussian and letting the NN learn the logarithm of the variance and the mean, i.e.,

$$p(y_m | \mathbf{x}_m; \boldsymbol{\theta}) = \mathcal{N}(y_m; [f(\boldsymbol{\theta}, \mathbf{x}_m)]_1, \exp([f(\boldsymbol{\theta}, \mathbf{x}_m)]_2)), \quad (3.1a)$$

$$f(\boldsymbol{\theta}, \mathbf{x}_m) \approx [\mu_y, \log(\sigma_y^2)]^\top, \quad (3.1b)$$

where μ_y and σ_y^2 is the mean and the variance of the signal y_m at time instance m , respectively. The maximum likelihood estimate (MLE) could then be found by minimizing the cost function

$$V_N(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N \frac{(y_n - [f(\boldsymbol{\theta}, \mathbf{x}_n)]_1)^2}{\exp([f(\boldsymbol{\theta}, \mathbf{x}_n)]_2)} + [f(\boldsymbol{\theta}, \mathbf{x}_n)]_2. \quad (3.2)$$

The model described in (3.1b) is referred to as the *Gaussian model* by [Gustafsson et al., 2019a, Kendall and Gal, 2017]. To make use of this modified cost function, the structure of the NN has to change.

There are different ways to change the structure of the NN. A naive approach would be to add the variance as a second output. This will be referred to as a one-headed network, see Figure 3.1a for an illustration. Another way is to first have a base network that is later split into two different heads, one head for the mean and one for the variance, see Figure 3.1b for illustration.

Figure 3.1: Illustration of the different NN structures. Blue circles represent hidden layers and arrows the connection between the layers.

3.3.2 Energy-based model

Another approach is to use an energy-based model where the NN, denoted $f(\mathbf{x}_m, y_m, \theta)$, is supposed to learn the energy function $\epsilon(\mathbf{x}_m, y_m)$, see [Gustafsson et al., 2019a]. These models are trained by minimizing the negative log-likelihood assuming that the probability of a new output y_m given the input $\mathbf{x}_m$ is given by

$$p(y_m|\mathbf{x}_m) = \frac{e^{-\epsilon(\mathbf{x}_m, y_m)}}{\int e^{-\epsilon(\mathbf{x}_m, y_m)} dy_m}. \quad (3.3)$$

The negative log-likelihood can be approximated by sampling from a proposal distribution $Q(y^{(n,k)}|\mathbf{x}_m)$ that depends on the true output values y_m ,

$$-\log p(y_m|\mathbf{x}_m; \theta) \approx \log \left(\frac{1}{K} \sum_{k=1}^K \frac{e^{f(\mathbf{x}_m, y^{(k)}, \theta)}}{Q(y^{(m,k)}|y_m)} \right) - f(\mathbf{x}_m, y_m, \theta), \quad (3.4)$$

and the MLE is given by the model that is minimizing the cost function

$$V_N(\theta) = \frac{1}{N} \sum_{n=1}^N \log \left(\frac{1}{K} \sum_{k=1}^K \frac{e^{f(\mathbf{x}_n, y^{(n,k)}, \theta)}}{Q(y^{(n,k)}|y_n)} \right) - f(\mathbf{x}_n, y_n, \theta). \quad (3.5)$$

In Gustafsson et al. [2019a], the suggested proposal distribution is a mixture of equally weighted Gaussian centered at the output value y_m , i.e.,

$$Q(y^{(n,k)}|y_m) = \frac{1}{L} \sum_{l=1}^L \mathcal{N}(y^{(n,k)}; y_m, \sigma_l^2). \quad (3.6)$$

For new data $\mathbf{x}_m^*$ the predicted output is found by maximizing the estimated energy function (likelihood) with respect to the output in the training data set, i.e.,

Figure 3.2: Two headed structure

$y_m^* = \arg \max_y f(\mathbf{x}_m^*, y, \theta)$, where y is sampled from the output in the training data set. That is, using observation of the output in the training data, a grid of the approximation of the energy function is created, where the output that gives the highest likelihood is selected as the predicted output for the new sample. In Gustafsson et al. [2019a], the prediction is later also improved with a gradient-based refinement step to improve the accuracy of the predicted new output.

The structure of the network can be seen in Figure 3.2, where there are two different sorts of inputs to the base of the NN; one for the input $\mathbf{x}_m$ and one for the output y_m , that are combined to one output further into the network.

3.4 Gaussian processes

An example of a non-parametric model is GPs. The concept of uncertainty is naturally included in GPs, where the kernel function represents some covariance function for the learned function. A GP is a collection of random variables that have a joint Gaussian distribution. It can be described completely by a mean function and a kernel function. Assuming $f(\mathbf{x}_m)$ to be some process, it can be approximated with a GP as

$$f(\mathbf{x}_m) \sim \mathcal{GP}(m(\mathbf{x}_m), k(\mathbf{x}_m, \mathbf{x}')), \quad (3.7)$$

where $m(\mathbf{x}_m)$ is the mean of the real process, $k(\mathbf{x}_m, \mathbf{x}')$ is the covariance (called kernel function), and $\mathbf{x}'$ is data from $\mathbf{x}_{1:N}$ which is data used to train the GP. This interpretation of a GP is sometimes referred to as the *function-space view*. One example of such kernel is the squared exponential¹

$$k(\mathbf{x}_m, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2L^2}(\mathbf{x}_m - \mathbf{x}')^2\right), \quad (3.8)$$

where L is the length scale, and σ_f^2 the signal variance.

An alternative and equivalent view is to think of GPs as a linear regression $f(\mathbf{x}_m) = \boldsymbol{\varphi}^\top(\mathbf{x}_m)\boldsymbol{\theta}$ with a basis of a finite number of functions $\boldsymbol{\varphi}^\top(\mathbf{x}_m)$, and a Gaussian prior on the parameters $\boldsymbol{\theta}$. This is referred to as the *weight-space view*, [Williams and Rasmussen, 2006]. The neat thing with the weight-space view of GPs is that if the basis function is selected such that they are eigenfunctions of

¹Sometimes called Radial Basis Functions or Gaussian.

the kernel function, using *Mercer's theorem*, the kernel can be written as

$$k(\mathbf{x}_m, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i [\boldsymbol{\varphi}^\top(\mathbf{x}_m)]_i [\boldsymbol{\varphi}^\top(\mathbf{x}')]_i, \quad (3.9)$$

which is sometimes called the *kernel trick*, [Theodoridis, 2015, Williams and Rasmussen, 2006]. Here λ_i is the eigenvalues of the i th eigenfunction $[\boldsymbol{\varphi}^\top(\mathbf{x}_m)]_i$. The sum can also be truncated, i.e., to find a GP that explains your data, one can parametrize the kernel using a finite number of basis functions. Where the values of the parameters come from the solution of the linear regression problem. Using the weight-space view, it is shown in Solin and Särkkä [2019] that using just a couple of Laplacian polynomials as basis functions, any smooth kernel function, such as the squared exponential in (3.8), can accurately be approximated. The derivation is using the fact that the covariance operator can be written as a convolution between the kernel and a function. Then, it can be shown that the Laplacian operator can be assigned as a formal kernel.

GPs are expressive models and a powerful tool to express general dependence in data. As a result GPs have shown great potential to learn nonlinear state-space models, [Svensson and Schön, 2017], but due to high computational cost, they have not been used so frequently. In recent years this has changed, which has been enabled by both increased computational power as well as the smart implementation of GPs, e.g., [Svensson and Schön, 2017, Wilson and Nickisch, 2015]. Another reason for the increased use of GPs is the use of NNs to learn the kernel function of the GP, see e.g., [Al-Shedivat et al., 2017, Wilson et al., 2016a,b].

3.5 Bayesian neural networks

An alternative approach to quantify uncertainty in the NN predictions is to assume some posterior distribution, $p(\boldsymbol{\theta}|y_m)$, for the parameters of the NN, $\boldsymbol{\theta}$, given some observations of the output y_m . This view of NNs falls into the category of BNN. Then the task becomes to learn the posterior $p(\boldsymbol{\theta}|y_m)$ of the parameters given the data, rather than to find $\hat{\boldsymbol{\theta}}_N$ using optimization algorithms such as RMS-prop or ADAM. Then there is no need to calculate the prediction error uncertainty using, e.g., (5.43), one could evaluate the NN with different values of parameters sampled from $p(\boldsymbol{\theta}|y_m)$. To learn the posterior, a sampling scheme has to be used. The Metropolis-Hastings sampling algorithm is an example of such a scheme. Given the current sample of the parameters $\boldsymbol{\theta}^{(i)}$, the proposed value next of the parameters $\boldsymbol{\theta}^*$ for the sampling algorithm is calculated as

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}^{(i)} + Q, \quad (3.10)$$

where Q is a proposed density. The proposed density describes how to take the next sample of $\boldsymbol{\theta}$, and should preferably be close to the prior of the parameters. The proposed new value of the parameter is accepted if a sample from a uniform

distribution on the interval $[0, 1]$ is smaller than

$$\alpha = \min \left(\frac{p(\theta^*|y_m)Q(\theta^{(i)}|\theta^*)}{p(\theta^{(i)}|y_m)Q(\theta^*|\theta^{(i)})}, 1 \right). \quad (3.11)$$

where α is called the acceptance probability. For more details on Metropolis-Hastings sampling and other sampling schemes, see [Dobson and Barnett, 2008].

In this work, the proposal distribution will be a random walk, e.g., $Q \sim \mathcal{N}(0, 1)$. Then, $Q(\theta^{(i)}|\theta^*) = Q(\theta^*|\theta^{(i)})$, since the random walk is symmetric. Assuming that the observation noise is Gaussian distributed, the likelihood of an observation given the current value of the parameter, $p(y_m|\theta)$ becomes Gaussian distributed. The prior on the data $p(y_m)$ and the prior is the same for all samples and one can also assume that the prior of the parameters, $p(\theta^*)$ and $p(\theta^{(i)})$, is the same. Then, using Bayes formula

$$p(\theta|y_m) = \frac{p(y_m|\theta)p(\theta)}{p(y_m)}, \quad (3.12)$$

the quotient in (3.11), becomes a quotient between the likelihoods of sample y_m given the parameters $\theta^{(i)}$ and θ^* . Taking the logarithm of these expressions, one is left with comparing the mean squared error between the new and old sample to some threshold value, i.e., comparing

$$\frac{1}{N} \sum_{n=1}^N (y_n - f(\theta^*, \mathbf{x}_n))^2 - \frac{1}{N} \sum_{n=1}^N (y_n - f(\theta^{(i)}, \mathbf{x}_n))^2, \quad (3.13)$$

to the logarithm of a sample from an uniform distribution on the interval $[0, 1]$. For more details and other sampling schemes see [Dobson and Barnett, 2008].

BNN were first popularized in the '90s by [MacKay, 1992, Neal, 1996], but due to high computational cost they never took off. Thanks to more powerful computers and smart implementations of BNNs they have recently become more popular in the current wave of machine learning research [Blundell et al., 2015, Gustafsson et al., 2019b, Kendall and Cipolla, 2016, Osband et al., 2016]. There is also a close connection between BNN, where a prior is assumed on the parameters, and GPs. For example, Neal [1996] shows that a large number of different BNN structures will converge to GPs as the number of nodes in the BNN tends to infinity.

4

Parameter and prediction uncertainty for canonical neural networks

Given the canonical model, defined in Section 2.6.1, this chapter provides an analysis of how the prediction error variance depends on if the true system is in the model set or not. The analysis will begin by investigating the parameter covariance to later show how the parameter covariance can be propagated to variance in the output. The main results in this chapter are a review of the results presented in Ljung [1999] and Ljung and Caines [1980]. That is, that asymptotically the random variable $\sqrt{N}(\hat{\theta}_N^c - \theta^{c*})$, i.e., the parameter estimation error, is Gaussian distributed under weak assumptions. Furthermore, in this chapter also includes an analysis of how the parameter covariance can be propagated to the variance of the output, i.e., prediction error variance. Compared to Ljung [1999] and Ljung and Caines [1980] that are focusing on dynamic systems such as nonlinear autoregressive with external input, the focus in this thesis is instead to consider nonlinear static models, which neural networks (NN) are an example of.

4.1 Parameter covariance

First, define the prediction error and the cost function for the canonical model as

$$\varepsilon_c(\mathbf{x}_m, \theta^c) \triangleq y_m - \hat{s}_m = f^*(\mathbf{x}_m) + e_m - f_c(\mathbf{x}_m, \theta^c), \quad (4.1a)$$

and

$$V_{N,c}(\theta^c) \triangleq \frac{1}{N} \sum_{n=1}^N \|\varepsilon_c(\mathbf{x}_n, \theta^c)\|^2, \quad (4.1b)$$

Then the estimated parameters $\hat{\theta}_N^c$ are found by minimizing (4.1b), i.e.,

$$\hat{\theta}_N^c = \arg \min_{\theta^c} V_{N,c}(\theta^c). \quad (4.2)$$

4.1.1 True system not in model set

The analysis begins with the general case where the true system is not necessarily in the model set, $\mathcal{S} \notin \mathcal{M}^*$, and then derive the other one as a special case. From the definition of the canonical representation of the model and the transformation between the canonical and non-canonical representation in Section 2.6.1, there exists a unique $T_i(\theta^{c*}) \in \mathcal{D}_{\theta^*}$. Since $\hat{\theta}_N^c$ is minimizing the cost function in (4.1b), $\hat{\theta}_N^c \rightarrow \theta^{c*}$ with probability 1 as $N \rightarrow \infty$, [Ljung, 1999]. Define the gradient of the prediction error with respect to the canonical parameter as

$$\psi(\mathbf{x}_m, \theta^{c*}) \triangleq -\frac{\partial}{\partial \theta^c} \varepsilon_c(\mathbf{x}_m, \theta^c) \Big|_{\theta^c = \theta^{c*}} = \frac{\partial}{\partial \theta^c} f_c(\mathbf{x}_m, \theta^c) \Big|_{\theta^c = \theta^{c*}}, \quad (4.3)$$

and the gradient of the cost function for the canonical model as

$$-V'_{N,c}(\theta^{c*}) \triangleq -\frac{\partial}{\partial \theta^c} V_{N,c}(\theta^c) \Big|_{\theta^c = \theta^{c*}} = \frac{1}{N} \sum_{n=1}^N \psi(\mathbf{x}_n, \theta^{c*}) \varepsilon_c(\mathbf{x}_n, \theta^{c*}). \quad (4.4)$$

For stochastic variables g_n , the asymptotic mean¹ is defined as

$$\bar{E}[g_n] \triangleq \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=1}^N E[g_n], \quad (4.5)$$

where $E[\mathbf{x}]$ is the mathematical expectation operator. Define the asymptotic cost function as

$$\bar{V}_c(\theta^c) \triangleq \bar{E}[\varepsilon_c^2(\mathbf{x}_n, \theta^c)]. \quad (4.6)$$

From the definition of θ^{c*} being a parameter minimizing (4.1b)

$$\bar{V}'_c(\theta^{c*}) = \frac{\partial}{\partial \theta^c} \bar{V}_c(\theta^c) \Big|_{\theta^c = \theta^{c*}} = -\bar{E}[\psi(\mathbf{x}_n, \theta^{c*}) \varepsilon_c(\mathbf{x}_n, \theta^{c*})] = 0. \quad (4.7)$$

In Ljung and Caines [1980] it is shown that (4.4) can be written as a sum of random variable $\psi(\mathbf{x}_m, \theta^{c*}) \varepsilon_c(\mathbf{x}_m, \theta)$ with zero mean and

$$D_N = E \left[\frac{1}{N} \sum_{n=1}^N \psi(\mathbf{x}_n, \theta^{c*}) \varepsilon_c(\mathbf{x}_n, \theta^{c*}) - \bar{E}[\psi(\mathbf{x}_n, \theta^{c*}) \varepsilon_c(\mathbf{x}_n, \theta^{c*})] \right]. \quad (4.8)$$

which quickly tends to zero. Using the central limit theorem, one can show that (4.4) is asymptotically normal distributed, for details see [Ljung and Caines, 1980]. This is guaranteed by the regularity conditions given in Section 2.7.

¹For the asymptotic mean, the expectation is taken over the observation noise, while to take in which order the data is revived in consideration, the limit is included.

From assumptions on the models $f(\mathbf{x}_m, \boldsymbol{\theta})$ in the model set, presented in Section 2.7, the second derivative of the asymptotic cost function

$$\bar{V}_c''(\boldsymbol{\theta}^{c*}) \triangleq \frac{\partial^2}{\partial(\boldsymbol{\theta}^c)^2} \bar{V}_c(\boldsymbol{\theta}^c) \Big|_{\boldsymbol{\theta}^c = \boldsymbol{\theta}^{c*}}, \quad (4.9)$$

is positive definite. Then, as the final result

$$\sqrt{N}(\hat{\boldsymbol{\theta}}_N^c - \boldsymbol{\theta}^{c*}) \rightarrow \mathcal{N}(0, P_{\boldsymbol{\theta}^c}(\boldsymbol{\theta}^{c*})) \quad \text{as } N \rightarrow \infty, \quad (4.10)$$

where

$$P_{\boldsymbol{\theta}^c}(\boldsymbol{\theta}^{c*}) = (\bar{V}_c''(\boldsymbol{\theta}^{c*}))^{-1} Q (\bar{V}_c''(\boldsymbol{\theta}^{c*}))^{-1}, \quad (4.11a)$$

$$Q = \lim_{N \rightarrow \infty} NE\{(V'_{N,c}(\boldsymbol{\theta}^{c*}))^\top (V'_{N,c}(\boldsymbol{\theta}^{c*}))\}. \quad (4.11b)$$

This result can be summarized as follows.

Theorem 4.1. *Consider the estimation of $\hat{\boldsymbol{\theta}}_N^c$ given in (4.2) and (4.1b) for the canonical model. Assume that the model structure used to parametrize the model follows the regularity conditions described in Section 2.7. Also, assume that*

$$\hat{\boldsymbol{\theta}}_N^c \rightarrow \boldsymbol{\theta}^{c*}, \quad \text{w.p. 1 as } N \rightarrow \infty, \quad (4.12a)$$

$\bar{V}_c''(\boldsymbol{\theta}^{c*})$ is positive definite, and that

$$\sqrt{N}D_N \rightarrow 0, \quad \text{as } N \rightarrow \infty, \quad (4.12b)$$

where D_N is defined in (4.8). Then

$$\sqrt{N}(\hat{\boldsymbol{\theta}}_N^c - \boldsymbol{\theta}^{c*}) \rightarrow \mathcal{N}(0, P_{\boldsymbol{\theta}^c}(\boldsymbol{\theta}^{c*})) \quad \text{as } N \rightarrow \infty, \quad (4.13)$$

where $P_{\boldsymbol{\theta}^c}(\boldsymbol{\theta}^{c*})$ is given in (4.11).

Proof: See Ljung and Caines [1980]. The main idea of the proof is to split

$$-\sqrt{N}V'_{N,c}(\boldsymbol{\theta}^{c*}) = \frac{1}{\sqrt{N}} \sum_{n=1}^N \psi(\mathbf{x}_n, \boldsymbol{\theta}^{c*}) \varepsilon_c(\mathbf{x}_n, \boldsymbol{\theta}^{c*}), \quad (4.14)$$

into two parts, one that satisfies a certain independence condition between its terms and one that is small. $\square$

Connecting to Section 2.3, one can recognize $\bar{V}_c''(\boldsymbol{\theta}^{c*})$ as the Hessian of the cost function, where the solution of the system of equations

$$\bar{V}_c''(\hat{\boldsymbol{\theta}}_N^c) \Delta \boldsymbol{\theta}^c = -V'_{N,c}(\hat{\boldsymbol{\theta}}_N^c), \quad (4.15)$$

gives the Newton search direction. Similar calculation as in (2.40) would lead to the same result as in Theorem 4.1. Hence again, one can see that the uncertainty of the parameters is proportional to the inverse of the Hessian of the cost function.

4.1.2 True system in model set

If $S \in \mathcal{M}^*$, and $T_i(\theta^{c0}) \in \mathcal{D}_{\theta^0}$, the prediction error $\varepsilon_c(\mathbf{x}_m, \theta^{c0}) = e_m$ has zero mean and variance λ_0 by assumption. As a first consequence, $D_N = 0$. Furthermore, the covariance in (4.11a) can be greatly simplified by noting that $Q = \lambda_0 \tilde{V}_c''(\theta^{c0})$, and as a result, (4.11a) simplifies to

$$P_{\theta^c}(\theta^{c0}) = \lambda_0 \mathcal{I}_{\theta^c}^{-1}, \quad (4.16a)$$

$$\mathcal{I}_{\theta^c} = \bar{E}[\psi^\top(\mathbf{x}_n, \theta^{c0})\psi(\mathbf{x}_n, \theta^{c0})], \quad (4.16b)$$

where if e_m is Gaussian distributed, $\mathcal{I}_{\theta^c}$ is the Fisher information matrix for θ^c .

Returning to the calculations in Section 2.3, this would be the same as if the second term in the Hessian in (2.26b) would be zero. The situation would be the same as if the model had been linear in the parameters. Hence, if the true model is in the model set, it is a good approximation that the second term in the Hessian of the cost function is dominated by the first term.

In practice, one way to make it highly likely that the true system is included in the model set is to choose a very rich model set, and then use regularization to avoid overfitting the model to the data. L^2 -regularization is one example of a regularization that introduces an explicit cost for using models with many parameters, i.e., adding the l_2 -norm of the parameters to the loss function. This would imply that a scaled identity matrix is added to the Fisher information $\mathcal{I}_{\theta^c}$. In a Bayesian setting, the added identity is interpreted as a prior on the weights. This in turn would be equivalent to adding additional knowledge about the true system in terms of model order and smoothness of the function $f^*(\mathbf{x}_m)$. It is in particular necessary to use regularization if the number of parameters exceeds the number of training data samples. One downside with regularization is that it will introduce a bias in the estimated parameters, which in turn will propagate to a bias of the model.

4.1.3 Cramér-Rao lower bound

From classic statistic theory and system identification literature, e.g., [Kay, 1993, Liero and Zwanzig, 2011, Ljung, 1999], it is known that if $f^* \in \mathcal{M}^*$, any unbiased estimator $\hat{\theta}_N^c$ has a lower bound on the stochastic uncertainty given by

$$\text{Cov}(\hat{\theta}_N^c) \geq \frac{1}{N} P_{\theta}(\theta^{c0}), \quad (4.17)$$

where $\geq$ denotes that the difference between the left-hand side and the right-hand side is positive semidefinite. This bound is referred to as the CRLB. In particular, if the noise in (2.1b) is Gaussian distributed, the central limit theorem can be used to show that the parameter estimated by (4.2) is the MLE with its asymptotic covariance given by CRLB in (4.16a). This gives us that if one assumes that a given NN is large enough to include the true system in the model set, $S \in \mathcal{M}^*$, (4.16a) can be used to calculate a lower bound on the variance of the parameters in an NN.

4.1.4 Approximating the variance expressions

The previous results presented in this chapter are asymptotic in the number of points in the training data. In practice, there is nothing as an infinite number of data points, hence, results from a finite number of data points are required.

If the parameters θ^{c0} and λ_0 are unknown, $p_{\theta^c}(\theta^{c0})$ and λ_0 in (4.11a) can be approximated with data. That is having processed N data points, the approximation

$$\hat{P}_{N,c} = \hat{\lambda}_{N,c} \left(\frac{1}{N} \sum_{n=1}^N \psi^\top(\mathbf{x}_n, \hat{\theta}_N^c) \psi(\mathbf{x}_n, \hat{\theta}_N^c) \right)^{-1}, \quad (4.18a)$$

$$\hat{\lambda}_{N,c} = \frac{1}{N} \sum_{n=1}^N \varepsilon_c^2(\mathbf{x}_n, \hat{\theta}_N^c), \quad (4.18b)$$

can be used, where θ^{c0} is approximated by $\hat{\theta}_N^c$. This gives us that $\text{Cov}(\hat{\theta}_N^c)$ in (2.55) can be approximated by $\hat{P}_{N,c}/N$. Asymptotically, these expressions converge to the true values.

If the $\mathcal{S} \in \mathcal{M}^*$, the parameter estimate for an NN has a lower bound on the stochastic uncertainty that can be approximated by (4.17) with the right-hand side approximated with $\hat{P}_N$ given in (4.18) for a large but finite N .

4.1.5 Parameter covariance for irreducible neural networks

Knowing that parameters for NNs will converge to different clusters, as seen in Figure 2.4b, one question which naturally comes to mind is that if the parameter covariance is the same for the different clusters? That is, how will the transformation described in Section 2.6 affect the uncertainty of the parameter estimate after they have converged to one of the clusters. To answer this, one has to define what one means with equally uncertain. To compare the uncertainty of parameters between the different clusters, one could compare the determinant of their respective parameter covariance matrices.

From Remark 2.1 in Section 2.6, the transformation between parameters in different clusters of irreducible parameters is known to be linear, hence

$$\text{Cov}(\hat{\theta}_N^c) = \text{Cov}(T_i \hat{\theta}_N^i) = T_i \text{Cov}(\hat{\theta}_N^i) T_i^\top. \quad (4.19)$$

Then for the determinant

$$\det(\text{Cov}(\hat{\theta}_N^c)) = \det(T_i) \det(\text{Cov}(\hat{\theta}_N^i)) \det(T_i^\top). \quad (4.20)$$

Since T_i is composed by permutation matrices and lower triangular matrices² with ± 1 on the diagonal, the determinant of the transformation T_i has to be either ± 1 . Hence

$$\det(\text{Cov}(\hat{\theta}_N^c)) = \det(\text{Cov}(\hat{\theta}_N^i)). \quad (4.21)$$

²Or upper triangular depending how the parameters are sorted

which tells us that the parameters are equally uncertain. Another way to compare the uncertainty between the different clusters is to compare the trace of their parameter covariance matrices. In the following example, the aim is to show that the uncertainty in the different clusters in Figure 2.4b are similar, regarding both equal determinant and trace of the parameter covariance matrix.

Example 4.1

Given the same setup as in Example 2.2 where a 2-layer NN is trained on the output from signals generated by (2.45). Studying the realization chosen as the canonical (one of the realizations in the orange cluster), and one realization in another cluster (light blue cluster in Figure 2.4b), the transformation T_i between the canonical and non-canonical parameters is given by changing hidden nodes with each other, i.e., (2.49). The different parameter covariances are given by

$$\text{Cov}(\hat{\theta}_N^i) = 10^{-7} \begin{pmatrix} 51300 & -90300 & 858 & 176 & -2090 & 3150 & -557 \\ -90300 & 324000 & -2580 & -18.0 & 6410 & -7810 & 780 \\ 858 & -2580 & 364 & 123 & -80 & 93.0 & -27.7 \\ 177 & -18.0 & 123 & 99.8 & -19.3 & 21.4 & -6.34 \\ -2090 & 6410 & -80.3 & -19.3 & 172 & -196 & 14.4 \\ 3150 & -7810 & 93.0 & 21.4 & -196 & 2486 & -29.2 \\ -557 & 780 & -27.7 & -6.34 & 14.4 & -29.2 & 9.44 \end{pmatrix}, \quad (4.22a)$$

and

$$\text{Cov}(\hat{\theta}_N^c) = 10^{-7} \begin{pmatrix} 254000 & -70200 & -24.6 & -2010 & -5990 & 4860 & 628 \\ -70120 & 39742 & 147 & 670 & 2400 & -1550 & -446 \\ -24.6 & 147 & 78.4 & 96.7 & 16.9 & -15.1 & -5.15 \\ -2010 & 670 & 96.7 & 288 & 71.2 & -60.1 & -22.2 \\ -5990 & 2400 & 16.9 & 71.2 & 187 & -146 & -23.0 \\ 4860 & -1550 & -15.1 & -61.0 & -146 & 128 & 10.9 \\ 628 & -446 & -5.15 & -22.2 & -23.0 & 10.9 & 7.71 \end{pmatrix}. \quad (4.22b)$$

The Frobenius norm of the difference between (4.22a) and the transformed version of (4.22b) using the transformation defined in (2.49) computed as

$$\|\text{Cov}(\hat{\theta}_N^c) - T_i \text{Cov}(\hat{\theta}_N^i) T_i^\top\|_F = 0.0017. \quad (4.23)$$

Hence one can conclude that the covariance is almost the same for the different clusters.

In Figure 4.1, the eigenvalues for the parameter covariances in the different clusters are plotted. For all the clusters and the canonical representations, the determinant (product of eigenvalues) and trace (sum of eigenvalues) for the parameter covariance is calculated. These are both measures of how uncertain the parameters are. As one can see they are similar for all the different realizations. One can also notice that it is a big difference in magnitude between the most certain and the most uncertain direction. The uncertainty looks similar and almost

Figure 4.1: The eigenvalues of the parameter covariance matrices for the different clusters seen in in Figure 2.4b. The cluster are color-coded the same way as in Figure 2.4b, and one can remember that the canonical parameters belong to the orange cluster. The parameter covariance is calculated using (4.25b). In the legend of the figure, the sum and product of the eigenvalues can be seen for the different clusters. One could notice that both the numbers are of the same magnitude, and that the eigenvalues of the different clusters seems to be similar as well.

identical for this small example, hence one can conclude that the covariance for all the different clusters is likely to be the same.

A natural follow-up question is how the uncertainty is distributed between the different parameters. In Figure 4.2, this is illustrated for the NN in Example 2.2 where the uncertainty in the NN is plotted. Much of the uncertainty seems to be located in the parameters closer to the input while parameters closer to the output have a lower uncertainty. This observation can also be seen in (4.22), where the parameter covariance for the first two parameters is several orders of magnitudes larger than for the rest of the parameters. Intuitively, this could be motivated by viewing the problem of training an NN as two separate problems, the first one is to find the nonlinear basis functions in the last layer, $(\mathbf{h}^{(L)})$, and the second problem to find a linear combination these basis functions that minimizes the squared error. The second problem, which is also described in (2.56), can be found as the solution to a linear least squares problem, for which one can fairly easily find the minimizing parameters. On the other hand, finding the parameters used to construct the nonlinear basis functions is a much harder problem. Hence one would expect the uncertainty for those parameters to be larger compared to the parameters that solve the linear least squares problem.

Figure 4.2: Illustration of how the parameter covariance calculated for a two layer NN with two hidden nodes is distributed between the different parameters in the NN. The parameter covariance is calculated using (4.25b), and the numbering of the parameters is the same as in Figure 2.3 and (2.46), i.e., they sorted so that parameters close to the input have lower indices. One can see that parameters close to the output have a lower variance compared to than parameters close to the input.

4.2 Prediction error variance

As stated in Section 2.8, if the prediction error is small, the covariance of the model parameters can be propagated to the variance of the model output. This is done using the first order Taylor approximation of the model around the true network parameters, sometimes also referred to as the delta method, see [Gustafsson, 2018, Liero and Zwanzig, 2011]. That is, the prediction error of the NN model is given as linearization of the model around its parameters, i.e.,

$$\hat{s}_m \approx f_c(\mathbf{x}_m, \theta^{c*}) + \psi(\mathbf{x}_m, \theta^{c0})(\hat{\theta}_N^c - \theta^{c*}). \quad (4.24a)$$

Given that $\sqrt{N}(\hat{\theta}_N^c - \theta^{c*}) \rightarrow \mathcal{N}(0, P_{\theta^c}(\theta^{c*}))$ as $N \rightarrow \infty$ it holds that

$$\sqrt{N}(\hat{s}_m - f_c(\mathbf{x}_m, \theta^{c*})) \rightarrow \mathcal{N}(0, P_{s_m}(\theta^{c*})) \quad \text{as } N \rightarrow \infty, \quad (4.24b)$$

where

$$P_{s_m}(\theta^{c*}) = \psi(\mathbf{x}_m, \theta^{c*}) P_{\theta^c}(\theta^{c*}) \psi^\top(\mathbf{x}_m, \theta^{c*}), \quad (4.24c)$$

and the variance of the prediction error is given by

$$\text{Var}(\hat{s}_m) \approx \frac{1}{N} P_{s_m}(\theta^{c*}), \quad (4.24d)$$

This expression can also be approximated having processed a finite amount of data, namely, the variance of the signal can be approximated as

$$\text{Var}(\hat{s}_m) \approx \psi(\mathbf{x}_m, \hat{\theta}_N^c) \text{Cov}(\hat{\theta}_N^c) \psi^\top(\mathbf{x}_m, \hat{\theta}_N^c), \quad (4.25a)$$

where

$$\text{Cov}(\hat{\theta}_N^c) \sim \frac{\hat{P}_{N,c}}{N}. \quad (4.25b)$$

From Figure 2.4b it is known that for irreducible NN, the parameters converges to different cluster. In Section 4.1.5, it is also shown that the parameter covariance is similar for these different clusters. This is also the case for the prediction error variance, where (2.55) and (4.25a) give the same prediction error variance for all irreducible models (including the canonical representation of the model). To show this, first notice that the derivative of a irreducible model with respect to its parameters can rewritten as

$$f'_{\theta}(\mathbf{x}_m, \theta^{i0}) \triangleq \frac{\partial}{\partial \theta^i} f(\mathbf{x}_m, \theta^i) \Big|_{\theta^i = \theta^{i0}} = \frac{\partial}{\partial \theta^i} f_c(\mathbf{x}_m, T_i(\theta^i)) \Big|_{\theta = \theta^0} \quad (4.26a)$$

$$= \frac{\partial}{\partial \theta^c} f_c(\mathbf{x}_m, \theta^c) \Big|_{\theta^c = \theta^{c0}} \frac{\partial \theta^c}{\partial \theta^i} \Big|_{\theta^i = \theta^{i0}} \quad (4.26b)$$

$$= \psi(\mathbf{x}_m, \theta^{c0}) T_i. \quad (4.26c)$$

Since T_i is invertible

$$\text{Cov}(\hat{\theta}_N^i) = T_i^{-1} \text{Cov}(\hat{\theta}_N^c) (T_i^\top)^{-1}. \quad (4.27)$$

For any irreducible model, the prediction error variance is given as by

$$f'_{\theta}(\mathbf{x}_m, \theta^{i0}) \text{Cov}(\hat{\theta}_N^i) f'_{\theta}(\mathbf{x}_m, \theta^{i0})^\top \quad (4.28a)$$

$$= \psi(\mathbf{x}_m, \theta^{c0}) T_i T_i^{-1} \text{Cov}(\hat{\theta}_N^c) (T_i^\top)^{-1} T_i \psi^\top(\mathbf{x}_m, \hat{\theta}_N^c) \quad (4.28b)$$

$$= \psi(\mathbf{x}_m, \hat{\theta}_N^c) \text{Cov}(\hat{\theta}_N^c) \psi^\top(\mathbf{x}_m, \hat{\theta}_N^c). \quad (4.28c)$$

Hence, it can be conclude that the prediction error variance for all the irreducible models is equal to the prediction error variance of the canonical representation of the model.

4.3 Summary

In this chapter, an approach to obtain prediction error variance for a model by first calculating covariance of the parameters in the model and then linearizing the model with respect to those parameters is presented. The asymptotic properties are covered as well as how the expression changes depending on if the true system is in the model set or not. Furthermore, for the example considered in Example 4.1, it is shown that it is likely that the parameter covariance is similar in the different clusters of parameters, and the prediction error variance is the same. Where the different clusters come from the ambiguities in NN, discussed in Section 2.6.2. In Figure 4.2, one can also see how the uncertainty is distributed between parameters in the NN, where it is clear that parameters closer to the input it has higher variance compared to parameters close to the output.

5

Prediction error variance for overparameterized models

In practice, one does not usually have the canonical representation of a model available but rather a non-canonical representation of the model. In this chapter, it will be shown that even though one does not have the canonical representation of the model, with some modification, the analytic expression for the prediction error variance presented in Chapter 4 can still be used.

That chapter begins with a motivating example of how the situation changes when an overparameterized model is used, followed by showing that the prediction error variance is the same for the canonical and the non-canonical representations of model if their parameters span the same model set. Then it is shown that if one uses a model from a model set that is too flexible i.e., the range of the model structure includes models richer than are needed to describe the true system, then the prediction error variance is bounded from below by the canonical representation of a model that is less flexible and which model set has a smaller range.

5.1 Overparameterization in neural networks

The use of overparameterized models is, in particular, true for NN where one often uses a network with too many nodes in the hidden layers, which can result in more ambiguities than the transformation described in Section 2.6. Let us motivate this with an example.

Example 5.1

Consider the same set up as in Example 2.2, but instead of a two-layer NN with two nodes in each hidden layer, consider a two-layer NN with five nodes in each layer, i.e., replace an NN with seven parameters with an NN with sixteen parameters. Since only seven parameters were needed to model the true system, the

overparameterized models are too flexible. One can also notice that even though they are overparameterized, the models have no redundant parameters. In the experiments, which samples that are included in the different batches, and what initial values that are used for the parameters, are the only varying conditions.

In Figure 5.1a, two of the parameters connecting the hidden layer to the output layer, $W_{11}^{(1)}$ and $W_{12}^{(1)}$, are plotted for the different realizations, with the cost function for the network in the background¹. Comparing the spread of the parameters in Figure 2.4a and Figure 5.1a, one can see that the pattern from ambiguities is not visible for the NN that are more flexible.

In Figure 5.1b the eigenvalues of the parameter covariance is plotted for a couple of realizations. One can see that the magnitude of the eigenvalues stretches for multiple decades. This will result in the parameter covariance matrix is ill-conditioned.

In this chapter, two categories of overparameterized models will be distinguished, i.e., *models with redundant parameters* and *models with more flexibility*.

Definition 5.1 (Models with redundant parameters). A model with redundant parameters is an overparameterized model which has parameters that are not adding any flexibility to the model.

In the linear case, a model with redundant parameters is a model where some of the basis functions of the model can be written as a linear combination of the other basis functions. An example of this can be seen in Example 5.2.

Definition 5.2 (Models with more flexibility). A model with more flexibility is an overparameterized model where the extra parameters increase the flexibility of the model.

An example of such model can be seen in Example 5.4.

Throughout this chapter, it is assumed that the model set is rich enough to include the true system. This is motivated by the fact that in black-box modelling, one usually chooses a model flexible enough such that the true model is included in the model set. Then, one could add regularization during the estimation of parameters of the model to counteract overfitting due to overparameterization. When the true system is included in the model set, the prediction error becomes a sequence of independent random variables with zero mean and variance λ_0 , which will simplify the expressions for the prediction error variance.

5.2 Models with redundant parameters

As a first step, let us repeat the problem formulation of Chapter 2 and the main result from Chapter 4. Given a signal model

$$s_m = f(\mathbf{x}_m, \boldsymbol{\theta}), \quad \boldsymbol{\theta} \in \mathbb{R}^d, \quad (5.1a)$$

$$y_m = s_m + e_m, \quad (5.1b)$$

¹If the model would have been irreducible, this would have been the canonical model.

(a) Optimal parameters.

(b) Eigenvalues of the parameter covariance

Figure 5.1: Illustration of how symmetries in the NN can lead to different optimal parameters. In (a) the optimal parameters for two-layer NNs with five nodes in the hidden layer for different initialisations on the weights. The plotted parameters is connecting the hidden layer to the output layer. In (b) the eigenvalues of the parameter covariance are plotted for a couple of realizations. The parameter covariance is calculated using (4.25b). One can see that there is a large difference in magnitude between the eigenvalues.

with a minimal canonical representation given by

$$s_m = f_c(\mathbf{x}_m, \boldsymbol{\theta}^c), \quad \boldsymbol{\theta}^c \in \mathbb{R}^{d_c}, \quad (5.1c)$$

$$y_m = s_m + e_m, \quad (5.1d)$$

where one is interested in fitting the data y_m to the models $f(\mathbf{x}_m, \boldsymbol{\theta})$ and $f_c(\mathbf{x}_m, \boldsymbol{\theta}^c)$, where y_m are observations. The signal s_m is assumed to be distorted by some noise e_m with variance λ_0 .

Assume that the non-canonical representation of the model and the canonical representation of the model has the same range. First, find the parameter $\hat{\boldsymbol{\theta}}_N^i$ and $\hat{\boldsymbol{\theta}}_N^c$ that minimizes the loss function given in (4.1b) and (2.4), respectively. After that, calculate the covariance of those parameter estimates. By propagating this parameter covariance, it is possible to obtain an estimate of the variance of the estimated signal $\hat{s}_m$. For the canonical model, it is given by

$$\text{Var}(\hat{s}_m) \approx \frac{1}{N} \psi(\mathbf{x}_m, \boldsymbol{\theta}^{c0}) P_{\boldsymbol{\theta}^c}(\boldsymbol{\theta}^{c0}) \psi^\top(\mathbf{x}_m, \boldsymbol{\theta}^{c0}), \quad (5.2a)$$

with $\psi^\top(\mathbf{x}_m, \boldsymbol{\theta}^{c0})$ defined in (4.3), and where the covariance of the parameters is a $d_c \times d_c$ matrix given by

$$P_{\boldsymbol{\theta}^c}(\boldsymbol{\theta}^{c0}) = \lambda_0 \mathcal{I}_{\boldsymbol{\theta}^c}^{-1}, \quad (5.2b)$$

$$\mathcal{I}_{\boldsymbol{\theta}^c} = \bar{E}[\psi^\top(\mathbf{x}_n, \boldsymbol{\theta}^{c0}) \psi(\mathbf{x}_n, \boldsymbol{\theta}^{c0})], \quad (5.2c)$$

Here, $\psi(\mathbf{x}, \boldsymbol{\theta}^{c0})$ is the derivative of the model with respect to the canonical parameters which is defined in (4.3). Assume that $\boldsymbol{\theta}^{i0}$ is one of the parameters of the non-canonical representation of the model which results in an identical prediction of the signal as $\boldsymbol{\theta}^{c0}$ would for the canonical representation of the model. Furthermore, assume that there exists a transformation T_i between the canonical and non-canonical parameters in the form $\boldsymbol{\theta}^c = T_i(\boldsymbol{\theta}^i)$. Also remember that by the definition of the canonical model, the output values of the canonical and non-canonical model coincides for all values of $\mathbf{x}_m$. Denote the derivative of the non-canonical representation of the model with respect to its parameters as

$$f'_\theta(\mathbf{x}_m, \boldsymbol{\theta}^{i0}) \triangleq \frac{\partial}{\partial \boldsymbol{\theta}} f(\mathbf{x}_m, \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{i0}}. \quad (5.3)$$

Then, using the chain rule and the definition of the canonical form and the mapping between the non-canonical model and the canonical model, $T_i(\boldsymbol{\theta}^i)$, one can show that

$$f'_\theta(\mathbf{x}_m, \boldsymbol{\theta}^{i0}) \triangleq \frac{\partial}{\partial \boldsymbol{\theta}^i} f(\mathbf{x}_m, \boldsymbol{\theta}^i)|_{\boldsymbol{\theta}^i=\boldsymbol{\theta}^{i0}} = \frac{\partial}{\partial \boldsymbol{\theta}^i} f_c(\mathbf{x}_m, T_i(\boldsymbol{\theta}^i))|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \quad (5.4a)$$

$$= \frac{\partial}{\partial \boldsymbol{\theta}^c} f_c(\mathbf{x}_m, \boldsymbol{\theta}^c)|_{\boldsymbol{\theta}^c=\boldsymbol{\theta}^{c0}} \frac{\partial \boldsymbol{\theta}^c}{\partial \boldsymbol{\theta}^i}|_{\boldsymbol{\theta}^i=\boldsymbol{\theta}^{i0}} \quad (5.4b)$$

$$= \frac{\partial}{\partial \boldsymbol{\theta}^c} f_c(\mathbf{x}_m, \boldsymbol{\theta}^c)|_{\boldsymbol{\theta}^c=\boldsymbol{\theta}^{c0}} \frac{\partial}{\partial \boldsymbol{\theta}^i} T_i(\boldsymbol{\theta}^i)|_{\boldsymbol{\theta}^i=\boldsymbol{\theta}^{i0}} \quad (5.4c)$$

$$= \psi(\mathbf{x}_m, \boldsymbol{\theta}^{c0}) T'_{i,\boldsymbol{\theta}}(\boldsymbol{\theta}^{i0}), \quad (5.4d)$$

where

$$T'_{i,\theta}(\theta^{i0}) \triangleq \left. \frac{\partial T_i(\theta^i)}{\partial \theta^i} \right|_{\theta^i = \theta^{i0}}, \quad (5.5)$$

is a $d_c \times d$ matrix.

The goal in this section is to show that the prediction error variance for the signal $\hat{s}_m$ obtained by propagating the covariance of the parameter will be identical for the canonical and non-canonical representations of the model. That is, for some $d \times d$ matrix $P_\theta(\theta^{i0})$ it holds that

$$\text{Var}(\hat{s}_m) \approx \frac{1}{N} f'_\theta(\mathbf{x}_m, \theta^{i0}) P_\theta(\theta^{i0}) (f'_\theta(\mathbf{x}_m, \theta^{i0}))^\top, \quad (5.6)$$

is the same as (5.2a).

5.2.1 Linear regression

Consider a model that is linear in its parameters

$$f(\mathbf{x}_m, \theta^i) = \varphi_m^\top \theta^i, \quad \varphi_m^\top = g(\mathbf{x}_m), \quad (5.7)$$

with the associated canonical form

$$f_c(\mathbf{x}_m, \theta^c) = \varphi_{c,m}^\top \theta^c, \quad \varphi_{c,m}^\top = g_c(\mathbf{x}_m), \quad (5.8)$$

where g and g_c are linear or nonlinear basis functions of the state $\mathbf{x}_m$ used to construct the regressors $\varphi_m^\top$ and $\varphi_{c,m}^\top$ of dimension d and d_c , respectively. Recall that the range of the model structure for the canonical form and the non-canonical form are equal, i.e., $f(\mathbf{x}_m, \theta)$ might not be uniquely determined and θ might include elements that are linear combinations of each other, i.e., redundant parameters. The transformation between the canonical form and non-canonical form is given by

$$\theta^c = T_i(\theta) = T\theta^i, \quad (5.9a)$$

and hence

$$T'_{i,\theta}(\theta^{i0}) = T, \quad (5.9b)$$

where $T \in \mathbb{R}^{d_c \times d}$, $d \geq d_c$ is a matrix with $\text{rank}(T) = d_c$, i.e., T is a long matrix that has full row rank. Then

$$f_c(\mathbf{x}_m, \theta^c) = f_c(\mathbf{x}_m, T_i(\theta)) = \varphi_{c,m}^\top T\theta^i. \quad (5.10)$$

Having access to N data points $\mathbf{x}_{1:N} = (\mathbf{x}_n)_{n=1}^N$, one can write (5.7) and (5.8) as

$$f(\mathbf{x}_{1:N}, \theta^i) = \Phi_N^\top \theta^i, \quad (5.11a)$$

$$f_c(\mathbf{x}_{1:N}, \theta^c) = \varphi_{c,N}^\top \theta^c, \quad (5.11b)$$

where

$$\Phi_N^\top = \begin{pmatrix} \boldsymbol{\varphi}_1^\top & \cdots & \boldsymbol{\varphi}_N^\top \end{pmatrix}^\top, \quad (5.11c)$$

$$\Phi_{c,N}^\top = \begin{pmatrix} \boldsymbol{\varphi}_{c,1}^\top & \cdots & \boldsymbol{\varphi}_{c,N}^\top \end{pmatrix}^\top, \quad (5.11d)$$

where $\Phi_N^\top$ are a $N \times d$ matrix and $\Phi_{c,N}^\top$ are a $N \times d_c$ matrix with $\text{rank}(\Phi_{c,N}) = \text{rank}(\Phi_N) = d_c$. From the convex minimization problem for the canonical representation

$$\hat{\boldsymbol{\theta}}_N^c = \arg \min_{\boldsymbol{\theta}^c} V_{N,c}(\boldsymbol{\theta}^c), \quad (5.12a)$$

$$V_{N,c}(\boldsymbol{\theta}^c) = \|\mathbf{y}_N - \Phi_{c,N}^\top \boldsymbol{\theta}^c\|^2, \quad (5.12b)$$

and the non-canonical representation

$$\hat{\boldsymbol{\theta}}_N^i = \arg \min_{\boldsymbol{\theta}} V_N(\boldsymbol{\theta}), \quad (5.13a)$$

$$V_N(\boldsymbol{\theta}^i) = \|\mathbf{y}_N - \Phi_N^\top \boldsymbol{\theta}^i\|^2, \quad (5.13b)$$

where

$$\mathbf{y}_N = \begin{pmatrix} y_1^\top & \cdots & y_N^\top \end{pmatrix}^\top, \quad (5.14)$$

one can find the parameters $\hat{\boldsymbol{\theta}}_N^c$ and $\hat{\boldsymbol{\theta}}_N^i$ that are minimizing (5.12b) and (5.13b), respectively. Since $\Phi_{c,N}$ has full rank, the parameter estimate is unique and given by

$$\hat{\boldsymbol{\theta}}_N^c = \left(\Phi_{c,N} \Phi_{c,N}^\top \right)^{-1} \Phi_{c,N} \mathbf{y}_N. \quad (5.15a)$$

This is called the LLSS and can be found from requiring the derivate of (5.13b) with respect to $\boldsymbol{\theta}^c$ to be zero, i.e., from the first-order necessary and sufficient condition of optimality. The parameter covariance is

$$\text{Cov}(\hat{\boldsymbol{\theta}}_N^c) = \lambda_0 \mathcal{I}_{\boldsymbol{\theta}^c}^{-1}, \quad (5.15b)$$

$$\mathcal{I}_{\boldsymbol{\theta}^c} = \Phi_{c,N} \Phi_{c,N}^\top, \quad (5.15c)$$

and the estimated signal is given by

$$\hat{s}_m = \boldsymbol{\varphi}_{c,m}^\top \hat{\boldsymbol{\theta}}_N^c. \quad (5.15d)$$

If the prediction error is small, the variance of the parameters can be propagated to the prediction error variance of the estimated signal

$$\text{Var}(\hat{s}_m) = \lambda_0 \boldsymbol{\varphi}_{c,m}^\top \left(\Phi_{c,N} \Phi_{c,N}^\top \right)^{-1} \boldsymbol{\varphi}_{c,m}. \quad (5.16)$$

For (5.13a), an optimal value of the parameters is given by the solution to

$$\Phi_N \Phi_N^\top \boldsymbol{\theta}^i = \Phi_N \mathbf{y}_N, \quad (5.17)$$

that is found by the first order optimality condition, i.e., setting the derivative of (5.12b) with respect to θ^i to zero. Assuming $\Phi_N \Phi_N^\top$ has full rank, using the SVD

$$\Phi_N^\top = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^\top = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^\top = U_1 \Sigma V^\top, \quad (5.18)$$

the solution of (5.17) can be expressed as

$$\hat{\theta}_N^i = (\Phi_N \Phi_N^\top)^{-1} \Phi_N \mathbf{y}_N \quad (5.19a)$$

$$= (V \Sigma^2 V^\top)^{-1} V \Sigma U_1^\top \mathbf{y}_N \quad (5.19b)$$

$$= V \Sigma^{-1} U_1^\top \mathbf{y}_N \quad (5.19c)$$

$$= \sum_{j=1}^d \frac{\mathbf{u}_j^\top \mathbf{y}_N}{\sigma_j} \mathbf{v}_j, \quad (5.19d)$$

where σ_j is the j th singular value and $\mathbf{u}_j$ and $\mathbf{v}_j$ denotes the j th column of U and V , respectively. The non-canonical model is in general underdetermined, hence there exists no unique solution and some of the singular values are equal to zero. Then as shown in [Nocedal and Wright, 2006] the solution can be written as²

$$\hat{\theta}_N^i = \sum_{j=1}^{d_c} \frac{\mathbf{u}_j^\top \mathbf{y}_N}{\sigma_j} \mathbf{v}_j + \sum_{j=d_c+1}^d \tau_j \mathbf{v}_j, \quad (5.20)$$

for arbitrary coefficients τ_j . Note that $\mathbf{v}_j, j = d_c+1, \dots, d$, is a basis for the nullspace of the Hessian of the cost function, i.e., the nullspace of $\Phi_N \Phi_N^\top$. One natural choice of τ_j is the one that gives the minimum norm solution, i.e., $\tau_j = 0, \forall j$. This solution can also be found using the Moore–Penrose inverse, which by design neglects singular values that are zero, and hence work for rank deficient matrices. For a matrix A that has full row rank, the Moore–Penrose inverse is written as

$$A^+ = A^\top (A A^\top)^{-1}. \quad (5.21)$$

Then the *minimum norm solution* can be constructed such as

$$\hat{\theta}_N^i = (\Phi_N \Phi_N^\top)^+ \Phi_N \mathbf{y}_N, \quad (5.22a)$$

with the estimate of the signal given by

$$\hat{s}_m = \boldsymbol{\varphi}_m^\top \hat{\theta}_N^i. \quad (5.22b)$$

Define the covariance of the non-canonical parameters as

$$\text{Cov}(\hat{\theta}_N^i) \triangleq (\Phi_N \Phi_N^\top)^+ \Phi_N \text{Cov}(\mathbf{y}_N) \Phi_N^\top (\Phi_N \Phi_N^\top)^+ \quad (5.23a)$$

$$= \left[\text{Cov}(\mathbf{y}_N) = \lambda_0 I \right] = \lambda_0 (\Phi_N \Phi_N^\top)^+, \quad (5.23b)$$

²It is assumed that the singular values are sorted in ascending order.

and the information matrix as

$$\mathcal{I}_\theta = \Phi_N \Phi_N^\top. \quad (5.23c)$$

Where it has been used that the Moore-Penrose inverse is defined such that $AA^+ = AA^\top(AA^\top)^{-1} = I$. If the prediction error is small, the parameter uncertainty can be propagated to the variance in the output such as

$$\text{Var}(\hat{s}_m) = \varphi_m^\top \text{Cov}(\hat{\theta}_N^i) \varphi_m. \quad (5.24)$$

Then, one can show the result that the prediction error variance for (5.16) and (5.24) are identical if the range of the canonical and non-canonical representations of the model coincide.

Theorem 5.1. *Given the problem specified in (5.12) and (5.13), i.e., a canonical and a non-canonical representation of the linear model, which both are linear in the parameters, are assumed to approximate some given observations. Assume that both of them are expressive enough to include the true model in the model set, $\mathcal{S} \in \mathcal{M}^*$, as well as the range of the model structures are the same, and that the data is informative enough. As a consequence, $\text{rank}(\mathcal{I}_{\theta^c}) = d_c$ and $\text{rank}(\mathcal{I}_\theta) = d_c$. Then the estimated signals from the canonical and non-canonical representation of the model coincide, i.e.,*

$$\varphi_m^\top \hat{\theta}_N^i = \varphi_{c,m}^\top \hat{\theta}_N^c, \quad (5.25a)$$

where the parameters are estimated according to (5.22a) and (5.15a), respectively. Furthermore, the prediction error variances are identical, i.e.,

$$\varphi_m^\top \text{Cov}(\hat{\theta}_N^i) \varphi_m = \varphi_{c,m}^\top \text{Cov}(\hat{\theta}_N^c) \varphi_{c,m}, \quad (5.25b)$$

where the parameter covariances are calculated according to (5.23a) and (5.15b) respectively.

Proof: First, let us state some properties of the Moore-Penrose inverse. If A has full column rank and B has full row rank, then

$$(AB)^+ = B^+ A^+. \quad (5.26a)$$

If C is invertible, i.e., a quadratic matrix with full column rank and full row rank, then

$$(ACB)^+ = B^+ C^+ A^+. \quad (5.26b)$$

Moreover, notice that

$$\varphi_m^\top = \varphi_{c,m}^\top T \Rightarrow \varphi_{c,m}^\top = \varphi_m^\top T^+, \quad (5.27a)$$

and

$$\Phi_N^\top = \Phi_{c,N}^\top T \Rightarrow \Phi_{c,N}^\top = \Phi_N^\top T^+, \quad (5.27b)$$

holds, where T has full row rank. Hence $T^\top$ has full column rank, T^+ full column rank, and $(T^\top)^+$ full row rank. The minimum norm solution to the non-canonical model can be written as

$$\hat{\theta}_N^i = (\Phi_N \Phi_N^\top)^+ \Phi_N \mathbf{y}_N \quad (5.28a)$$

$$= (T^\top \Phi_{c,N} \Phi_{c,N}^\top T)^+ T^\top \Phi_{c,N} \mathbf{y}_N \quad (5.28b)$$

$$= \left[T \text{ has full row rank, and } \Phi_{c,N} \Phi_{c,N}^\top \text{ invertible.} \right] \quad (5.28c)$$

$$= T^+ (\Phi_{c,N} \Phi_{c,N}^\top)^+ (T^\top)^+ T^\top \Phi_{c,N} \mathbf{y}_N \quad (5.28d)$$

$$= T^+ (\Phi_{c,N} \Phi_{c,N}^\top)^{-1} (T^+)^+ T^\top \Phi_{c,N} \mathbf{y}_N \quad (5.28e)$$

$$= T^+ (\Phi_{c,N} \Phi_{c,N}^\top)^{-1} (T T^\top)^{-1} T T^\top \Phi_{c,N} \mathbf{y}_N \quad (5.28f)$$

$$= T^+ (\Phi_{c,N} \Phi_{c,N}^\top)^{-1} \Phi_{c,N} \mathbf{y}_N \quad (5.28g)$$

$$= T^+ \hat{\theta}_N^c, \quad (5.28h)$$

with the signal estimated given as

$$\hat{s}_m = \boldsymbol{\varphi}_m^\top \hat{\theta}_N^i = \boldsymbol{\varphi}_{c,m}^\top T T^+ \hat{\theta}_N^c = \boldsymbol{\varphi}_{c,m}^\top T T^\top (T T^\top)^{-1} \hat{\theta}_N^c = \boldsymbol{\varphi}_{c,m}^\top \hat{\theta}_N^c. \quad (5.29)$$

Hence the estimated signal is independent of if the canonical or non-canonical model is used. Now it is only left to show that the prediction error variance is the same as well.

Using the properties of the transformation T , the parameter covariance for the non-canonical model in (5.23a) can be rewritten as

$$\text{Cov}(\hat{\theta}_N^i) = \lambda_0 (\Phi_N \Phi_N^\top)^+ \quad (5.30a)$$

$$= \lambda_0 (T^\top \Phi_{c,N} \Phi_{c,N}^\top T)^+ \quad (5.30b)$$

$$= \left[T \text{ has full row rank, and } \Phi_{c,N} \Phi_{c,N}^\top \text{ invertible.} \right] \quad (5.30c)$$

$$= \lambda_0 T^+ (\Phi_{c,N} \Phi_{c,N}^\top)^{-1} (T^+)^+ \quad (5.30d)$$

$$= T^+ \text{Cov}(\hat{\theta}_N^c) (T^+)^+. \quad (5.30e)$$

Using the expression of the parameter covariance for the non-canonical model,

the propagated variance of the output prediction error can be rewritten as

$$\text{Var}(\hat{s}_m) = \boldsymbol{\varphi}_m^\top \text{Cov}(\hat{\boldsymbol{\theta}}_N^i) \boldsymbol{\varphi}_m \quad (5.31a)$$

$$\lambda_0 \boldsymbol{\varphi}_m^\top (\Phi_N \Phi_N^\top)^+ \boldsymbol{\varphi}_m \quad (5.31b)$$

$$= \lambda_0 \boldsymbol{\varphi}_{c,m}^\top T (T^\top \Phi_{c,N} \Phi_{c,N}^\top T)^+ T^\top \boldsymbol{\varphi}_{c,m} \quad (5.31c)$$

$$= \left[T \text{ has full row rank, and } \Phi_{c,N} \Phi_{c,N}^\top \text{ invertible.} \right] \quad (5.31d)$$

$$= \lambda_0 \boldsymbol{\varphi}_{c,m}^\top T T^+ (\Phi_{c,N} \Phi_{c,N}^\top)^{-1} (T^+)^T T^\top \boldsymbol{\varphi}_{c,m} \quad (5.31e)$$

$$= \lambda_0 \boldsymbol{\varphi}_{c,m}^\top T T^\top (T T^\top)^{-1} (\Phi_{c,N} \Phi_{c,N}^\top)^{-1} (T T^\top)^{-1} T T^\top \boldsymbol{\varphi}_{c,m} \quad (5.31f)$$

$$= \lambda_0 \boldsymbol{\varphi}_{c,m}^\top (\Phi_{c,N} \Phi_{c,N}^\top)^{-1} \boldsymbol{\varphi}_{c,m} \quad (5.31g)$$

$$= \boldsymbol{\varphi}_{c,m}^\top \text{Cov}(\hat{\boldsymbol{\theta}}_N^c) \boldsymbol{\varphi}_{c,m}, \quad (5.31h)$$

which coincides with the prediction error variance for the canonical model in (5.16). $\square$

One interpretation of this result is that the Moore-Penrose inverse includes information about the transformation T in the non-canonical model. This, will now be demonstrated with an example inspired by NN. The nonlinear basis functions used here resemble the last hidden layer in a two-layer NN, see (2.56).

Example 5.2

Consider the setup in Example 2.5, where the linear system of nonlinear basis functions is assumed to represent how the NTF depends on the wheel slip in a car. Given the canonical representation of the model

$$\boldsymbol{\varphi}_{c,m} = \begin{pmatrix} \sigma(-40x_m + 0.0061) & \sigma(-6.8x_m + 0.0036) & 1 \end{pmatrix}^\top, \quad (5.32)$$

with the transformation

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & -1 \\ 1 & -1 & 3 & 0 & 2 \end{pmatrix}, \quad (5.33)$$

to the non-canonical representation

$$\boldsymbol{\varphi}_m = \begin{pmatrix} \sigma(-40x_m + 0.0061) - \sigma(-6.8x_m + 0.0036) + 1 \\ -\sigma(-6.8x_m + 0.0036) - 1 \\ 3 \\ \sigma(-6.8x_m + 0.0036) \\ -\sigma(-6.8x_m + 0.0036) - 2 \end{pmatrix} \quad (5.34)$$

of the model, simulate 200 data points from

$$y_m = \boldsymbol{\varphi}_{c,m}^\top \begin{pmatrix} 1.43 \\ -0.60 \\ -0.46 \end{pmatrix} + e_m. \quad (5.35)$$

Where e_m is i.i.d. noise with zero mean and variance, $\lambda_0 = 10^{-2}$, that gives an SNR on 20 dB, and $\{x_m\}_{m=1}^{200} \in [-0.6, 0.6]$. Using the cost function in (5.12a) and (5.13a), a canonical model and a non-canonical model is fit to the data using the least squares solution and the minimum norm solution, respectively.

In Figure 5.2 the prediction and the prediction error variance for the canonical and non-canonical model can be seen. Here, the prediction error variance is calculated using (5.2a) and (5.6) for the canonical and non-canonical model, respectively. One can see that both the predictions and prediction error variances are the same for the canonical and non-canonical model that is linear in the parameters.

One can also notice that since the model is linear in the parameters, the Gaussian distribution of the output prediction follows from the Gaussian distribution of the noise.

Remark 5.1. If the observation noise e_m is Gaussian distributed with zero mean and variance λ_0 , $\hat{\theta}_N^c$ is the MLE of the parameters given the cost function in (5.12a). This gives us that the parameter covariance in (5.15b) also is the so-called CRLB of the parameters, since the least squares estimate obtain this lower bound when the observation noise is Gaussian distributed. Hence, even though the transformation T is unknown, the estimator of the prediction error for the non-canonical model will have minimum variance.

5.2.2 Nonlinear regression

Now assume that the canonical and non-canonical representations of the models is nonlinear, but that the range of their model structures still are equal, i.e., that the expressiveness of the model structures is the same. Furthermore, assume that the prediction error is small such that the prediction error variances are given in (5.2a) and (5.6), respectively. Define the $d \times d$ information matrix for the non-canonical model as

$$\mathcal{I}_\theta \triangleq \bar{E}[(f'_\theta(\mathbf{x}_n, \theta^{i0}))^\top f'_\theta(\mathbf{x}_n, \theta^{i0})], \quad (5.36a)$$

and covariance matrix of the asymptotic distribution

$$P_\theta(\theta^{i0}) = \lambda_0 \mathcal{I}_\theta^+. \quad (5.36b)$$

The inverse of the covariance matrix of the canonical representation of the model has been replaced by the Moore-Penrose inverse. This because the information matrix of the non-canonical model might be singular. The Moore-Penrose inverse is still well-defined even though there exist linear combinations of parameters with or without low excitation in $\mathcal{I}_\theta$, i.e., with zero singular values. This is to prevent infinite variance in (5.6), when the parameter uncertainty is propagated to uncertainty in the output. Then again, one can show that the prediction and the prediction error variance is the same for the canonical and the non-canonical model.

Figure 5.2: Prediction and prediction error variance for a linear model using both a canonical and non-canonical representation of a model. The parameters are estimated using (5.15a) and (5.22a) for the canonical and non-canonical representation, respectively. The data in the experiment is generated from a model describing how the NTF depends on the wheel slip of a car.

Theorem 5.2. *Given the problem specified in (4.2) and (2.3). Assume that the canonical and non-canonical representation of the nonlinear model that both includes the true model in their model sets, $\mathcal{S} \in \mathcal{M}^*$, that their respective model structures have the same range, and that the data is informative enough. As a consequence, $\text{rank}(\mathcal{I}_{\theta^c}) = d_c$ and $\text{rank}(\mathcal{I}_\theta) = d_c$, where the information matrices are given by (4.16b) and (5.36a). Then, their predictions are identical, i.e.,*

$$f_c(\mathbf{x}_m, \hat{\boldsymbol{\theta}}_N^c) = f(\mathbf{x}_m, \hat{\boldsymbol{\theta}}_N^i). \quad (5.37a)$$

Furthermore, if the prediction error is small, the prediction error variance obtained by the first order Taylor expansion coincides for the canonical model and the non-canonical representation, i.e.,

$$\frac{1}{N} f'_\theta(\mathbf{x}_m, \boldsymbol{\theta}^{i0}) P_\theta(\boldsymbol{\theta}^{i0}) (f'_\theta(\mathbf{x}_m, \boldsymbol{\theta}^{i0}))^\top = \frac{1}{N} \psi(\mathbf{x}_m, \boldsymbol{\theta}^{c0}) P_{\theta^c}(\boldsymbol{\theta}^{c0}) \psi^\top(\mathbf{x}_m, \boldsymbol{\theta}^{c0}), \quad (5.37b)$$

where the information matrix for the parameters for the non-canonical model are given by (5.36a).

Proof: As a first observation, by the definition of the canonical form, the prediction from the canonical and non-canonical representations of the model are the same, i.e.,

$$\hat{s}_m = f_c(\mathbf{x}_m, \hat{\boldsymbol{\theta}}_N^c) = f_c(\mathbf{x}_m, T_i(\hat{\boldsymbol{\theta}}_N^i)) = f(\mathbf{x}_m, \hat{\boldsymbol{\theta}}_N^i). \quad (5.38)$$

From the assumption that $f'_\theta(\mathbf{x}_{1:N}, \boldsymbol{\theta}^{i0})$ and $\psi(\mathbf{x}_{1:N}, \boldsymbol{\theta}^{c0})$ span the same space $T'_{i,\theta}(\boldsymbol{\theta}^{i0})$ has full row rank, i.e., $\text{rank}(T'_{i,\theta}(\boldsymbol{\theta}^{i0})) = d_c$. Hence the result about the Moore-Penrose inverse in (5.26) can be used. Note that the definition of $T_i(\boldsymbol{\theta}^i)$ has been extended compared to (5.9), where it was a linear transformation of the parameters. Here $T_i(\boldsymbol{\theta}^i)$ could be any transformation of the parameters as long as $\text{rank}(T'_{i,\theta}(\boldsymbol{\theta}^{i0})) = d_c$.

Then expand $\mathcal{I}_\theta$ using the definition of the canonical model such as

$$\mathcal{I}_\theta = \bar{E} \left[T'_{i,\theta}(\boldsymbol{\theta}^{i0})^\top \psi^\top(\mathbf{x}_n, \boldsymbol{\theta}^{c0}) \psi(\mathbf{x}_n, \boldsymbol{\theta}^{c0}) T'_{i,\theta}(\boldsymbol{\theta}^{i0}) \right] \quad (5.39a)$$

$$= \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=1}^N E \left[T'_{i,\theta}(\boldsymbol{\theta}^{i0})^\top \psi^\top(\mathbf{x}_n, \boldsymbol{\theta}^{c0}) \psi(\mathbf{x}_n, \boldsymbol{\theta}^{c0}) T'_{i,\theta}(\boldsymbol{\theta}^{i0}) \right] \quad (5.39b)$$

$$= T'_{i,\theta}(\boldsymbol{\theta}^{i0})^\top \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=1}^N E \left[\psi^\top(\mathbf{x}_n, \boldsymbol{\theta}^{c0}) \psi(\mathbf{x}_n, \boldsymbol{\theta}^{c0}) \right] T'_{i,\theta}(\boldsymbol{\theta}^{i0}) \quad (5.39c)$$

$$= T'_{i,\theta}(\boldsymbol{\theta}^{i0})^\top \bar{E}[\psi^\top(\mathbf{x}_n, \boldsymbol{\theta}^{c0}) \psi(\mathbf{x}_n, \boldsymbol{\theta}^{c0})] T'_{i,\theta}(\boldsymbol{\theta}^{i0}) \quad (5.39d)$$

$$= T'_{i,\theta}(\boldsymbol{\theta}^{i0})^\top \mathcal{I}_{\theta^c} T'_{i,\theta}(\boldsymbol{\theta}^{i0}). \quad (5.39e)$$

where $T'_{i,\theta}(\boldsymbol{\theta}^{i0})$ can be moved outside of the expectation operator since it does not depend on the number of samples N . Here $\mathcal{I}_{\theta^c}$ is the $d_c \times d_c$ information matrix for the canonical representation of the parameters, which is defined in (4.16b).

Using the expansion of $\mathcal{I}_\theta$, the prediction error variance for the non-canonical model can be rewritten as follows:

$$\text{Var}(\hat{s}_m) = \frac{1}{N} f'_\theta(\mathbf{x}_m, \theta^{i0}) P_\theta(\theta^{i0}) (f'_\theta(\mathbf{x}_m, \theta^{i0}))^\top \quad (5.40a)$$

$$= \frac{\lambda_0}{N} f'_\theta(\mathbf{x}_m, \theta^{i0}) \mathcal{I}_\theta^+ (f'_\theta(\mathbf{x}_m, \theta^{i0}))^\top \quad (5.40b)$$

$$= \frac{\lambda_0}{N} f'_\theta(\mathbf{x}_m, \theta^{i0}) (T'_{i,\theta}(\theta^{i0})^\top \mathcal{I}_{\theta^c} T'_{i,\theta}(\theta^{i0}))^+ (f'_\theta(\mathbf{x}_m, \theta^{i0}))^\top \quad (5.40c)$$

$$= \left[T'_{i,\theta}(\theta^{i0}) \text{ has full row rank, and } \mathcal{I}_{\theta^c} \text{ invertible} \right] \quad (5.40d)$$

$$= \frac{\lambda_0}{N} f'_\theta(\mathbf{x}_m, \theta^{i0}) T'_{i,\theta}(\theta^{i0})^+ \mathcal{I}_{\theta^c}^{-1} (T'_{i,\theta}(\theta^{i0})^\top)^+ (f'_\theta(\mathbf{x}_m, \theta^{i0}))^\top \quad (5.40e)$$

$$= \frac{\lambda_0}{N} \psi(\mathbf{x}_m, \theta^{c0}) T'_{i,\theta}(\theta^{i0}) T'_{i,\theta}(\theta^{i0})^+ \mathcal{I}_{\theta^c}^{-1} (T'_i(\theta^{i0})^\top)^+ T'_i(\theta^{i0})^\top \psi^\top(\mathbf{x}_m, \theta^{c0}) \quad (5.40f)$$

$$= \left[T'_{i,\theta}(\theta^{i0}) T'_{i,\theta}(\theta^{i0})^+ = \mathbf{I}_{d_c \times d_c} \right] \quad (5.40g)$$

$$= \frac{\lambda_0}{N} \psi(\mathbf{x}_m, \theta^{c0}) \mathcal{I}_{\theta^c}^{-1} \psi^\top(\mathbf{x}_m, \theta^{c0}) \quad (5.40h)$$

$$= \frac{1}{N} \psi(\mathbf{x}_m, \theta^{c0}) P_{\theta^c}(\theta^{c0}) \psi^\top(\mathbf{x}_m, \theta^{c0}). \quad (5.40i)$$

Hence the prediction error variances for the canonical and non-canonical models coincide. $\square$

This strengthens the conclusion from Section 5.2.1, namely that for the prediction error variance, information about the canonical representation of a model is not necessary. One can rather use a non-canonical representation, which is the representation one does usually have. Also notice that in the same sense as the parameter covariance of the canonical parameters can be approximated by (5.41), the covariance of the non-canonical model can be approximated as

$$\text{Cov}(\hat{\theta}_N^i) \sim \frac{\hat{P}_N}{N}, \quad (5.41)$$

where

$$\hat{P}_N = \hat{\lambda}_N \left(\frac{1}{N} \sum_{n=1}^N (f'_\theta(\mathbf{x}_n, \hat{\theta}_N^i))^\top f'_\theta(\mathbf{x}_n, \hat{\theta}_N^i) \right)^+, \quad (5.42a)$$

$$\hat{\lambda}_N = \frac{1}{N} \sum_{n=1}^N \varepsilon^2(\mathbf{x}_n, \hat{\theta}_N^c), \quad (5.42b)$$

Hence, having processed N observations the prediction error variance can be approximated by

$$\text{Var}(\hat{s}_m) \approx f'_\theta(\mathbf{x}_m, \hat{\theta}_N^i) \text{Cov}(\hat{\theta}_N^i) (f'_\theta(\mathbf{x}_m, \hat{\theta}_N^i))^\top. \quad (5.43)$$

To illustrate this, consider an example where the nonlinear function is inspired by an NN.

Example 5.3

Consider the same nonlinear system as in Example 2.2 that describes how the normal NTF depends on the wheel slip in a car, which is further described in Section 6.1. Assume that this nonlinear system has a canonical representation given by

$$f_c(x_m, \theta^c) = \theta_1^c \sigma(\theta_2^c x_m + \theta_3^c) + \theta_4^c \sigma(\theta_5^c x_m + \theta_6^c) + \theta_7^c, \quad (5.44)$$

with a non-canonical representation given by

$$f(x_m, \theta) = \theta_1 \sigma(\theta_2(x_m + 1) + \theta_3(2 - x_m) + 5\theta_4) + \theta_5 \sigma(\theta_6^c x_m + \theta_7^c) + \theta_8^c, \quad (5.45)$$

Simulate $m = 1, \dots, 201$ data points from the signal model with the same structure as the canonical representation with parameters

$$\theta^{c0} = \begin{pmatrix} 40.9273 & 6.6459 & -0.0067 & -0.0044 & 1.4294 & -0.6016 & -0.4252 \end{pmatrix}^\top, \quad (5.46a)$$

where the signal is distorted by noise, i.e.,

$$y_m = f_c(x_m, \theta^{c0}) + e_m, \quad (5.46b)$$

where e_m is i.i.d. noise with zero mean and variance, $\lambda_0 = 10^{-2}$, that is an SNR of 20 dB, and $\{\mathbf{x}_m\}_{m=1}^{200} \in [-0.6, 0.6]$. Using a nonlinear least squares solver, it is possible to find the parameters that minimizes (4.1b) and (2.4) for the canonical and non-canonical model, respectively.

In Figure 5.3 the prediction and the prediction error variance for the canonical and non-canonical model can be seen. Once again, the predictions and the prediction error variances are the same for both the canonical and non-canonical representations of the model, hence one can conclude that the information about the transformation $T_i(\theta)$ is unnecessary if one is interested in the output prediction and output prediction error variance.

In Chapter 4, the result that the prediction error is asymptotically Gaussian distributed has been shown. Combining this with Theorem 5.2, which gives us that the prediction and prediction error variance is the same for the canonical and non-canonical model, one can conclude that the prediction error for the non-canonical model also is Gaussian distributed similarly.

5.3 Models with more flexibility

A model can often represent data generated from a model with lower model order perfectly, e.g., a polynomial of degree $n + 1$ can represent data generated from a

(a) Prediction.

(b) Prediction error variance.

Figure 5.3: Prediction and prediction error variance for a nonlinear model using both a canonical and non-canonical representation. The parameter covariance for the canonical and non-canonical representation of the model is calculated using (4.25b) and (5.41), respectively. The data in the experiment is generated from a model describing how the NTF depends on the wheel slip of a car.

polynomial of degree n perfectly. This is sometimes referred to as a *nested model*, since the smaller representation of the model can be seen as a special case of the larger representation of the model with more parameters, see [Gustafsson et al., 2011]. That is, the model est for the canonical model representation of the model is contained in the model set of the non-canonical representation of the model, i.e., $\mathcal{M}_c^* \subset \mathcal{M}_i^*$. This can be thought of as that the model structure of the canonical model set can be obtained by setting some parameters in the non-canonical model structure to zero.

5.3.1 Linear regression

For models that are linear in the parameters, the variance of the signal for the non-canonical representation of the model is overestimated. This is a consequence of the notion that the basis function of a smaller model is included in the basis functions for the larger model. This can be illustrated with an example, using NN inspired nonlinear basis functions.

Example 5.4

Consider the setup in Example 5.2 where nonlinear basis functions inspired from NN are used, and where the measured signal is given by (5.35). Instead of using a non-canonical model with the same range as the canonical, a more flexible model is used which has larger range, i.e.,

$$\boldsymbol{\varphi}_m^\top = (\boldsymbol{\varphi}_{c,m}^\top \quad \boldsymbol{\varphi}_{o_j,m}^\top), \quad (5.47a)$$

$$\boldsymbol{\varphi}_{o_j,m}^\top = (\sigma(W_{o_j}x_m + b_{o_j}) \quad \boldsymbol{\varphi}_{o_{j-1},m}^\top), \quad o_j = 1, \dots, 3, \quad (5.47b)$$

where

$$W_{o_j} = b_{o_j} = o_j. \quad (5.47c)$$

In Figure 5.4 the prediction and the prediction error variance for the overparameterized models and the canonical (minimal) model are plotted. For the overparameterized models, two to four basis functions were added. One can see that the prediction error variance for the overparameterized models is bounded from below by the canonical representation of the model.

This result can formulate as a theorem.

Theorem 5.3. *Given the problem specified in (5.12) and (5.13). Assume that the linear model has a canonical representation of degree d_c where $\boldsymbol{\theta}^c \in \mathbb{R}^{d_c}$ and an overparameterized model where $\boldsymbol{\theta} \in \mathbb{R}^d$ where $d_c < d$. Also assume that the model set of the canonical representation is contained in the model set of the overparameterized model, i.e., $\mathcal{M}_c^* \subset \mathcal{M}_o^*$, where $\mathcal{M}_o^*$ is the model set for the overparameterized model.*

If $\text{rank}(\mathcal{I}_{\boldsymbol{\theta}^c}) = d_c$ and $\text{rank}(\mathcal{I}_{\boldsymbol{\theta}}) = d$, the parameter, the parameter covariance, and the prediction error variance are then given by (5.15a), (5.15b), and (5.16)

Figure 5.4: Prediction and prediction error variance for a set of linear models using both a canonical and non-canonical representation of the models. Here the non-canonical representation higher model order compared to the canonical representation, see (5.47). The parameters are estimates as the LLSS.

for both the canonical and overparameterized model, respectively. Then the overparameterized model has equal or higher prediction error variance compared to the canonical representation of the model at all points, i.e.,

$$\lambda_0 \boldsymbol{\varphi}_{c,m}^\top (\Phi_{c,N} \Phi_{c,N}^\top)^{-1} \boldsymbol{\varphi}_{c,m} \leq \lambda_0 \boldsymbol{\varphi}_m^\top (\Phi_N \Phi_N^\top)^{-1} \boldsymbol{\varphi}_m, \quad \forall m. \quad (5.48)$$

Proof: Without loss of generality, the overparameterized model can be written in terms of the canonical model such as

$$\boldsymbol{\varphi}_m^\top = (\boldsymbol{\varphi}_{c,m}^\top \quad \boldsymbol{\varphi}_{o,m}^\top), \quad (5.49a)$$

$$\Phi_N^\top = (\Phi_{c,N}^\top \quad \Phi_{o,N}^\top), \quad (5.49b)$$

$$\Phi_N \Phi_N^\top = \begin{pmatrix} \Phi_{c,N} \Phi_{c,N}^\top & \Phi_{c,N} \Phi_{o,N}^\top \\ \Phi_{o,N} \Phi_{c,N}^\top & \Phi_{o,N} \Phi_{o,N}^\top \end{pmatrix}, \quad (5.49c)$$

where $\boldsymbol{\varphi}_{o,m}^\top$ and $\Phi_{o,N}^\top$ correspond to the parameters added in the overparameterized model. Using the block-wise inverse

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{pmatrix}, \quad (5.50)$$

the parameter covariance of the overparameterized model, $\lambda_0 (\Phi_N \Phi_N^\top)^{-1}$ can be written as

$$\lambda_0 (\Phi_N \Phi_N^\top)^{-1} = \lambda_0 \begin{pmatrix} (\Phi_{c,N} \Phi_{c,N}^\top)^{-1} + R_2^\top R_1 R_2 & -R_2^\top R_1 \\ -R_1 R_2 & R_1 \end{pmatrix}^{-1}, \quad (5.51a)$$

where

$$R_1 = (\Phi_{o,N} \Phi_{o,N}^\top - \Phi_{c,N} \Phi_{o,N}^\top (\Phi_{c,N} \Phi_{c,N}^\top)^{-1} \Phi_{o,N} \Phi_{c,N}^\top)^{-1}, \quad (5.51b)$$

$$R_2 = \Phi_{c,N} \Phi_{o,N}^\top (\Phi_{c,N} \Phi_{c,N}^\top)^{-1}. \quad (5.51c)$$

Then the variance of the overparameterized model can be written as

$$\begin{aligned} & \lambda_0 \boldsymbol{\varphi}_m^\top (\Phi_N \Phi_N^\top)^{-1} \boldsymbol{\varphi}_m \\ &= \lambda_0 \begin{pmatrix} \boldsymbol{\varphi}_{c,m}^\top & \boldsymbol{\varphi}_{o,m}^\top \end{pmatrix} \begin{pmatrix} (\Phi_{c,N} \Phi_{c,N}^\top)^{-1} + R_2^\top R_1 R_2 & -R_2^\top R_1 \\ -R_1 R_2 & R_1 \end{pmatrix} \begin{pmatrix} \boldsymbol{\varphi}_{c,m} \\ \boldsymbol{\varphi}_{o,m} \end{pmatrix} \end{aligned} \quad (5.52a)$$

$$\begin{aligned} &= \lambda_0 \boldsymbol{\varphi}_{c,m}^\top (\Phi_{c,N} \Phi_{c,N}^\top)^{-1} \boldsymbol{\varphi}_{c,m} + \lambda_0 \boldsymbol{\varphi}_{c,m}^\top R_2^\top R_1 R_2 \boldsymbol{\varphi}_{c,m} \\ &\quad - \lambda_0 \boldsymbol{\varphi}_{c,m}^\top R_2^\top R_1 \boldsymbol{\varphi}_{o,m} - \lambda_0 \boldsymbol{\varphi}_{o,m}^\top R_1 R_2 \boldsymbol{\varphi}_{c,m} + \lambda_0 \boldsymbol{\varphi}_{o,m}^\top R_1 \boldsymbol{\varphi}_{o,m}. \end{aligned} \quad (5.52b)$$

If one minimizes the variance of the overparameterized model with respect to the regressor that corresponds to the extra parameters, $\boldsymbol{\varphi}_{o,m}$, one can find that the minimum is obtained at

$$\boldsymbol{\varphi}_{o,m} = R_2 \boldsymbol{\varphi}_{c,m}. \quad (5.53)$$

This implies that the minimum variance of the overparameterized model is

$$\lambda_0 \boldsymbol{\varphi}_{c,m}^\top (\Phi_{c,N} \Phi_{c,N}^\top)^{-1} \boldsymbol{\varphi}_{c,m}, \quad (5.54)$$

i.e., the variance of the canonical model. □

Remark 5.2. If the order of the basis functions $\boldsymbol{\varphi}_m$ would be different, i.e., the canonical basis functions $\boldsymbol{\varphi}_{c,m}$ might not come first, this can be solved by introducing an invertible permutation matrix P that sorts them in the correct order. Since the transformation acts on both $\boldsymbol{\varphi}_m$ and $[\Phi_N]_m$, when calculating the prediction error variance, the contribution of the transformation will be canceled. □

5.3.2 Nonlinear regression

If the observation noise is small, this is also true for a nonlinear model, and the prediction error variance is bounded by the variance of the less complex models. This can be illustrated with an example with NN as the nonlinear model.

Example 5.5

Consider the same nonlinear system as in Example 5.3 that describes how the normal NTF depends on the wheel slip in a car, which is further described in Section 6.1. Assume that the nonlinear system has a canonical representation given by a two-layer NN with two nodes in the hidden layer, i.e.,

$$f_c(x_m, \boldsymbol{\theta}^c) = \boldsymbol{\theta}_1^c \sigma(\boldsymbol{\theta}_2^c x_m + \boldsymbol{\theta}_3^c) + \boldsymbol{\theta}_4^c \sigma(\boldsymbol{\theta}_5^c x_m + \boldsymbol{\theta}_6^c) + \boldsymbol{\theta}_7^c, \quad (5.55)$$

which is denoted by $\text{NN}_{2,2}$. Then, consider the overparameterized models $\text{NN}_{2,l}$ with $l = 3, \dots, 6$ nodes in the hidden layer. Simulate 201 samples from the canonical representation of the model with the parameters given by

$$\boldsymbol{\theta}^{c0} = \begin{pmatrix} 40.9273 & 6.6459 & -0.0067 & -0.0044 & 1.4294 & -0.6016 & -0.4252 \end{pmatrix}^\top. \quad (5.56a)$$

Then distort the observation with e_m , which is i.i.d. noise with zero mean and variance $\lambda_0 = 10^{-3}$, i.e.,

$$y_m = f_c(x_m, \boldsymbol{\theta}^{c0}) + e_m. \quad (5.56b)$$

This gives a signal SNR of 30 dB.

The prediction and the prediction error variance for both the canonical and overparameterized models can be seen in Figure 5.5. The predictions for the models look very similar. However, the prediction error variances are varying between the different models, but it is always bounded from below by the prediction error variance for the canonical model.

Figure 5.5: Prediction and prediction error variance for a nonlinear model using both a canonical and non-canonical model with higher model order. The data in the experiment is generated from a model describing how the NTF depends on the wheel slip of a car.

5.4 Summary

This chapter has covered how prediction error variance calculated using a linearization approach is affected by overparameterization of the model. The analysis is first done for parts, first for overparameterized non-canonical representation of models with redundant parameters followed by non-canonical representation of overparameterized models with more flexibility. It shown in Theorem 5.1 and Theorem 5.2 that for non-canonical representation of models with redundant parameters, the prediction error variance is the same as for the canonical representation.

Furthermore, overparameterized models linear in the parameters with more flexibility, it is shown in Theorem 5.3 that the prediction error variance for the overparameterized non-canonical representation of models is bounded from below by the prediction error variance of the canonical representation of the model. If the prediction error is small, this would also be true for a nonlinear model.

6

Experimental and simulation study

To illustrate how the expression for uncertainty, derived in Chapter 4, can be applied on NNs in practice, and hence increase the usefulness of NNs, a simulation study and a study using experimental data have been conducted. The focus of the simulation study is to illustrate the confidence intervals derived in Chapter 4 and Chapter 5. while the focus of the study using experimental data is to emphasize how uncertainty in the prediction varies between models from different model sets.

First, the setting in which the data is collected and the nonlinear model used to simulate the data for the simulation study will be introduced. Afterwards, the simulation study is presented followed by the results of the study done on the experimental data.

6.1 Tire-road friction

The data and models considered in the simulation and experimental studies are inspired by autonomous car applications, where the drive-line is a part of the model for the vehicular dynamics control systems. The focus in this thesis is on the tire-road friction, which is a critical component in many ADAS. For instance, the friction level influences the safety distance in an ACC, which avoidance manoeuvres that are most effective, and vehicular dynamics control systems such as the ABS. If the NN is intended to be used in any of these cases, the NN should also be able to approximate the friction level from wheel slip measurements. The simulation data is generated from a parametric model called the *magic tire formula* presented by Pacejka and Besselink [1997]. The model describes how the

NTF depends on the wheel slip and is given by

$$\mu = D \sin \left(C \arctan \left((1 - E)s + E/B \arctan(Bs) \right) \right), \quad (6.1)$$

where μ is the NTF, s is the wheel slip, and the parameters B, C, D, E depend on the surface. The wheel slip is a ratio between the speed of the vehicle and the speed of the driving wheels, i.e., how much the wheels are slipping on the surface.

6.1.1 Data processing

The friction data used in this thesis comes from experiments conducted by NIRA Dynamics. The data includes measurements from experiments with three different vehicles driving on snow and gravel. A first step of processing is to remove data where actively braking and shifting gear. This is done since the tire-friction curve is not expected to follow the *magic tire formula* under those circumstances

The slip, s , and NTF, μ , are calculated according to

$$s = \frac{\omega_j R - v}{v}, \quad (6.2a)$$

$$\mu = \frac{F_f}{N_f}, \quad (6.2b)$$

where $\omega_j, j = \{l, r\}$, is the angular velocity of the right or left driven wheel, R the wheel radius, v the velocity of the vehicle, F_f the traction force between the tire and the road, and N_f is the normal force. This can be derived from the single wheel dynamics seen in Figure 6.1.

In practice, F_f and N_f are not directly measurable, and need to be estimated by using data from sensors measuring other quantities. This can for instance be done using

$$F_f = \frac{Ti}{2R}, \quad (6.3a)$$

$$N_f = \frac{Mgx_b + z_g a_x M}{x_b + x_f}, \quad (6.3b)$$

where T is the engine torque, $i = v_e/(\omega_l R + \omega_r R)$ is the gear ratio, v_e is the engine speed, x_b is the vertical distance between the vehicle's center of gravity and the driving wheel, x_f is the vertical distance between the center of gravity and the driving and the non-driving wheel, z_g is the horizontal distance from the center of gravity to the ground, M is the mass of the vehicle, and a_x is the longitudinal acceleration of the vehicle, which are quantities that can either be measured or are known. For derivation see [Gustafsson, 1997].

6.1.2 Magic tire formula with bias

In the *magic tire formula*, given in (6.1), it is assumed that the NTF is zero when the slip is zero. To handle the case when there exists a bias in the collected data,

Figure 6.1: Single wheel dynamics.

a bias term μ_0 is added as parameter to the parametric model according to

$$f(\mathbf{x}_n, \boldsymbol{\theta}) = \mu_0 + D \sin \left(C \arctan \left((1 - E)\mathbf{x}_m + E/B \arctan(B\mathbf{x}_m) \right) \right) \quad (6.4a)$$

$$\boldsymbol{\theta} = (B, C, D, E, \mu_0), \quad (6.4b)$$

where $\mathbf{x}_m$ is the measured wheel slip at time m .

6.2 Simulation study

The simulation study aims to illustrate the practical use of the confidence bound derived earlier in the thesis, as well as to show how the different sources of uncertainty contribute to the total uncertainty in the model, which is described in Chapter 2. Especially, the flexibility of NNs is illustrated, which is seen from the fact that model error is significant smaller than the estimation error in this example.

6.2.1 Simulation setup

In order to separate the effect of the estimation error from the model error, the simulation is performed in two steps. First the simulated data from the magic tire formula, i.e., the true output model $f^*(\mathbf{x}_n)$, is used to estimate a reference model $f(\mathbf{x}_n, \boldsymbol{\theta}^{i*})$. Then noise is added to the output of the reference model, which in turn is used to estimate models of the same structure, referred to as $f(\mathbf{x}_n, \hat{\boldsymbol{\theta}}_N^{i(m)})$, in an MC-like procedure. In the MC-simulation, 10^3 different models, $f(\mathbf{x}_n, \hat{\boldsymbol{\theta}}_N^{i(m)})$,

$m = 1, \dots, 10^3$, are fitted to the reference model. Then the model error e_M and the estimation error e_E are computed as

$$e_M = \frac{1}{N} \sum_{n=1}^N \|f^*(\mathbf{x}_n) - f(\mathbf{x}_n, \boldsymbol{\theta}^{i*})\|^2, \quad (6.5a)$$

$$e_E = \frac{1}{NM} \sum_{m=1}^M \sum_{n=1}^N \|f(\mathbf{x}_n, \boldsymbol{\theta}^{i*}) - f(\mathbf{x}_n, \hat{\boldsymbol{\theta}}_N^{i(m)})\|^2. \quad (6.5b)$$

The simulations are performed for a different number of parameters in the reference model, i.e., one MC-simulation is performed for each choice of the number of parameters. This is to illustrate how the different sources of error and uncertainty depend on the flexibility of the model. It can also be seen how a model that is more flexible than needed (overparameterized) will result in larger uncertainty. This is especially true in areas where there is no training data available.

For training, $N = 200$ samples were used; both for the MC-simulation and the reference model. The noise added in the MC-simulations is Gaussian distributed with known variance $\lambda_0 = 0.01$. This gives an SNR of 20 dB. As a nonlinear model structure for the reference model, NNs with two-layers $\text{NN}_{2,l}$ s with a sigmoid as activation function is used and the number of nodes in the hidden layer varies between $l = 1, \dots, 4$. The NNs are trained with the ADAM optimizer, presented by Kingma and Ba [2015] with a learning rate of $\eta = 0.01$ and the standard values of the hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

The results from the simulations are separated into three parts:

- How the different sources of uncertainty contribute to the prediction error variance, in Section 6.2.3.
- An illustration of the prediction error in areas with and without training data, in Section 6.2.4.
- How many observations are needed to make the asymptotic results valid, in Section 6.2.5.

Before moving on to the results of the simulation study, we would like to point out some key differences compared to previous work.

6.2.2 Relation to previous work

There are some important differences in how the main result from Chapter 4 is motivated in this thesis compared to previous work. In Hwang and Ding [1997], $n + 1$ data points are generated where n points are used for training the model and predicting the $(n + 1)$ th sample within a confidence interval. By repeating the experiment multiple times and counting the number of times the confidence interval covers the true point, the analytic expression of the variance is motivated. This can be compared to our simulation study, where the confidence interval is calculated once followed by multiple realizations of NNs where the prediction is done over a grid. Rivals and Personnaz [2000] compare the analytic expression to

Figure 6.2: Total error, model error, and estimation error, calculated according to (6.5), as function of the size of the model.

the variance computed from multiple realizations of NNs, but use the true parameters θ^{c0} to compute the uncertainty in the NN prediction, instead of the estimate $\hat{\theta}_N^i$ used in our simulation study. He and Li [2011] and Papadopoulos et al. [2001], use simulated data, and results are presented over multiple realizations of NNs, but the analytic expressions are calculated over an ensemble of NNs, while Chrysosolouris et al. [1996] evaluate the estimated variance on real data, but does not compare the result to some other models, as is in done in the experimental study in this thesis.

Another difference is that these papers state that the output is Student- t distributed. This would have been true if the model is linear. In Chapter 4, the analysis has been asymptotic in the number of training data. Since asymptotically, when the number of training data N increases, the Student- t distribution converges to a Gaussian distribution, similar results as the cited papers are expected in regions where training data is available. This claim is strengthened by the provided analysis of how many observations are needed for the asymptotic result to be reasonable, which are fewer than would be expect.

Furthermore, neither of these publications covers the case where the NN extrapolates to areas where no data was available during training, and how this affects the variance of the model. The work in this thesis also includes investigating the contribution of different sources of uncertainty, as well as the results from Chapter 5 where overparameterized models are discussed.

6.2.3 Model and estimation error

In Figure 6.2 the model and estimation error are plotted as a function of the size and flexibility of the model, i.e., for NN as a function of the number of nodes in the hidden layer. As expected, and seen from the Figure 6.2, the model error decreases with the size of the model, and the estimation error increases. Already in an NN with 2 nodes the model error is a magnitude smaller than the estimation error, and increasing the model order further will not significantly contribute to decreasing the prediction error. This highlights how flexible model classes that includes NNs are. It can also be conclude that in this example, the estimation error is of greater interest than the model error when investigating the uncertainty in the prediction error for NNs.

6.2.4 Prediction error uncertainty

The prediction and the MC realizations for a $NN_{2,2}$ and $NN_{2,3}$ are plotted with a 3σ confidence interval, see Figure 6.3a and Figure 6.3b, respectively. The interval is calculated by taking the variance of the MC realizations (in dashed blue) and using (4.25a) with $P_\theta(\theta^{i0}) = \hat{P}_N$ (in dashed red), i.e., a lower bound on the variance. The region with training data is indicated by a grey box, and the true magic tire formula in dashed green.

As one would expect, the uncertainty is relatively small in the region with training data and grows in regions away from where training data was collected. The mean of the MC realizations and the simulated output from the magic tire formula coincide in the region containing training data, but they start to diverge from each other the further away from the region with training data the prediction gets. This is especially true for the 3σ confidence interval; in the region with training data, the confidence interval calculated from the MC realizations and the one calculated using (4.25a) coincide while they do not in the region without training data.

Since already $NN_{2,2}$ can represent the output from the magic tire formula almost perfectly one can conclude that the larger $NN_{2,3}$ is an overparameterized model, hence one would expect the confidence interval to be bounded from below by the model with lower model order. This is especially clear in regions without training data, as the confidence interval calculated using (4.25a) in Figure 6.3b is much larger compared to the one in Figure 6.3a, this is also true for the confidence interval from the MC realizations. Hence the price paid for a more flexible model is that its prediction could be more uncertain. In reality, the true model structure is seldom known, hence overparameterized models such as $NN_{2,3}$ are more commonly used. Then, a large confidence interval in regions without training data is essential to indicate uncertainty in the model there. Overparameterization can thus be an instrument to reveal when the model extrapolates in regions that lack training data.

The problem with overfitting can also be seen in Figure 6.4 where a simulation is done for a two-layer NN with sixteen nodes in the hidden layer. As soon as the prediction leaves the region with training data, as for the $NN_{2,3}$, the confidence interval gets large and does not give more information than that the prediction is

(a) Prediction error of NN with 2 nodes

(b) Prediction error of NN with 3 nodes

Figure 6.3: Analytically and empirically calculated prediction uncertainty for a two-layer NN with 2 and 3 nodes in the hidden layer, respectively.

Figure 6.4: Analytically and empirically calculated prediction uncertainty for a two layer NN with sixteen nodes in the hidden layer.

done in a region without training data. But here, the mean of the MC realizations also diverges from the prediction of the reference model (NN prediction in red). It can also be seen that for some realizations the estimated function is not as smooth in the region with the training data compared to the models with fewer parameters.

6.2.5 Asymptotic results

The results presented in Chapter 4 are asymptotic in the number of observations, hence it is of interest to see how many observations that are needed for the approximation to be valid. This can be achieved by training multiple two-layer NNs with two hidden nodes in the hidden layer for different numbers of observations that are randomly selected between -60% and 60% wheel slip. The same observations are used in all the MC simulations. The result of such an experiment can be seen in Figure 6.5. It is possible to see that already after 40 observations the approximation seems to be valid. In Figure 6.6, one can see that the estimation error decreases as the number of observations increases, which is a well-known result.

(a) 20 observations

(b) 40 observations

Figure 6.5: Prediction error variance for a two-layer NN with two hidden nodes in the hidden layer for an increasing number of observations.

Figure 6.6: Model and estimation error as the number of observations increases, calculated according to (6.5).

6.3 Experimental study

In the experimental study, presented in this section, the uncertainty of different models trained on real-world data is investigated. The tire-road friction data set is provided by NIRA Dynamics, which is described in Section 6.1. The tire-road friction is a well-studied experiment setup and the analysis has recently been extended to the Bayesian framework with uncertainty in the prediction, see [Berntorp and Hiroaki, 2019].

In this study, model sets that are both linear and nonlinear in the parameters are investigated. A commonly used numeric measure to compare the different models is the model fit, which is defined as

$$\text{fit}(\hat{\mathbf{y}}_N) = 100 \left(1 - \sqrt{\frac{\sum_{n=1}^N (y_n - \hat{y}_n)^2}{\sum_{n=1}^N (y_n - \frac{1}{N} \sum_{n=1}^N y_n)^2}} \right). \quad (6.6)$$

It is often presented as a percentage of the fit to the data since the best possible outcome is 100. The output of the model can however be arbitrarily bad, and a negative fit is possible.

Apart from predicting the NTF, a 95% confidence interval is calculated for all the different models, i.e., a 2σ interval where $\sigma_n = \sqrt{\text{Var}(\hat{y}_n)}$, i.e., the variance of the estimated signal. This variance is calculated as the sum of the variance of the model and the variance of the measurement noise, i.e.,

$$\text{Var}(\hat{y}_n) = \text{Var}(\hat{s}_n) + \lambda_N. \quad (6.7)$$

6.3.1 Model order selection

In Figure 6.7 the cost function, the AIC, and BIC are shown for a different model structure that is linear in the parameters. For the friction data, the different polynomial model classes are tested, i.e., normal polynomials, Chebyshev polynomials, and Laplacian polynomials, and NNs.

For the NNs, the notation $[d_x, d_1, \dots, d_L, d_s]$ is used, where d_x is the number of inputs, d_s is the number of outputs and d_l is the number of nodes in l th hidden layer. An NN with d_x inputs, d_s outputs, and L hidden layer has

$$(d_x + 1)d_1 + \sum_{l=1}^{L-1} (d_l + 1)d_{l+1} + (d_L + d_L)d_s, \quad (6.8)$$

parameters. The result for NNs can be seen in Figure 6.8. An interesting observation is that the model order that minimizes the AIC and BIC criteria for the real data coincides with the model order where the total error mainly consists of estimation error for the simulated data.

6.3.2 Results

The included model sets in the survey are:

- The three different choices of basis functions described in Section 2.2.1. Namely:
 - Polynomials
 - Chebyshev polynomials
 - Laplacian polynomials
- Fully-connected NNs described in Section 2.2.2.
- BNNs described in Section 3.5 with prior of the parameter given by the Xavier initialization, [Glorot and Bengio, 2010],
- One-head NNs that learn their own uncertainty described in Section 3.3
- The magic tire formula presented by Pacejka and Besselink [1997] with an added bias term (6.4).
- GP described in Section 3.4 with the squared exponential kernel with length scale

$$L = \frac{1}{N} \sum_{i=1}^{d_x} \sqrt{\text{Var}[\mathbf{x}_n]_i}, \quad (6.9a)$$

and standard deviation of the predictor as

$$\sigma_f^2 = \frac{1}{\sqrt{2}} \sqrt{\text{Var}(y)}. \quad (6.9b)$$

(a) Polynomial

(b) Chebyshev polynomial

(c) Laplacian polynomial

Figure 6.7: Cost function AIC and BIC for different models linear in the parameters evaluated on the friction data.

Figure 6.8: Cost function AIC and BIC for NNs evaluated on the friction data.

The uncertainty in the prediction is calculated using the linearization described in Chapter 4 and Chapter 5 for the parametric models, and for the Bayesian models the uncertainty is obtained by sampling from the posterior. Using the result from Section 6.3.1 to choose the correct model order, polynomials and NNs are fitted to the tire-road friction data. That is, for the polynomials six basis functions are used, for the Chebyshev polynomials four basis functions, for the Laplacian polynomials four basis functions, and for the fully-connected feed-forward NN two layers with two hidden units in the hidden layer and a sigmoid function as activation function.

In system identification, it is common to categorize model structures based on how much they come from physical insights about the system or if they are universal approximators, [Schoukens and Ljung, 2019]. The magic tire formula falls into the category of gray-box models, a family of models derived primarily by the usage of physical laws where the parameters are estimated by data. Polynomials, NNs, and GPs on the other hand, would fall into the category of black-box models, i.e., universal approximators without any assumptions on the underlying system. The inclusion of GPs in the survey is further motivated by their close connection to both NNs and Laplacian polynomials, see [Neal, 1996, Svensson and Schön, 2017], as well as model uncertainty being naturally captured.

For the linear models, the LLSS is used to fit the models, while for the non-linear model different optimization algorithms were used to solve the nonlinear least squared problem, i.e., fit the model to the data. The different optimization algorithms used are: ADAM optimizer [Kingma and Ba, 2015] for the fully-connected NN, RMS-prop [Tieleman and Hinton, 2012] for the one-headed NN with uncertainty, and LM algorithm [Nocedal and Wright, 2006] for the magic tire formula with bias. The Bayesian NNs are trained by the Metropolis-Hastings sampler, and the GPs are implemented using the SML (Statistics and Machine Learning) toolbox in MATLAB. A summary of the optimization algorithm, model order, and categorization for the different models can be seen in Table 6.1.

Figure 6.9: Estimation of friction data using different polynomial models.

(a) Fully-connected NN

(b) One-headed NN with learned uncertainty

(c) Bayesian NN with Gaussian prior on the parameters

Figure 6.10: Estimation of friction data using different NN structures.

(a) Magic tire formula with a bias

(b) GP

Figure 6.11: Estimation of friction data using different models.

Table 6.1: Table of the settings for the different models used to estimate the friction data. This includes, implementation specifics such as chosen optimization (learning) algorithm, which model order (or model setting) that was used, and categorization of the different model structures, i.e., classification of the model into linear black-box (LBB) models, nonlinear black-box (NLBB) models, and nonlinear gray-box (NLGB) models.

Model	Learning algorithm	Model order	Classification
Polynomials	LLSS	$P = 6$	LBB
Chebyshev polynomials	LLSS	$P = 4$	LBB
Laplacian polynomials	LLSS	$P = 4$	LBB
Fully-connected NN	ADAM	$[1, 2, 1]$.	NLBB
One-head with uncertainty	RMS-prop	$[1, 6, 6, 6, 2]$.	NLBB
BNN	Metropolis-Hastings	$[1, 2, 1]$, $p(\theta) = (2.33)$	NLBB
Magic tire formula	LM	See (6.4)	NLGB
GP	SML toolbox	Kernal: (3.8), $L = (6.9a)$ $\sigma_f^2 = (6.9b)$.	NLBB

Table 6.2: Model fit for different models used to estimate the friction data.

Model	Fit to data	
	Training	Validation
Polynomials	69.26	69.15
Chebyshev polynomials	68.82	68.87
Laplacian polynomials	68.82	68.87
Fully-connected NN	67.74	67.04
One head with uncertainty	66.32	66.32
BNN	66.87	66.48
Magic tire formula	68.47	68.26
GP	69.20	68.03

Studying the result in Figure 6.9, Figure 6.10, and Figure 6.11, it can be seen that predictions using models from different model sets look very similar. This is supported by the fact that the fits of the different models seen in Table 6.2 are similar for the different choices of model sets. It can also be noticed that the fit is quite low for all models. This is likely a consequence of the observations being noisy. Hence, without also modeling the noise the fit of the model will be low.

Another observation is that the uncertainty in prediction for different models looks very similar. Especially, it can be noted that the uncertainty in the prediction for polynomials, which are linear in the parameters has similar uncertainty as to the linearized versions of the nonlinear models, namely, NN, magic tire formula, and GP. It also seems like by changing the cost function of an NN, it is possible to include the uncertainty as an additional output of the model.

6.4 Summary

In this chapter, the linearization approach to obtain prediction error variance in NN presented in Chapter 4 is compared to an ensemble approach and NNs that learn their own uncertainty. The comparison to an ensemble approach is made in the simulation study in Section 6.2. There it is shown that the prediction error variance calculated using the linearization approach yields similar result as the ensemble approach in regions with training data, but underestimate the prediction error variance for overparameterized models in regions without training data. In the experimental study in Section 6.3, a comparison is made between the linearization approach and NNs that learn their own uncertainty. It can be concluded that for the considered example the prediction error variance looks similar for the two approaches.

The simulation study also includes an investigation of different sources of uncertainty, i.e., uncertainty due to model error and uncertainty due to estimation error. It is shown that in the considered example, the contribution to the prediction error variance due to estimation error is significantly larger than the contribution due to the model error.

In the experimental study, the prediction and prediction error variance from the NNs are also compared to the prediction and prediction error variance from other black-box models and a gray-box models. Especially, the prediction error variance looks similar for the linearization approach and the prediction error variance for the GPs and BNNs, which are models that naturally include uncertainty in their prediction.

7

Conclusion

Applications of NNs in e.g., computer vision have shown amazing performance, but also spectacular failures. Biased training data is the main reason for failures, where the network must extrapolate its model to regions without support. Interpolation in regions with good support of training data is an easier task. The holy grail in NN applications would be an uncertainty assessment of the prediction that can handle both cases. In that way, NNs could be used in safety-critical applications where it is fused with other information sources. For instance, in automotive collision avoidance systems, object detection from a computer vision NN can be combined with the prediction from radar and lidar by weighing the prediction uncertainty from each subsystem appropriately,

In search for a way to quantify the uncertainty in the predictions produced by NN, the application of methods for uncertainty estimation in system identification tools, has investigated. More precisely, the feasibility and challenges of estimating the prediction error variance of the NN by linearizing the model and then propagating the uncertainty of the learned model parameters to the output have been investigated. The results show that, compared to existing methods based upon creating ensembles of NNs or extending the NN structure to also learn the uncertainty, the linearization method produces similar results for the estimated prediction error variance. Furthermore, the investigation has highlighted three main challenges in using the linearization approach to quantify the uncertainty in the predictions produced by NNs. These are:

- (i) the calculation of the covariance of the learned model parameters,
- (ii) the non-unique parameterization of the NN, and
- (iii) the effects of over- and under-modeling.

As shown in the thesis, the covariance of the parameter estimate can be calculated from the Hessian of the cost-function. Since the Hessian of the cost function

generally is calculated during the minimization of the cost-function, it is in theory straight forward to calculate the parameter covariance. Unfortunately, for NNs with a large number of parameters the matrix inversion needed to calculate the parameter covariance from the Hessian might be infeasible to perform. Therefore, for the linearization approach to be applicable to large NN methods to approximate the parameter covariance must be developed.

The analysis of the non-unique parameterization has shown that using the linearization approach, the ambiguities can be neglected for irreducible NNs, i.e., NNs that are not overparameterized nor have redundant parameters. This is because, for all parameter sets that minimize the cost function of the irreducible NN, the parameter covariance and the prediction error variance become equivalent. In practice, very large NNs that rarely are irreducible, but rather overparameterized, are commonly used. However, the conducted analysis has shown that the prediction error variance for the overparameterized model is lower bounded by the prediction error variance of the irreducible NN.

In regions where the NN extrapolates, i.e., makes predictions outside the training data region, it is shown that the linearization approach underestimated the prediction error variance for overparameterized NNs. That is, the linearization approach produces a lower bound on the uncertainty for a given model, where the bound is tight in regions where training data is available. This highlights one of the main difficulties in calculating uncertainty for NN predictions. Without knowledge about which regions the training data has covered, it is hard to know if the confidence in the prediction is too low due to that the NN extrapolate. The results presented in this thesis are for small NNs with low dimensional inputs. As the number of parameters of the NN increases and the dimension of the input grows, the notion of which regions the training data has covered get more complicated. Hence, it is difficult to know where the NN extrapolate.

To summarize, many questions remain about uncertainties quantization of predictions by NNs, e.g., how should one identify and handle situations when the NN extrapolates? Or how does the choice of optimization algorithm used in the training of the NN affect the results? The result in this thesis is has been derived without taking into account the regularization commonly used in the training of NN. It is known that regularization might add some bias to the parameters, which in turn would be propagated to a bias in the prediction. But how does regularization affect the prediction error variance calculated using the linearization approach? These are some questions that could be answered in the continuation of this work.

Bibliography

- Hirotsugu Akaike. A new look at the statistical model identification. *IEEE Trans. Autom. Control*, 19(6):716–723, 1974.
- Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P Xing. Learning scalable deep kernels with recurrent structure. *The J. of Mach. Learn. Research (JMLR)*, 18(1):2850–2886, 2017.
- Carl Andersson, Antônio H Ribeiro, Koen Tiels, Niklas Wahlström, and Thomas B Schön. Deep convolutional networks in system identification. In *Proc. of IEEE 58th Conf. on Decision and Contr. , (CDC)*, pages 3670–3676, Nice France, 2019, 11-19 Dec.
- Saeed Asadi Bagloee, Madjid Tavana, Mohsen Asadi, and Tracey Oliver. Autonomous vehicles: challenges, opportunities, and future implications for transportation policies. *J. of Modern Trans.*, 24(4):284–303, 2016.
- BBC News. Tesla in fatal California crash was on Autopilot. 31 March 2018. URL <https://www.bbc.com/news/world-us-canada-43604440>. [Last accessed: 9 Sep. 2020].
- Karl Berntorp and Kitano Hiroaki. Bayesian learning of tire friction with automotive-grade sensors by gaussian-process state-space models. In *Proc. of IEEE 58th Conf. on Decision and Contr. , (CDC)*, pages 6681–6686, Nice France, 2019. IEEE. 11-19 Dec.
- C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN 0-387-31073-8. New York, NY, USA.
- Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. In *Proc. of the 32Nd Int. Conf. on Mach. Learn. (ICML)*., pages 1613–1622, Lille, France, 2015. 6–11 Jul.
- George EP Box. Science and statistics. *J. of the Am. Stat. Assoc. (JSTOR)*, 71(356): 791–799, 1976.
- J. G. Carney, P. Cunningham, and U. Bhagwan. Confidence and prediction intervals for neural network ensembles. In *Proc. of the 6th Int. Joint Conf. Neural Netw. (IJCNN)*, Washington DC, DC, USA, 1999. 10-16, July.

- G. Chrysosoiouris, M. Lee, and A. Ramsey. Confidence interval prediction for neural network models. *IEEE Trans. Neural Netw.*, 1996.
- Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *Proc. of IEEE Int. Conf. on Robot. and Autom. (ICRA)*, pages 1–9, Stockholm, Sweden, 2018. 16–21, May.
- Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Proc. of 16th Int. Conf. on Artificial Intell. and Statistics. (AISTATS)*, Scottsdale, AZ, USA, 2013. Apr 29 - May 1.
- Alexander D’Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, Farhad Hormozdiari, Neil Houlsby, Shaobo Hou, Ghasen Jerfel, Alan Karthikesalingam, Mario Lucic, Yian Ma, Cory McLean, Diana Mincu, Akinori Mitani, Andrea Montanari, Zachary Nado, Vivek Nataraajan, Christopher Nielson, Thomas F. Osborne, Rajiv Raman, Kim Ramasamy, Rory Sayres, Jessica Schrouff, Martin Seneviratne, Shannon Sequeira, Harini Suresh, Victor Veitch, Max Vladymyrov, Xuezhi Wang, Kellie Webster, Steve Yadlowsky, Yun Taedong, Xiaohua Zhai, and D. Sculley. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020.
- Annette J Dobson and Adrian G Barnett. *An introduction to generalized linear models*. Chapman and Hall/CRC, Boca Raton, FL, USA, 2008.
- Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proc. of IEEE 1st Annual Conf. on Robot Learn. , (CoRL)*, pages 1–16, Mountain View, CA, USA, 2017. Nov 13-15.
- Abdelrahman Eldesokey, Michael Felsberg, and Fahad Shahbaz Khan. Propagating confidences through cnns for sparse data regression. In *British Mach. Vision Conf. (BMVC)*, page 14, 2018. Newcastle, UK, Sep 3-6.
- Kevin Eykholt, Ivan Evtimov, Earlene Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust Physical-World Attacks on Deep Learning Visual Classification. In *Proc. of IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR)*, pages 1625–1634, 2018. Salt Lake City, UTAH, USA, June 18-22.
- Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Appendix. *arXiv:1506.02157*, 2015.
- Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proc. of the 33rd Int. Conf. on Mach. Learn. (ICML)*., pages 1050–1059, New York, NY, USA, 2016. 20–22 Jun.

- Yrin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, January 2017.
- Marta Garnelo, Jonathan Schwatz, Dan Rosenbaum, Fabio Viola, Danilo Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural Processes. In *Proc. of the 35th Int. Conf. on Mach. Learn. (ICML) Workshop on Theo. Founda. and Appl. of Deep Generative Models*, Stockholm, Sweden, 2018. URL <https://arxiv.org/abs/1807.01622>. 10–15 Jul.
- Claude Gasquet and Patrick Witomski. *Fourier analysis and applications: filtering, numerical computation, wavelets*, volume 30. Springer Science & Business Media, New York, NY, USA, 2013.
- Michel Gevers, Alexandre Sanfelice Bazanella, Xavier Bombois, and Ljubisa Miskovic. Identification and the information matrix: how to get just sufficiently rich? *IEEE Trans. Autom. Control*, 54(12):2828–2840, 2009.
- Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452 – 459, 2015. ISSN 0028-0836.
- Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proc. of the 34th Int. Conf. on Mach. Learn. (ICML)*., pages 1263–1272, Sydney, Australia, 2017. 06–11 Aug.
- Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of 16th Int. Conf. on Artificial Intell. and Statistics. (AISTATS)*, pages 249–256, Sardinia, Italy, 2010.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Adv. in Neural Inf. Process. Syst. (NIPS) 28*, pages 2672–2680, Montréal, QC, Canada, 2014. 8-13 Dec.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016. ISBN 9780262035613. London, England.
- Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *J. of Field Robotics*, 37(3):362–386, 2020.
- F. Gustafsson. *Statistical Sensor Fusion*. Studentlitteratur: Lund Sweden, 2018.
- F. Gustafsson, L. Ljung, and M. Millner. *Signal Processing*. Studentlitteratur: Lund Sweden, 2011.
- Fredrik Gustafsson. Slip-based tire-road friction estimation. *Automatica*, 33(6): 1087–1099, 1997.
- Fredrik K Gustafsson, Martin Danelljan, Goutam Bhat, and Thomas B Schön. Energy-Based Models for Deep Probabilistic Regression. In *Proc. of 16th European Conf. on Comput. Vision (ECCV)*, Online, 2019a. 23-28 Aug.

- Fredrik K. Gustafsson, Martin Danelljan, and Thomas B. Schön. Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision. In *Adv. in Neural Inf. Process. Syst. (NIPS)* 33, 2019b. Vancouver, Canada.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. of IEEE Int. Conf. on Comput. Vision (ICCV)*, pages 1026–1034, Santiago, Chile, 2015. 7-13 Dec.
- Shouling He and Jiang Li. Confidence Intervals for Neural Networks and Applications to Modeling Engineering Materials. In *Artificial Neural Netw.*, chapter 16. IntechOpen, 2011.
- Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.*, 29 (6):82–97, 2012a.
- Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. In *arXiv:1207.0580*, 2012b.
- J. T. Gene Hwang and A. Adam Ding. Prediction Intervals for Artificial Neural Networks. *J. Am. Stat. Assoc. (JSTOR)*, 92(438):748–757, 1997.
- Eddy Ilg, Ozgun Cicek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proc. of 15th European Conf. on Comput. Vision (ECCV)*, Munich, Germany, 2018. 8-14 Sep.
- Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proc. of the 32Nd Int. Conf. on Mach. Learn. (ICML)*, Lille, France, 2015. 6–11 Jul.
- Steven M Kay. *Fundamentals of statistical signal processing Estimation theory*. Prentice Hall PTR, cop. 1993: Upper Saddle River, NJ, USA, 1993.
- A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *Proc. of IEEE Int. Conf. on Robot. and Autom. (ICRA)*, pages 4762–4769, Stockholm, Sweden, 2016. 16–21, May.
- Alex Kendall and Yarin Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Adv. in Neural Inf. Process. Syst. (NIPS)* 31, pages 5574–5584. Curran Associates, Inc., 4–9 Dec 2017. Long Beach, CA, USA, 4–9 Dec.
- Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proc. of 3rd Int. Conf. for Learn. Representations (ICLR)*, 7–9 May 2015. San Diego, CA, USA.

- Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. In *Proc. of the 34th Int. Conf. on Mach. Learn. (ICML)*., pages 1885–1894, Sydney, Australia, 2017. 06–11 Aug.
- Erwin Kreyszig. *Introductory functional analysis with applications*, volume 1. Wiley and Sons New York, NY, USA, 1978.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Adv. in Neural Inf. Process. Syst. (NIPS)* 25, pages 1097–1105, Lake Tahoe, NVn USA, 2012. 3-8, Dec.
- Hannelore Liero and Silvelyn Zwanzig. Introduction to the theory of statistical inference. In *Chapman and Hall CRC Texts in Statistical Science*, Boca Raton, FL, USA. 2011. ISBN 9781439852927.
- Lennart Ljung. *System identification: theory for the user (2nd edition)*. PTR Prentice Hall: Upper Saddle River, NJ, USA, 1999.
- Lennart Ljung and Peter E Caines. Asymptotic normality of prediction error estimators for approximate system models. *Stochastics*, 3, 1980.
- Lennart Ljung, Carl Andersson, Koen Tiels, and Thomas B. Schön. Deep Learning and System Identification. In *Proc. of 21th IFAC World Congress, (IFAC)*, Online (Berlin, Germany), 2020. 11-17 Jul.
- David JC MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.
- Magnus Malmström, Isaac Skog, Sara Modarres Razavi, Yuxin Zhao, and Fredrik Gunnarsson. 5G Positioning - A Machine Learning Approach. In *Proc. of IEEE 16th Workshop on Positioning Navig. Commun. , (WPNC)*, Bremen, Germany, 2019. 23-24 Oct.
- Magnus Malmström, Isaac Skog, Daniel Axehill, and Fredrik Gustafsson. Asymptotic Prediction Error Variance for Feedforward Neural Networks. In *Proc. of 21th IFAC World Congress, (IFAC)*, Online (Berlin, Germany), 2020. 11-17 Jul.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proc. of 1st Int. Conf. for Learn. Representations (ICLR)*, Scottsdale, AZ, USA, 2013. 2-4, May.
- Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media: New York, NY, USA, 1996.
- Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media,: New York, NY, USA, 2006.
- NTSB. Preliminary Report Highway HWY18MH010. Technical specification (ts), National Transportation Safty Board (NTSB), 05 2018. URL <https://www.nts.gov/investigations/AccidentReports/Reports/HWY18MH010-prelim.pdf>.

- Ian Osband. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *Adv. in Neural Inf. Process. Syst. (NIPS) 30 Workshop on Bayesian Deep Learn.*, volume 192, 2016. Barcelona, Spain, 5–10 Dec.
- Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Adv. in Neural Inf. Process. Syst. (NIPS) 30*, 2016. Barcelona, Spain, 5–10 Dec.
- HB Pacejka and IJM Besselink. Magic formula tyre model with transient properties. *Veh. syst. dynamics-Int. J. of Veh. Mechanics and Mobility*, 27(S1):234–249, 1997.
- G. Papadopoulos, P.J. Edwards, and A.F Murray. Confidence estimation methods for neural networks: a practical comparison. *IEEE Trans. Neural Netw.*, 2001.
- Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab, and Federico Tombari. Sampling-free Epistemic Uncertainty Estimation Using Approximated Variance Propagation. In *Proc. of IEEE Int. Conf. on Comput. Vision (ICCV)*, pages 2931–2940, Seoul, South Korea, 2019. Oct 27 - Nov 3.
- Reuters. Tesla in fatal California crash was on Autopilot. 21 Aug 2017. URL <https://www.reuters.com/article/us-tesla-crash-idUSKCN1B12DF>. [Last accessed: 9 Sep. 2020].
- Isabelle Rivals and Léon Personnaz. Construction of confidence intervals for neural networks based on least squares estimation. *Elsevier J. Neural Netw.*, 13, 2000.
- Arturo Rosenblueth and Norbert Wiener. The role of models in science. *Philosophy of science*, 1945.
- David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- Johan Schoukens and Lennart Ljung. Nonlinear system identification: A user-oriented road map. *IEEE Control Syst. Mag.*, 39(6):28–99, 2019.
- Gideon Schwarz. Estimating the dimension of a model. *Ann. of Stat.*, 6(2):461–464, 1978.
- Jonas Sjöberg, Håkan Hjalmarsson, and Lennart Ljung. Neural networks in system identification. *IFAC Proceedings Volumes*, 27(8):359–382, 1994.
- Arno Solin and Simo Särkkä. Hilbert space methods for reduced-rank Gaussian process regression. *Statistics and Computing*, 30:1–28, 2019.
- N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, Feb 2014.
- Héctor J Sussmann. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural netw.*, 5(4):589–593, 1992.

- Andreas Svensson and Thomas B Schön. A flexible state–space model for learning nonlinear dynamical systems. *Automatica*, 80:189–199, 2017.
- Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian Uncertainty Estimation for Batch Normalized Deep Networks. In *Proc. of the 35th Int. Conf. on Mach. Learn. (ICML)*., 6–11 Jul 2018. Stockholm, Sweden, 6–11 Jul.
- Sergios Theodoridis. *Machine learning: a Bayesian and optimization perspective*. Academic Press, London, UK, 2015.
- Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 2012.
- Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, London, UK, 2006.
- Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *Proc. of the 32Nd Int. Conf. on Mach. Learn. (ICML)*., Lille, France, 2015. 6–11 Jul.
- Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Adv. in Neural Inf. Process. Syst. (NIPS)* 29, Barcelona, Spain, 2016a. 5-10 Dec.
- Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Proc. of 19th Int. Conf. on Artificial Intell. and Statistics. (AISTATS)*, pages 370–378, Cadiz, Spain, 2016b. 9-11 May.

Licentiate Theses
Division of Automatic Control
Linköping University

- P. Andersson:** Adaptive Forgetting through Multiple Models and Adaptive Control of Car Dynamics. Thesis No. 15, 1983.
- B. Wahlberg:** On Model Simplification in System Identification. Thesis No. 47, 1985.
- A. Isaksson:** Identification of Time Varying Systems and Applications of System Identification to Signal Processing. Thesis No. 75, 1986.
- G. Malmberg:** A Study of Adaptive Control Missiles. Thesis No. 76, 1986.
- S. Gunnarsson:** On the Mean Square Error of Transfer Function Estimates with Applications to Control. Thesis No. 90, 1986.
- M. Viberg:** On the Adaptive Array Problem. Thesis No. 117, 1987.
- K. Ståhl:** On the Frequency Domain Analysis of Nonlinear Systems. Thesis No. 137, 1988.
- A. Skeppstedt:** Construction of Composite Models from Large Data-Sets. Thesis No. 149, 1988.
- P. A. J. Nagy:** MaMiS: A Programming Environment for Numeric/Symbolic Data Processing. Thesis No. 153, 1988.
- K. Forsman:** Applications of Constructive Algebra to Control Problems. Thesis No. 231, 1990.
- I. Klein:** Planning for a Class of Sequential Control Problems. Thesis No. 234, 1990.
- F. Gustafsson:** Optimal Segmentation of Linear Regression Parameters. Thesis No. 246, 1990.
- H. Hjalmarsson:** On Estimation of Model Quality in System Identification. Thesis No. 251, 1990.
- S. Andersson:** Sensor Array Processing; Application to Mobile Communication Systems and Dimension Reduction. Thesis No. 255, 1990.
- K. Wang Chen:** Observability and Invertibility of Nonlinear Systems: A Differential Algebraic Approach. Thesis No. 282, 1991.
- J. Sjöberg:** Regularization Issues in Neural Network Models of Dynamical Systems. Thesis No. 366, 1993.
- P. Pucar:** Segmentation of Laser Range Radar Images Using Hidden Markov Field Models. Thesis No. 403, 1993.
- H. Fortell:** Volterra and Algebraic Approaches to the Zero Dynamics. Thesis No. 438, 1994.
- T. McKelvey:** On State-Space Models in System Identification. Thesis No. 447, 1994.
- T. Andersson:** Concepts and Algorithms for Non-Linear System Identifiability. Thesis No. 448, 1994.
- P. Lindskog:** Algorithms and Tools for System Identification Using Prior Knowledge. Thesis No. 456, 1994.
- J. Plantin:** Algebraic Methods for Verification and Control of Discrete Event Dynamic Systems. Thesis No. 501, 1995.
- J. Gunnarsson:** On Modeling of Discrete Event Dynamic Systems, Using Symbolic Algebraic Methods. Thesis No. 502, 1995.
- A. Ericsson:** Fast Power Control to Counteract Rayleigh Fading in Cellular Radio Systems. Thesis No. 527, 1995.
- M. Jirstrand:** Algebraic Methods for Modeling and Design in Control. Thesis No. 540, 1996.
- K. Edström:** Simulation of Mode Switching Systems Using Switched Bond Graphs. Thesis No. 586, 1996.

J. Palmqvist: On Integrity Monitoring of Integrated Navigation Systems. Thesis No. 600, 1997.

A. Stenman: Just-in-Time Models with Applications to Dynamical Systems. Thesis No. 601, 1997.

M. Andersson: Experimental Design and Updating of Finite Element Models. Thesis No. 611, 1997.

U. Forssell: Properties and Usage of Closed-Loop Identification Methods. Thesis No. 641, 1997.

M. Larsson: On Modeling and Diagnosis of Discrete Event Dynamic systems. Thesis No. 648, 1997.

N. Bergman: Bayesian Inference in Terrain Navigation. Thesis No. 649, 1997.

V. Einarsson: On Verification of Switched Systems Using Abstractions. Thesis No. 705, 1998.

J. Blom, F. Gunnarsson: Power Control in Cellular Radio Systems. Thesis No. 706, 1998.

P. Spångéus: Hybrid Control using LP and LMI methods – Some Applications. Thesis No. 724, 1998.

M. Norrlöf: On Analysis and Implementation of Iterative Learning Control. Thesis No. 727, 1998.

A. Hagenblad: Aspects of the Identification of Wiener Models. Thesis No. 793, 1999.

E. Tjärnström: Quality Estimation of Approximate Models. Thesis No. 810, 2000.

C. Carlsson: Vehicle Size and Orientation Estimation Using Geometric Fitting. Thesis No. 840, 2000.

J. Löfberg: Linear Model Predictive Control: Stability and Robustness. Thesis No. 866, 2001.

O. Härkegård: Flight Control Design Using Backstepping. Thesis No. 875, 2001.

J. Elbornsson: Equalization of Distortion in A/D Converters. Thesis No. 883, 2001.

J. Roll: Robust Verification and Identification of Piecewise Affine Systems. Thesis No. 899, 2001.

I. Lind: Regressor Selection in System Identification using ANOVA. Thesis No. 921, 2001.

R. Karlsson: Simulation Based Methods for Target Tracking. Thesis No. 930, 2002.

P.-J. Nordlund: Sequential Monte Carlo Filters and Integrated Navigation. Thesis No. 945, 2002.

M. Östring: Identification, Diagnosis, and Control of a Flexible Robot Arm. Thesis No. 948, 2002.

C. Olsson: Active Engine Vibration Isolation using Feedback Control. Thesis No. 968, 2002.

J. Jansson: Tracking and Decision Making for Automotive Collision Avoidance. Thesis No. 965, 2002.

N. Persson: Event Based Sampling with Application to Spectral Estimation. Thesis No. 981, 2002.

D. Lindgren: Subspace Selection Techniques for Classification Problems. Thesis No. 995, 2002.

E. Geijer Lundin: Uplink Load in CDMA Cellular Systems. Thesis No. 1045, 2003.

M. Enqvist: Some Results on Linear Models of Nonlinear Systems. Thesis No. 1046, 2003.

T. Schön: On Computational Methods for Nonlinear Estimation. Thesis No. 1047, 2003.

F. Gunnarsson: On Modeling and Control of Network Queue Dynamics. Thesis No. 1048, 2003.

S. Björklund: A Survey and Comparison of Time-Delay Estimation Methods in Linear Systems. Thesis No. 1061, 2003.

M. Gerdin: Parameter Estimation in Linear Descriptor Systems. Thesis No. 1085, 2004.

A. Eidehall: An Automotive Lane Guidance System. Thesis No. 1122, 2004.

E. Wernholt: On Multivariable and Nonlinear Identification of Industrial Robots. Thesis No. 1131, 2004.

J. Gillberg: Methods for Frequency Domain Estimation of Continuous-Time Models. Thesis No. 1133, 2004.

G. Hendeby: Fundamental Estimation and Detection Limits in Linear Non-Gaussian Systems. Thesis No. 1199, 2005.

D. Axehill: Applications of Integer Quadratic Programming in Control and Communication. Thesis No. 1218, 2005.

J. Sjöberg: Some Results On Optimal Control for Nonlinear Descriptor Systems. Thesis No. 1227, 2006.

D. Törnqvist: Statistical Fault Detection with Applications to IMU Disturbances. Thesis No. 1258, 2006.

H. Tidefelt: Structural algorithms and perturbations in differential-algebraic equations. Thesis No. 1318, 2007.

S. Moberg: On Modeling and Control of Flexible Manipulators. Thesis No. 1336, 2007.

J. Wallén: On Kinematic Modelling and Iterative Learning Control of Industrial Robots. Thesis No. 1343, 2008.

J. Harju Johansson: A Structure Utilizing Inexact Primal-Dual Interior-Point Method for Analysis of Linear Differential Inclusions. Thesis No. 1367, 2008.

J. D. Hol: Pose Estimation and Calibration Algorithms for Vision and Inertial Sensors. Thesis No. 1370, 2008.

H. Ohlsson: Regression on Manifolds with Implications for System Identification. Thesis No. 1382, 2008.

D. Ankelhed: On low order controller synthesis using rational constraints. Thesis No. 1398, 2009.

P. Skoglar: Planning Methods for Aerial Exploration and Ground Target Tracking. Thesis No. 1420, 2009.

C. Lundquist: Automotive Sensor Fusion for Situation Awareness. Thesis No. 1422, 2009.

C. Lyzell: Initialization Methods for System Identification. Thesis No. 1426, 2009.

R. Falkeborn: Structure exploitation in semidefinite programming for control. Thesis No. 1430, 2010.

D. Petersson: Nonlinear Optimization Approaches to $\mathcal{H}_2$ -Norm Based LPV Modelling and Control. Thesis No. 1453, 2010.

Z. Sjanic: Navigation and SAR Auto-focusing in a Sensor Fusion Framework. Thesis No. 1464, 2011.

K. Granström: Loop detection and extended target tracking using laser data. Thesis No. 1465, 2011.

J. Callmer: Topics in Localization and Mapping. Thesis No. 1489, 2011.

F. Lindsten: Rao-Blackwellised particle methods for inference and identification. Thesis No. 1480, 2011.

M. Skoglund: Visual Inertial Navigation and Calibration. Thesis No. 1500, 2011.

S. Khoshfetrat Pakazad: Topics in Robustness Analysis. Thesis No. 1512, 2011.

P. Axelsson: On Sensor Fusion Applied to Industrial Manipulators. Thesis No. 1511, 2011.

A. Carvalho Bittencourt: On Modeling and Diagnosis of Friction and Wear in Industrial Robots. Thesis No. 1516, 2012.

P. Rosander: Averaging level control in the presence of frequent inlet flow upsets. Thesis No. 1527, 2012.

N. Wahlström: Localization using Magnetometers and Light Sensors. Thesis No. 1581, 2013.

R. Larsson: System Identification of Flight Mechanical Characteristics. Thesis No. 1599, 2013.

Y. Jung: Estimation of Inverse Models Applied to Power Amplifier Predistortion. Thesis No. 1605, 2013.

M. Syldatk: On Calibration of Ground Sensor Networks. Thesis No. 1611, 2013.

M. Roth: Kalman Filters for Nonlinear Systems and Heavy-Tailed Noise. Thesis No. 1613, 2013.

D. Simon: Model Predictive Control in Flight Control Design — Stability and Reference Tracking. Thesis No. 1642, 2014.

J. Dahlin: Sequential Monte Carlo for inference in nonlinear state space models. Thesis No. 1652, 2014.

M. Kok: Probabilistic modeling for positioning applications using inertial sensors. Thesis No. 1656, 2014.

J. Linder: Graybox Modelling of Ships Using Indirect Input Measurements. Thesis No. 1681, 2014.

G. Mathai: Direction of Arrival Estimation of Wideband Acoustic Wavefields in a Passive Sensing Environment. Thesis No. 1721, 2015.

I. Nielsen: On Structure Exploiting Numerical Algorithms for Model Predictive Control. Thesis No. 1727, 2015.

C. Veibäck: Tracking of Animals Using Airborne Cameras. Thesis No. 1761, 2016.

N. Evestedt: Sampling Based Motion Planning for Heavy Duty Autonomous Vehicles. Thesis No. 1762, 2016.

H. Nyqvist: On Pose Estimation in Room-Scaled Environments. Thesis No. 1765, 2016.

Y. Zhao: Position Estimation in Uncertain Radio Environments and Trajectory Learning. Thesis No. 1772, 2017.

P. Kasebzadeh: Parameter Estimation for Mobile Positioning Applications. Thesis No. 1786, 2017.

K. Radnosrati: On Timing-Based Localization in Cellular Radio Networks. Thesis No. 1808, 2018.

G. Lindmark: Methods and Algorithms for Control Input Placement in Complex Networks. Thesis No. 1814, 2018.

M. Lindfors: Frequency Tracking for Speed Estimation. Thesis No. 1815, 2018.

D. Ho: Some results on closed-loop identification of quadcopters. Thesis No. 1826, 2018.

O. Ljungqvist: On motion planning and control for truck and trailer systems. Thesis No. 1832, 2019.

P. Boström-Rost: On Informative Path Planning for Tracking and Surveillance. Thesis No. 1838, 2019.

K. Bergman: On Motion Planning Using Numerical Optimal Control. Thesis No. 1843, 2019.

M. Klingspor: Low-rank optimization in system identification. Thesis No. 1855, 2019.

A. Bergström: Timing-Based Localization using Multipath Information. Thesis No. 1867, 2019.

F. Ljungberg: Estimation of Nonlinear Greybox Models for Marine Applications. Thesis No. 1880, 2020.

E. Hedberg: Control, Models and Industrial Manipulators. Thesis No. 1894, 2020.

R. Forsling: Decentralized Estimation Using Conservative Information Extraction. Thesis No. 1897, 2020.

D. Arnström: On Complexity Certification of Active-Set QP Methods with Applications to Linear MPC. Thesis No. 1901, 2021.

FACULTY OF SCIENCE AND ENGINEERING

Linköping Studies in Science and Technology, Licentiate Thesis No. 1902
Department of Electrical Engineering

Linköping University
SE-581 83 Linköping, Sweden

www.liu.se