Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 2318

Generalization under Model Mismatch and Distributed Learning

MARTIN HELLKVIST

ACTA UNIVERSITATIS
UPSALIENSIS
2023

ISSN 1651-6214
ISBN 978-91-513-1918-6
urn:nbn:se:uu:diva-513254

UPPSALA
UNIVERSITET

Dissertation presented at Uppsala University to be publicly examined in Polhemsalen, Ångströmlaboratoriet, Lägerhyddsvägen 1, Uppsala, Friday, 24 November 2023 at 09:00 for the degree of Doctor of Philosophy. The examination will be conducted in English. Faculty examiner: Associate Professor Martin Jaggi (EPFL, Machine Learning and Optimization Laboratory).

Abstract

Hellkvist, M. 2023. Generalization under Model Mismatch and Distributed Learning. *Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology* 2318. 50 pp. Uppsala: Acta Universitatis Upsaliensis. ISBN 978-91-513-1918-6.

Machine learning models are typically configured by minimizing the training error over a given training dataset. On the other hand, the main objective is to obtain models that can generalize, i.e., perform well on data unseen during training. A fundamental challenge is that a low training error does not guarantee a low generalization error. While the classical wisdom from statistical learning theory states that models which perfectly fit the training data are unlikely to generalize, recent empirical results show that massively overparameterized models can defy this notion, leading to double-descent curves. This thesis investigates this phenomenon and characterizes the generalization performance of linear models across various scenarios.

The first part of this thesis focuses on characterizing the generalization performance as a function of the level of model mismatch between the patterns in the data and the assumed model. Model mismatch is undesirable but often present in practice. We reveal the trade-offs between the number of samples, the number of features, and the possibly incorrect statistical assumptions on the assumed model. We show that fake features, i.e., features present in the model but not in the data, can significantly improve the generalization performance even though they are not correlated with the features in the underlying system.

The second part of this thesis focuses on generalization under distributed learning. We focus on the scenario where the model parameters are distributed over a network of learners. We consider two settings: the single-task setting, where the network learns a single task, and the continual learning setting, where the network learns multiple tasks sequentially. The obtained results show that for a wide family of feature probability distributions, the generalization performance heavily depends on how the model parameters are partitioned over the network. In particular, if the number of parameters per learner and the number of training samples are close, then the generalization error may be extremely large compared to other data partitioning schemes. For continual learning, our results quantify how the most favorable network structure for the generalization performance depends on the task similarities as well as the number of tasks.

Keywords: Machine learning, Signal processing, Generalization error, Training error, Double-descent, Double descent, Distributed learning, Distributed optimization, Learning over networks, Model mismatch, Model misspecification, Fake features, Missing features, linear regression, regularization

Martin Hellkvist, Department of Electrical Engineering, Signals and Systems, Box 65, Uppsala University, SE-751 03 Uppsala, Sweden.

ISSN 1651-6214
ISBN 978-91-513-1918-6
URN urn:nbn:se:uu:diva-513254 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-513254)

Till min fru, Frida

Sammanfattning

Människor har länge använt sig av maskiner för att underlätta utföranden av slitsamma, repetitiva och monotona uppgifter. Uppfinningar såsom tryckpressen, spinnmaskinen och mjölkmaskinen är exempel på tidiga lösningar som haft stor betydelse på vår samhällsutveckling. Datoriseringen under 1900-talets andra hälft har lett till elektroniska lösningar för post, bokföring, ordbehandling, databaser och styrteknik. Men vissa uppgifter är så komplicerade eller resurskrävande att de inte kan programmeras för hand, och här kommer maskininlärning in i bilden.

Maskininlärning handlar om metoder som använder data för att träna datorer så att de kan sätta upp regler som de sedan kan använda för att fatta beslut eller att göra prognoser, utan att ha programmerats med förutbestämda regler för hur uppgiften i fråga skall lösas. Inom maskininlärning talas det ofta om *modeller*: matematiska funktioner med justerbara parametrar. Modellen tränas genom att dess parametrar justeras så att uppgiften utförs så bra som möjligt givet en uppsättning exempel, kallade träningsdata. När modellen har tränats så kan den användas till att utföra uppgiften på ny data, som den inte har sett under träningsfasen. Begreppet *generalisering* beskriver modellens förmåga att utföra den aktuella uppgiften på nya data. Att modellen kan generalisera väl är huvudmålet med maskininlärning. Vidare använder vi begreppet *träningsfel* som ett prestandamått baserat på träningsdatan och begreppet *generaliseringsfel* för hur väl modellen presterar på nya data, där ett litet fel betyder bra prestanda.

En central utmaning inom maskininlärning är att ett litet träningsfel inte nödvändigtvis medför ett litet generaliseringsfel. Traditionella riktlinjer rekommenderar till och med att modeller utan, eller med väldigt litet, träningsfel bör undvikas då de är överanpassade till träningsdatan. Å andra sidan så har empiriska studier nyligen visat att modeller med ett stort antal parametrar, flera gånger större än antalet träningsdata, kan uppnå små generaliseringfel trots att de inte har något träningsfel. I och med dessa diskrepanser mellan träning och generalisering, samt mellan klassisk teori och nya empiriska resultat, så är mycket forskning inriktad på att karaktärisera generaliseringsfelet för olika maskininlärningsmodeller.

Denna avhandling behandlar karaktäriseringen av generaliseringsfelet från två olika problemaspekter: generaliseringsfelet under modelleringsfel och generaliseringsfelet i distribuerad maskininlärning.

Modelleringsfel syftar här på avvikelser mellan maskininlärningsmodellens strukur och de mönster som finns i datan. Modelleringsfel är vanligt i praktiken och något man helst vill undvika då det kan försämra modellens prestanda. I

den här avhandlingen presenterar vi ett ramverk under vilket vi studerar effekterna av modelleringsfel på generaliseringsfelet systematiskt. Vi fokuserar på linjära modeller, alltså modeller vars beräknade uppskattningar är linjära funktioner av modellparametrarna. Linjära modeller är väldigt användbara i sig och leder ofta till en lättolkad matematisk analys, jämfört med olinjära modeller. Våra resultat beskriver generaliseringsfelet som funktion av antalet träningsdata och antalet justerbara parametrar. Vi drar slutsatsen att generaliseringsfelet kan minska om modelleringsfelet ökar. Detta verkar överraskande, men vi visar att den här effekten av modelleringsfel liknar den effekt man får av vedertagna verktyg som används för att minska generaliseringsfelet.

Avhandlingens andra del undersöker generaliseringsfelet inom distribuerad maskininlärning. Distribuerad maskininlärning syftar till metoder där maskininlärning utförs av ett nätverk av datorer eller beräkningsnoder. Användningen av distribuerad maskininlärning är fördelaktig av flera skäl, till exempel, i situationer där modellens träning kräver omfattande beräkningsresurser. I sådana fall kan distribuerad maskininlärning använda beräkningskraften från flera noder för att minska belastningen per nod. I andra fall kan data vara av känslig natur, vilket innebär att data inte får delas över nätverket av säkerhetsskäl. Då kan distribuerad maskininlärning användas för att träna en modell som drar nytta av all tillgänglig data i nätverket, utan dela själva datan mellan noderna. Det finns ett stort intresse för sådana metoder, men studier om distribuerad inlärning tenderar att fokusera mer på hur träningsfelet utvecklas mellan olika kommunikationsrundor, snarare än att karaktärisera generaliseringsfelet. I denna avhandling studerar vi en etablerad metod för distribuerad inlärning vid namn CoCoA och karaktäriserar dess generaliseringsfel. Våra resultat visar att fördelningen, eller partitioneringen, av träningsdata mellan noderna i nätverket har en betydande inverkan på generaliseringsfelet. Mer specifikt så visar vi att man bör undvika att antalet parametrar som hänför sig till varje nod är nära antalet tillgängliga träningsdata. Detta är nödvändigt för att förhindra att individuella noder överanpassar sin lokala modell till träningsdatan, vilket skulle leda till stora generaliseringsfel.

List of papers

This thesis is based on the following papers, which are referred to in the text by their Roman numerals.

I M. Hellkvist and A. Özçelikkale, “Model mismatch trade-offs in LMMSE estimation,” in *29th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 2045–2049.

II M. Hellkvist, A. Özçelikkale, and A. Ahlén, “Estimation under model misspecification with fake features,” *IEEE Transactions on Signal Processing*, vol. 71, pp. 47–60, 2023.

III M. Hellkvist, A. Özçelikkale, and A. Ahlén, “Regularization trade-offs with fake features,” *Accepted to the 31st European Signal Processing Conference (EUSIPCO)*, 2023.

IV M. Hellkvist, A. Özçelikkale, and A. Ahlén, “Generalization error for linear regression under distributed learning,” in *IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2020, pp. 1–5.

V M. Hellkvist, A. Özçelikkale, and A. Ahlén, “Linear regression with distributed learning: A generalization error perspective,” *IEEE Transactions on Signal Processing*, vol. 69, pp. 5479–5495, 2021.

VI M. Hellkvist, A. Özçelikkale, and A. Ahlén, “Continual learning with distributed optimization: Does CoCoA forget?” *Conference submission*, 2022.

VII M. Hellkvist, A. Özçelikkale, and A. Ahlén, “Distributed continual learning with CoCoA,” *Journal submission*, 2023.

Contents

Acknowledgements

First and foremost, I would like to express the utmost gratitude to my supervisor Ayça Özçelikkale. Thank you for accepting me as your PhD student. Your unwavering support and encouragement have been invaluable, and you have been an exceptional supervisor.

I would also like to acknowledge the support and guidance of my co-supervisor, Anders Ahlén. Your expertise and insights throughout the years have been extremely valuable.

A special thanks goes out to Mikael Sternad, who encouraged me to pursue this PhD, and for always being in the mood for deep conversations about any given topic from least-squares to historical events.

I am thankful for all my wonderful colleagues which I've had in our corridor throughout the years. Joachim, thank you for your guidance and reassurance during the start of my PhD. Sri and Sanja, you are simply brilliant, thanks for all the laughs.

Jag vill tacka alla som jag haft äran att kalla mina vänner genom åren. Jocke, Filippa och Fredrik, ni som varit med från första start: jag hade inte kommit långt utan er. Erik, Josefine, August och Bylund, tack för fest, fika, tentaplugg och utflykt. Ni gjorde Uppsala till mitt hem, jag hade inte stannat länge om det inte vore för er.

Jag är djupt tacksam för mina fantastiska föräldrar och mina älskade syskon, Anders och Birgitta, Ida och Emma. Ni har alltid funnits där för mig och uppmuntrat mig till att tänka självständigt och kritiskt. Ert ovillkorliga stöd har betytt allt för mig och jag är oändligt tacksam för det.

Mitt sista tack går ut till min fantastiska familj, Frida och Alfred, ni är det bästa som har hänt mig. Frida, din kärlek, ditt tålamod, dina perspektiv och värderingar har format mig under de tretton år vi redan spenderat tillsammans. Alfred, jag visste inte att bebisar kunde vara så gulliga. Tack för att du förgyller varje dag och lär mig nya saker om mig själv.

Notation

The following examples illustrate the notation used in the thesis.

a	scalar
$\boldsymbol{a}$	vector
$\boldsymbol{A}$	matrix
$a \in \mathcal{A}$	element a belongs to set $\mathcal{A}$
$a \notin \mathcal{A}$	element a does not belong to set $\mathcal{A}$
$\mathbb{R}$	set of real numbers
$\mathbb{R}^{n \times p}$	set of real n by p matrices
$\mathbb{R}^{n \times 1}$	set of real n-dimensional column vectors
$\mathbb{R}^{1 \times p}$	set of real p-dimensional row vectors
$[a, b]$	set of real numbers on the closed interval from $a \in \mathbb{R}$ to $b \in \mathbb{R}$
$\boldsymbol{A}^\mathsf{T}$	transpose of a matrix $\boldsymbol{A}$
$\boldsymbol{A}^+$	Moore-Penrose pseudoinverse of a matrix $\boldsymbol{A}$
$\hat{x}$	estimate of a scalar parameter x
$\hat{\boldsymbol{x}}$	estimate of a vector of parameters $\boldsymbol{x}$
$\boldsymbol{x}^*$	ground-truth value of a vector of parameters $\boldsymbol{x}$
$\lVert\boldsymbol{a}\rVert_p$	p-norm of a vector $\boldsymbol{a}$, $p \geq 1, p \in \mathbb{R}$
$\lVert\boldsymbol{A}\rVert_p$	matrix norm of a matrix $\boldsymbol{A}$, induced by the vector p-norm
$x \sim P$	P is the probability density function of x
$h : \mathcal{A} \rightarrow \mathcal{B}$	function from set $\mathcal{A}$ to set $\mathcal{B}$
$\mathbb{E}[\cdot]$	expectation operator
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K})$	Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{K}$

Abbreviations

i.i.d.	Independent and identically distributed
LMMSE	Linear Minimum Mean Squared Error
MSE	Mean Squared Error

1. Introduction

Machine learning refers to processes where data-driven methods are used to train computers to identify patterns, make predictions, or take decisions with limited human intervention. This can be done by formulating a model, represented by a mathematical function, and training it. Training refers to finding a model configuration which minimizes the training error. This is typically done by using optimization methods. The training error is a performance measure evaluated on a given set of training data, which consists of examples of data, that is assumed to have some pattern in it. While the training is performed by finding a model which returns a low training error, a core objective in machine learning is for the trained model to generalize. Generalization here refers to the ability to perform well on new and unseen examples of data, which were not included in the training data, but has the same kind of pattern as the examples in the training data.

Supervised learning is a branch of machine learning where the model is trained to output a corresponding desired value when presented with an example of input data. The model is trained on pairs of input data and corresponding desired output data. If the training is successful, then the model can correctly produce, or predict, the output when presented with a new example of input data. For example, consider the problem of training a model to distinguish between photographs of cats and dogs. After a training dataset of input-output pairs has been collected, i.e., a set of photos and the corresponding labels, "cat" or "dog", the model is trained on these examples. The trained model is said to generalize well if it, when presented with new photos of cats or dogs which it did not see during training, is able to accurately label each new photo with either "cat" or "dog".

Characterizing the generalization properties for various models and algorithms is an active research topic. In particular, there is a line of research which focuses on models which are able to generalize well while interpolating the training data. In these situations, the number of adjustable parameters is so large that the model, with appropriately adjusted parameters, can explain all the training data perfectly, obtaining zero training error. That such models can generalize well is surprising with respect to guidelines from statistical learning textbooks, which suggest that models with zero training error typically generalize poorly. The "double-descent" phenomenon has recently been proposed as a qualitative description of the good generalization properties of large models with zero training error, suggesting that the generalization error may decrease twice as the number of model parameters increase: once before and once after the point of interpolation. Empirical observations of double-descent curves have been made for a wide range of models across various scenarios.

Double-descent curves are often observed under some type of model mismatch. Model mismatch here refers to deviations between the patterns that the model can represent and the actual patterns in the data. To avoid model mismatch is desirable within many learning applications, and considerable efforts have been put into developing methods and theory for this purpose. For example, the large field of system identification has been devoted to the problem of building accurate mathematical models, including aspects of selecting the model structure and estimation of the model parameters. Despite such efforts, it is difficult to eliminate all sources of model mismatch. Hence, model mismatch has been studied in a range of applications, such as in positioning, channel estimation and radar applications, as well as in reinforcement learning problems. A part of this thesis studies the setting where a mismatched model is used for estimation, and in particular how the generalization performance of this model is affected by the degree of model mismatch.

Another area of machine learning in which the generalization performance needs further investigation is that of distributed learning. Distributed learning is a machine learning paradigm in which the machine learning task is distributed over a networked system, such that the computational load is shared among several nodes, as compared to one single node performing the task alone. Despite the vast interest in distributed learning algorithms, most studies focus on convergence performance on the training data rather than the generalization error. This thesis considers distributed learning both in a setting with one single task as well as in a continual learning setting, where multiple tasks arrive sequentially. By characterizing the generalization error for a distributed learning algorithm in both these settings, this thesis gives insights and provides guidelines for distributed learning which cannot be obtained when solely considering the training performance.

1.1 Summary of Papers

Below, a summary of the papers included in the thesis is provided. The first author has been the primary writer of the papers in this thesis, and has derived the analytical results and implemented the numerical verifications and experiments. The co-authors have contributed to the papers by sharing their feedback for the manuscripts, as well as their ideas and professional expertise. The papers have been significantly improved by the joint efforts of all the co-authors.

We now present a few key concepts used throughout the summaries below. A set of training data is denoted by $\{(y_i, \boldsymbol{a}_i)\}_{i=1}^{n}$, where $y_i \in \mathbb{R}$ denotes the scalar valued output data, and $\boldsymbol{a}_i \in \mathbb{R}^{p\times 1}$ denotes the vector valued input data. Learning can be performed directly on the input vectors $\boldsymbol{a}_i$'s, but also on some (typically nonlinear) vector valued transformation $\boldsymbol{\varphi}(\boldsymbol{a}_i) \in \mathbb{R}^{d\times 1}$ on them. We refer to the vectors, $\boldsymbol{a}_i$'s, or the vector valued transformations $\boldsymbol{\varphi}(\boldsymbol{a}_i)$ as feature vectors, depending on the context. In the context of regression analysis, $\boldsymbol{a}_i$'s (or $\boldsymbol{\varphi}(\boldsymbol{a}_i)$, if applicable) are referred to as feature vectors and vectors of regressors interchangeably.

Paper I

M. Hellkvist and A. Özçelikkale, "Model mismatch trade-offs in LMMSE estimation," in *29th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 2045–2049.

The paper studies the linear estimation problem with a mismatched model in a Bayesian estimation setting. Here, a subset of the features of the underlying system is missing in the assumed model. Explicit notions of model mismatch is generally overlooked in studies on the generalization error. The presented problem formulation addresses this gap by considering a systematic notion of model mismatch with missing features. This paper considers standard Gaussian features. The main result is a closed form expression of the generalization error in this model mismatch setting. Additionally, conditions are presented under which the generalization error is either monotonically decreasing or monotonically increasing as a function of the number of missing features. Furthermore, the results show that the generalization error does not depend on the structure of the unknowns' covariance matrix, but only on the signal power of the included and the missing components of the unknowns, respectively.

Paper II

M. Hellkvist, A. Özçelikkale, and A. Ahlén, "Estimation under model misspecification with fake features," *IEEE Transactions on Signal Processing*, vol. 71, pp. 47–60, 2023.

The paper studies the linear estimation problem with a mismatched model in a Bayesian estimation setting, extending the problem formulation of Paper I to account for fake features. As in Paper I, the assumed model is missing features from the underlying system, but now also has a set of fake features included in the model. These fake features are unrelated to the features of the underlying system. The proposed framework enables a systematic study of the effects of model mismatch in terms of fake and missing features on the generalization error. With standard Gaussian features, the paper characterizes the generalization error, and its respective components related to the sets of included, missing, and fake features. The results reveal that the generalization error can be decreased by the inclusion of fake features in the assumed model, even though they are not related to the underlying system.

Paper III
M. Hellkvist, A. Özçelikkale, and A. Ahlén, "Regularization trade-offs with fake features," *Accepted to the 31st European Signal Processing Conference (EUSIPCO)*, 2023.

The paper investigates the implicit regularization due to the inclusion of fake features in the assumed model. In particular, the paper studies the linear estimation problem with ridge regularization when the assumed model has fake and missing features. The main result is a high-probability bound for the generalization error of the ridge regression solution under standard Gaussian features. The results reveal insights to the interplay between the implicit regularization provided by the fake features and the explicit ridge regularization provided by the ridge parameter. In particular, consider an unregularized scenario where a model without fake features has high generalization error due to being close the interpolation threshold. Our results show that one may improve the generalization performance of this model by adding fake features to the model, since fake features can move it away from the interpolation threshold; hence, provide implicit regularization. Adding explicit regularization to the formulation with a suitable ridge parameter can also decrease the generalization error. Our results illustrate the trade-off between these two types of regularization mechanisms. Furthermore, these results quantify how the optimal ridge parameter may depend on the number of fake features.

Paper IV
M. Hellkvist, A. Özçelikkale, and A. Ahlén, "Generalization error for linear regression under distributed learning," in *IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2020, pp. 1–5.

The paper studies the distributed learning problem with a linear model and standard Gaussian features, where the model unknowns are distributed over the network. The paper considers the distributed optimization framework CoCoA. The results quantify the effects of the partitioning of the model on the generalization error. In particular, partitioning schemes which set the number of unknowns in any node close to the number of available training data samples must be avoided, otherwise the generalization error will be extremely large. The main result analytically characterizes the expected generalization error after one round of communication, i.e., one iteration of the algorithm. The paper provides simulations which illustrate that the main take-aways regarding generalization error and partitioning are valid after convergence as well.

Paper V

M. Hellkvist, A. Özçelikkale, and A. Ahlén, "Linear regression with distributed learning: A generalization error perspective," *IEEE Transactions on Signal Processing*, vol. 69, pp. 5479–5495, 2021.

The paper studies the distributed learning of a linear model using CoCoA, where the model unknowns are distributed over the network. This paper extends the scope of the generalization error study in Paper IV from the standard Gaussian distribution to correlated Gaussian distributions and sub-gaussian distributions. Under this wide range of feature distributions, the paper presents high-probability bounds which quantify how the generalization error is affected by the problem dimensions and the partitioning over the network. These results show how the spectral properties of the local covariance matrices affect the bounds. This is in contrast to the centralized setting, as the local covariance matrices are submatrices of the full covariance matrix, which would be the relevant matrix in the centralized setting. This paper also shows that for massively overparameterized models, the algorithm converges in the first iteration, further motivating the relevance of the one-step setting of Paper IV. The paper presents numerical simulations on both synthetic and real-world data which illustrate the analytical results.

Paper VI

M. Hellkvist, A. Özçelikkale, and A. Ahlén, "Continual learning with distributed optimization: Does CoCoA forget?" *Conference submission*, 2022.

This paper considers distributed continual learning with CoCoA, i.e., the learning of multiple, possibly related, tasks in a sequential fashion over a network of nodes. This paper focuses on the setting with linear regression tasks where there is a solution which solves all the tasks simultaneously. The results consist of closed form expressions of the algorithm's solution in the overparameterized setting, as well as numerical simulations of the generalization error, the training error, and the convergence, under different conditions on task repetition and similarity. The numerical simulations illustrate that continual learning can be performed, but also that there are potential pitfalls. For instance, if the total number of samples of all tasks combined matches the number of unknowns, then the solution may diverge, leading to significant performance degradation on previously seen tasks.

Paper VII

M. Hellkvist, A. Özçelikkale, and A. Ahlén, "Distributed continual learning with CoCoA," *Journal submission*, 2023.

This paper considers distributed continual learning with CoCoA on a sequence of linear regression tasks with standard Gaussian features. The main result is an exact analytical characterization of the generalization error. In particular, the paper presents an expression of the generalization error as a function of the number of samples per task, the number of tasks, the number of features per node, the number of nodes, and the similarity between tasks. The main result does not assume any constraints on the relation between the respective optimal solutions of the tasks, hence the presented expression is general in terms of task similarity. This paper quantifies how the generalization performance depends on the network structure and how similar the tasks are. For instance, the results show that the most favorable number of nodes in the network, in terms of the generalization performance, may significantly depend on the number of tasks and the task similarity. The paper also observes that the generalization error is large if there are nodes where the number of parameters is close to the number of samples per task. This finding aligns with the observations made in the single-task setting of Paper V. The paper provides numerical verification of the theoretical analysis, and also illustrates how well CoCoA can perform continual learning on a real-world dataset.

1.2 Outline of Thesis

We categorize the papers in this thesis into two groups: those related to model mismatch, and those related to distributed learning. The remainder of this thesis is organized as follows. Chapter 2 first describes the problem of model mismatch and how it relates to the generalization error and the double-descent phenomenon, and then summarizes the contributions of Paper I – III. Chapter 3 first provides an introduction to distributed learning and summarizes the contributions of Paper IV and V. Then, Chapter 3 presents a description of the problem of continual learning, and summarizes the contributions of Paper VI and VII. Finally, Chapter 4 concludes the thesis and discusses possible directions of future work.

2. Model Mismatch

Supervised learning is a regime of machine learning, where the goal is to find a relationship between input and output data. After a relationship is found, then it may be used to make predictions or decisions when new data comes. There is a wide range of applications of supervised learning problems, including image and speech recognition, email spam detection and recommendation engines. Supervised learning is performed by choosing a *model* and then *training* it based on a set of training data. Here, the training data consists of pairs of input-output data. The training process consists of tuning the model such that when it is presented with examples of inputs from the training data, then it should predict the corresponding output value. If the training is successful, then the model can be used to correctly predict the output values for input data unseen during training. This is known as the model's ability to *generalize*, and it depends on several factors, such as the richness of the set of training data in relation to the data that the model may encounter after training, and the learning capacity of the model in relation to the complexity of the task at hand.

A model is represented by a mathematical function which maps from the input data to the corresponding output data. This function exists in some space of possible functions, i.e., the hypothesis space. Thus, the hypothesis space determines the set of potential model candidates which may be considered during training. The hypothesis space, also often called the model set, is chosen by the user, and may be based on the user's prior knowledge or beliefs about the input-output relationship. In this thesis, we consider *parametric* models, i.e., models represented by a set of adjustable parameters. Here, training the model corresponds to determining the values of the model parameters that are most suitable for the problem at hand, where the suitability is evaluated based on some performance criterion chosen by the user. This is typically done by formulating an optimization problem with the performance criterion as the objective function and the model parameters as the optimization variables, and applying an optimization algorithm.

The user's prior knowledge or beliefs about the input-output relationship on which the model construction is based, can be termed as *assumptions*. In this thesis and the included papers, we use the term *assumed model* to refer to the model structure which is a result of the assumptions made. Ideally, these assumptions are formulated such that the trained model should be able to account for the patterns in the data to be encountered after deployment, which is crucial to the performance of the trained model. In particular, a larger and richer set of training data, together with an appropriately applied optimization algorithm, will only improve the trained model's generalization performance up to a point determined by the learning capacity of the assumed model.

This thesis investigates the effects that the model assumptions have on the generalization performance. In particular, the potential quality of the assumed model is put into perspective of an *underlying system*. We consider scenarios where the input-output pairs come from an underlying system, where the output values are the result of feeding this underlying system with the input data. We focus on the setting where both the assumed model and the underlying system are linear in their parameters. We further assume that there is a *model mismatch*, in that the assumed model does not match the underlying system, even though both the assumed model and the underlying system are linear models. We henceforth refer to the degree of mismatch between the assumed model and the underlying system as *model mismatch*.

Model mismatch frequently occurs in practice and can take a multitude of different forms. For example, model mismatch can arise from a relevant variable left out of the model, an irrelevant variable included in the model, incorrect assumptions on the data statistics, or assuming the wrong family of models, i.e., the hypothesis space does not include the model structure of the underlying system. If one assumes that there is a model mismatch, it is natural to ask what the consequences will be. The papers included in this thesis systematically investigate the effects of model mismatch on the generalization error by formulating a framework around fake and missing features. The presented results characterize the generalization error under model mismatch, and quantify fundamental connections between the two. These results capture what is known as the *double-descent* phenomenon, a characteristic of the generalization error curve which we provide an introduction to in Section 2.1. In particular, model mismatch can induce a double-descent behaviour of the generalization error.

Application scenarios in which model mismatch has been studied specifically, include positioning problems [8], channel estimation [9] and radar applications [10]. Reinforcement learning is sometimes performed in an approximation of the true environment, which can be considered as a model mismatch problem [11]. As model mismatch may be difficult to avoid, robust methods have been proposed for a range of algorithms and settings. For example, the constrained minimum mean squared error estimator [12], the maximum a-posteriori estimator [13], and the generalized difference regret criterion [14], deal with potential model mismatch due to uncertainties in the assumed covariance matrices. In another line of work, the linear regression problem with uncertain regressors due to perturbations, was studied [15, 16]. Furthermore, robustness against missing features, i.e., relevant regressor variables which are not included in the assumed model, was investigated in [17]. Another line of work that considers model mismatch problems is compressive sensing [18], where the signal of interest is often assumed to be sparse with respect to the dictionary used for the signal recovery. Such problems with overcomplete dictionaries may be viewed as a model mismatch-problem as the model contains more features than "necessary". The compressive sensing literature acknowledges the potential problems arising from model misspecification, and has studied the sensitivity and performance bounds based on mismatched basis functions of the dictionaries used [19–21].

The studies in this thesis focus on linear regression models, which are useful in a range of applications within signal processing, statistics and machine learning. Although it has been well studied since its introduction in the early 19thcentury [22], minimum-norm interpolating solutions have recently received increased attention in the context of the double-descent phenomenon [23–28]. This thesis contributes to this line of work under a model mismatch framework.

2.1 Generalization Error and Double-Descent

2.1.1 Overview

Supervised learning is a regime of machine learning, where pairs of input and output data are used to train a model. Consider pairs of training data $(\boldsymbol{a}_1, y_1)$, . . . , $(\boldsymbol{a}_n, y_n)$, where $\boldsymbol{a}_i \in \mathbb{R}^{d\times 1}$ denotes the vectors of input data and $\boldsymbol{y}_i \in \mathbb{R}$ denotes the scalar output data. Using this set of training data, a predictive model (or function) denoted by $h : \mathbb{R}^{d\times 1} \to \mathbb{R}$ is trained. The ultimate goal is to use $h(\cdot)$ to predict the output y corresponding to a new input example $\boldsymbol{a}$, which was not seen during training. For a pair of data $(\boldsymbol{a}, y)$, we refer to the prediction of y using the model $h(\cdot)$ by $\hat{y} = h(\boldsymbol{a})$. If $h(\cdot)$ is a parametric model determined by a vector of parameters $\boldsymbol{x} \in \mathbb{R}^{p\times 1}$, then this is emphasized by explicitly writing $h(\boldsymbol{a}; \boldsymbol{x})$.

By hypothesis, the model $h(\cdot)$ is usually constrained to exist within some family of functions $\mathcal{H}$, which is often called the model set. Typically, $\mathcal{H}$ is formulated by choosing a model architecture such as, for example, linear regression, logistic regression or neural networks. A risk function $R(h) \in \mathbb{R}$ is formulated as a measure of "goodness" of the model. The best possible model is then considered to be the model $h \in \mathcal{H}$ that has the minimum risk. The risk $R(h)$ is defined as the expected loss of $h(\cdot)$, i.e., $R(h) = \mathbb{E}_P\left[\ell\left(h(\boldsymbol{a}),\, y\right)\right] \in \mathbb{R}$, where the loss $\ell\left(h(\boldsymbol{a}),\, y\right)$ measures the model's prediction quality, i.e., the quality of the prediction $\hat{y} = h(\boldsymbol{a})$ with respect to the desired output y. It is a typical underlying assumption that the input-output pairs are related through an unknown joint probability distribution $P(\boldsymbol{a}, y)$, and that the training data $(\boldsymbol{a}_i, y_i)$ are examples drawn from this distribution. Hence, the expectation in $R(h)$ is taken with respect to $P(\boldsymbol{a}, y)$. Two examples of the loss function are the squared loss $\ell\left(\hat{y},\, y\right) = \left(\hat{y} - y\right)^2$ for regression, and the zero-one loss $\ell\left(\hat{y},\, y\right) = \mathbb{1}_{\{\hat{y}\neq y\}}$ for classification, for example.

As the input-output relation, i.e., the joint probability $P(\boldsymbol{a}, y)$, is generally unknown, the risk $R(h)$ cannot be evaluated. Instead, an approximation of the risk is computed based on the available training data, i.e., $R_{\text{emp}}(h) =$

$\frac{1}{n}\sum_{i=1}^{n} \ell\,(h(\boldsymbol{a}_i),\, y_i)$, which is referred to as the *empirical risk* in order to emphasize that it is an empirical approximation of the risk. A supervised learning algorithm is then formulated based on minimizing the empirical risk, $\min_{h\,\in\,\mathcal{H}}\ R_{\text{emp}}(h)$. Thus, the empirical risk corresponds to the *training error*, as it is the error being optimized over during the training of the model. On the other hand, the true risk corresponds to the *generalization error*, as it measures the quality of the model's prediction on data unseen during training.

There is a fundamental discrepancy between achieving low generalization error and minimizing the training error, i.e., between the true risk and the empirical risk. In particular, a low or even near zero training error does not guarantee a low generalization error. This discrepancy poses a critical and central challenge in supervised machine learning. This section continues with discussions of this discrepancy, and in particular how the relationship between the training and the generalization error depends on the model assumptions.

The learning capacity of the possible models in $\mathcal{H}$ is central to the generalization performance of the model after training. For example, if $h(\cdot)$ is a linear regression model with p adjustable parameters, such as $h(\boldsymbol{a};\boldsymbol{x}) = \boldsymbol{a}^\intercal\boldsymbol{x}$ with $\boldsymbol{a} \in \mathbb{R}^{p\times 1}$, $\boldsymbol{x} \in \mathbb{R}^{p\times 1}$, then p is a measure of the learning capacity of $h(\cdot)$. The bias-variance tradeoff from classical statistical learning theory suggests that the capacity should not be too small or too large [29, Section 2.9]. This tradeoff is illustrated in the first half of Figure 2.1. If the capacity is too small, then the model performs poorly, having both a large training error and a large generalization error. The model is then said to be "underfitting" the training data. If the capacity is too large, although still within this first half of the figure, then the model obtains a very small training error but a large generalization error. The model is then said to be "overfitting" the training data. The name of the bias-variance tradeoff refers to how the generalization error can be decomposed into terms of bias and variance, respectively. The bias tends to decrease as the learning capacity increases, while the variance tends to increase as the learning capacity increases, hence the tradeoff.

As the learning capacity increases, the training error tends to decrease, and can eventually reach zero, i.e., $\hat{y}_i = y_i$, $i = 1, \ldots, n$, see the training error curve (dashed yellow) in Figure 2.1. If the training error is zero, the model is said to *interpolate* the training data. The lowest level of learning capacity for which the model can interpolate the training data is referred to as the *interpolation threshold*. High values of generalization error at the interpolation threshold is typically a result of noise in the data being captured by the model, rather than actual patterns in the data. Between the points of zero fit and interpolation, the bias-variance tradeoff suggests that the generalization error follows a U-shaped curve, see the first half of the generalization error curve (solid blue) in Figure 2.1, indicating that there is a sweet-spot in the capacity for which the generalization error is minimized. Hence, a general take-away from the bias-variance tradeoff is that models with zero training error generalize poorly [29].

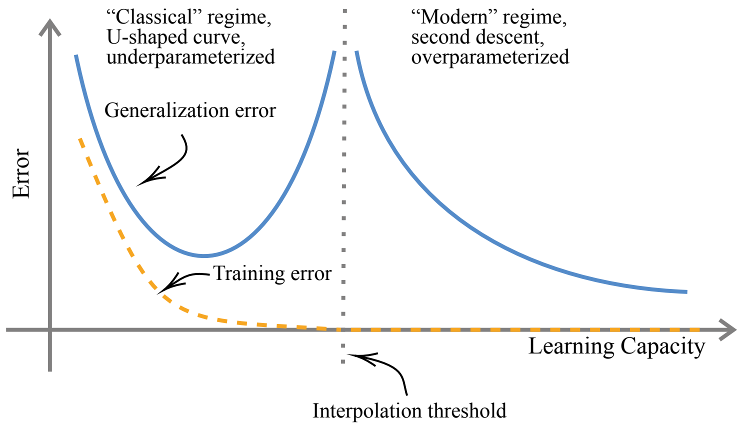

Figure 2.1. Conceptual illustration of the double-descent phenomenon.

However, recent empirical works have shown that highly overparameterized deep neural networks can generalize well while having zero or near zero training error [30], suggesting that the generalization error may descend if the capacity is increased beyond the point of interpolation. The double-descent curve is proposed in [31] as a "reconciliation" between the U-shaped curve from classical theory and the low generalization error observed for these highly overparameterized models. Double-descent curves have been observed for a multitude of models, including random forests and fully connected neural networks [31], as well as convolutional neural networks [32]. Utilizing that double-descent can be observed for linear regression, both for Gaussian and for random feature models, the work in [24] sparked a line of work which studies the phenomenon using linear models.

A conceptual illustration of the double-descent curve is provided in Figure 2.1. The generalization error first descends and then ascends according to the U-shaped curve of the bias-variance tradeoff, and then, when the learning capacity goes beyond the interpolation threshold, the generalization error decreases again, showing the double-descent behaviour.

2.1.2 Example: Polynomial Regression

This section provides an example which illustrates the double-descent phenomenon for a polynomial regression model.

The model is trained on a set of training data $\{(y_i, a_i)\}_{i=1}^{n}$, consisting of $n = 15$ training samples where the inputs $a \in \mathbb{R}$ is drawn uniformly on $a \in [-1, 1]$, and the outputs are generated as $y = 2a + \cos(25a) \in \mathbb{R}$. This underlying input-output relationship between a and y is assumed to be unknown when the assumed model is created in the below.

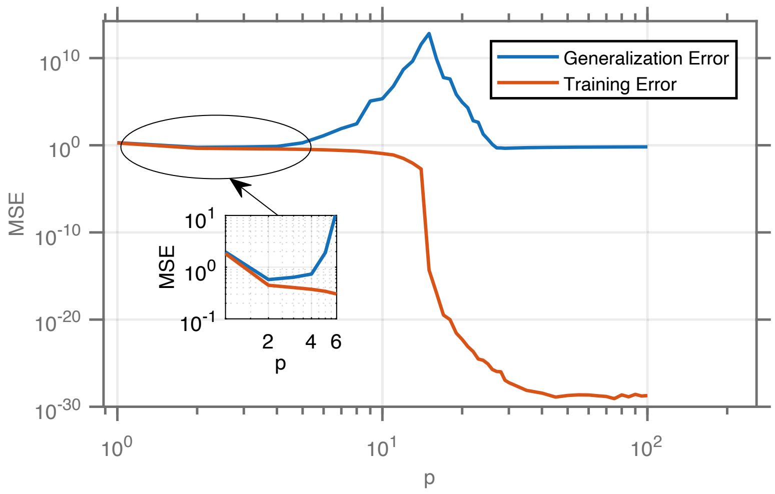

Figure 2.2. The polynomial regression models' training and generalization error versus the model size p.

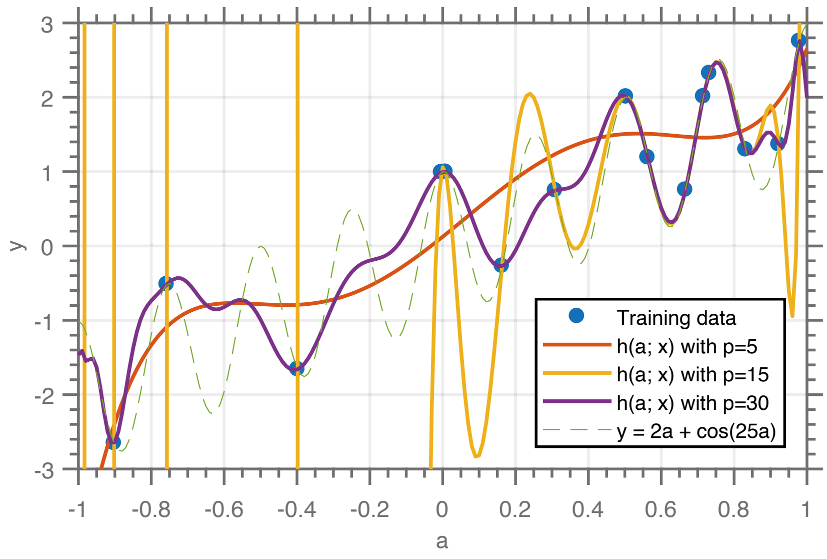

Figure 2.3. Polynomial regression models (solid lines) of different degrees p, trained on the training data (dots) from a sinusoidal model (dashed line).

The polynomial regression model with Legendre polynomials is defined as $h(a; \boldsymbol{x}) = \sum_{j=1}^{p} x_j \varphi_j(a)$, where $\boldsymbol{x} = \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{p\times 1}$ is the vector of model parameters to train, and $\varphi_j(a) \in \mathbb{R}$ denotes the Legendre polynomial of degree $j-1$ evaluated at a. Using the squared loss as the risk, the empirical risk minimization problem is

$$\min_{\boldsymbol{x}\,\in\,\mathbb{R}^{p\times 1}} R_{\text{emp}}\left(h(\cdot\,;\boldsymbol{x})\right) = \min_{\boldsymbol{x}\,\in\,\mathbb{R}^{p\times 1}} \frac{1}{n}\sum_{i=1}^{n}\left(h(a_i\,;\boldsymbol{x}) - y_i\right)^2 \tag{2.1}$$

$$= \min_{\boldsymbol{x}\,\in\,\mathbb{R}^{p\times 1}} \frac{1}{n}\left\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\right\|_2^2, \tag{2.2}$$

where $\boldsymbol{y} = \begin{bmatrix} y_1 & \ldots & y_n \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{n\times 1}$ is the vector of training output samples, and $\boldsymbol{A} \in \mathbb{R}^{n\times p}$ is the feature matrix constructed as

$$\boldsymbol{A} = \begin{bmatrix} \varphi_1(a_1) & \ldots & \varphi_p(a_1) \\ & \vdots & \\ \varphi_1(a_n) & \ldots & \varphi_p(a_n) \end{bmatrix}. \tag{2.3}$$

A solution to this problem is given by $\boldsymbol{x} = \boldsymbol{A}^+\boldsymbol{y}$, where $\boldsymbol{A}^+$ denotes the Moore-Penrose pseudoinverse of $\boldsymbol{A}$.

In Figure 2.2, the generalization error and the training error are plotted versus the number of parameters p in the polynomial regression model $h(\cdot)$. The training error here is the empirical risk $R_{\text{emp}}\left(h(\cdot\,;\boldsymbol{x})\right)$ evaluated on the training dataset with $n = 15$ samples, and the generalization error is evaluated as the average squared loss on the range $a \in [-1, 1]$ uniformly sampled with 0.001 increments.

Figure 2.2 illustrates that the training error decreases as the model size p increases. At $p = n = 15$, the training error comes very close to zero, less than 10^{-18}, perfectly fitting the training data. Thus, the model for $p = n$ is at the interpolation threshold. The figure illustrates that models for $p \geq n$ also interpolate the training data. The generalization error follows the U-shaped curve of the bias-variance tradeoff for $p = 1, \ldots, 15$, and reaches its peak at the interpolation threshold $p = n = 15$. This peak is consistent with the notion of overfitting in the classic bias-variance tradeoff: that models which interpolate the training data are unable to generalize. However, if the model size p is increased further, then the generalization error decreases, even though the model still interpolates the training data. Hence, this figure illustrates the double-descent phenomenon: As the learning capacity increases beyond the interpolation threshold, the generalization error may decrease to the same level as that of models close to the "sweet-spot" of the classical U-shaped curve.

Figure 2.3 provides an additional illustration by plotting the output predictions from three separate models of sizes $p = 5$, $p = 15$ and $p = 30$. Here, the output predictions $\hat{y} = h(a; \boldsymbol{x})$ are plotted along the y-axis, versus the values of a on the x-axis. The training data samples are marked as dots in the figure. The outputs of the sinusoidal system used to generate the data, i.e., $y = 2a + \cos(25a)$, is plotted as a dashed curve. How close each of the models' curves is to the training data represents the training error, and their closeness to that of the sinusoidal system represents the generalization error. Hence, Figure 2.3 illustrates the training and generalization error for three levels of model learning capacity, here measured by p.

The small model with $p = 5$ is relatively close to the true system curve, while it does not exactly fit the training samples. Hence, this model has non-zero training error and small generalization error. This model is close to the sweet-spot on the classical U-shaped curve, see Figure 2.2 for $p = 5$.

The medium size model with $p = 15$ exactly fits all of the training data samples, achieving zero training error. This model is precisely at the interpolation threshold of $p = n = 15$. This model's curve is generally very far from the sinusoidal system's curve, going outside the range of the plot for several values of a, which corresponds to a very poor generalization performance. The curve for this model illustrates the generalization error at the interpolation threshold, see Figure 2.2 for $p = 15$.

The large model with $p = 30$ retains the interpolation property, as it exactly fits the training data. Overall, it is much closer to the sinusoidal system's curve, having good generalization performance on the same level as the model with $p = 5$ parameters. Hence, this model's curve illustrates that the generalization error can go down as the model size increases beyond the interpolation threshold, see Figure 2.2 for $p = 30$.

2.1.3 Discussions

Double-descent curves have recently gained attention after empirical studies have observed that large-scale deep learning models [33] can generalize well even while achieving zero or near-zero training error [30, 32, 34]. The double-descent generalization error curve was proposed in [31] as a way to combine these recent observations with the classical wisdom of the bias-variance trade-off [29]. Double-descent behaviour has since been observed empirically for a range of models, as in [31], where it was empirically observed for a random Fourier features model, a neural network and a random forest model. Analytical studies on double-descent have devoted much attention to linear models [23–26, 35–40]. Investigating the generalization error and the double-descent phenomenon using linear models typically provides more tractable and rigorous mathematical analysis than non-linear counterparts. Linear models with Gaussian features facilitate one of the more straight-forward settings to analyze, and tractable closed form expressions of the generalization error have been presented [24]. Subgaussian feature distributions extend some of the properties of Gaussian distributions, hence they are attractive to study, and generalization error bounds have been presented [23, 26]. The generalization error is also studied under linear models (in the parameters) with nonlinear feature mappings, such as Fourier series [23,24], random Fourier features [38,40] and nonlinear activation functions [25,36], often with the motivation that they facilitate a closer representation of nonlinear neural networks than Gaussian features.

Additional peaks and descents in the generalization error curve have been observed and investigated. The generalization error peak of the double-descent curve typically occurs when then number of samples is close to the number of features, but in [41], it is shown that when the input is transformed with a nonlinear feature mapping, then a peak can also occur when the number of samples is close to the input dimension. Block-correlated features can also induce generalization error peaks, as shown in [35]. Thus, the generalization error curves are heavily affected by the distribution of the features, and it has even been shown that the generalization error can be increased or decreased 'by design' if one can choose the distribution of an additional feature added to the problem [42].

In addition to the above lines of work focusing on characterizing the generalization error curve, another line of work focuses in particular on quantifying necessary conditions for which *benign overfitting* can occur, i.e., good generalization performance achieved at the same time as interpolation of the training data. In [26], it is shown that that the possibility of benign overfitting in linear regression depends on the eigenvalue decay of the feature covariance matrix.

Regularization can be used to avoid overfitting in order to improve the generalization performance [29], hence regularization affects the generalization error curve and the possibilities of achieving benign overfitting. Studies on double-descent and regularization has shown that the generalization error peak at the interpolation threshold can be dampened and mitigated by using explicit regularization [35]. Also implicit regularization, i.e., regularization not explicitly formulated in the optimization problem, affects the double-descent behaviour and the performance in the overparameterized regime, such as provided by weak features [43], asymptotic overparameterization [44], least-squares ensembles [45], and the kernel shape in kernel regression [46].

2.1.4 Model Mismatch and Double-Descent

It is clear that the model mismatch, i.e., the mismatch between the assumed model in relation to the underlying system, affects the achievable generalization error. Hence, the generalization error curve is often studied under model mismatch, and double-descent has been observed and studied under model mismatch types such as suboptimal ridge regularization [35,44], inclusion of asymptotically weak features [43], and missing features [24,25,47].

By formulating a systematic framework for model mismatch, based around differences between the sets of features in the underlying system and those included in the assumed model, this thesis contributes to the above line of work by studying how the generalization error curve and the double-descent behaviour are affected by model mismatch. Note that, rather than trying to correct the model, this thesis investigates the effects of using the mismatched model for estimation. The contributions are summarized in Section 2.2.

2.2 Contributions

In Paper I and II, one of the main contributions is the presented problem formulation, which is discussed in detail in Section 2.2.1. This formulation introduces a systematic notion of model mismatch to the statistical learning problem with fake and missing features as well as general covariance matrices for the unknowns' priors. In particular, Paper I considers the setting with missing features, and without fake features, while Paper II considers both missing features and fake features. Paper III is based on the framework with both fake and missing features, but considers deterministic priors. Section 2.2.2 provides an overview of the contributions of Paper I – III.

2.2.1 Mismatch in Linear Models

We consider data that is generated from the following linear system,

$$\boldsymbol{y} = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} + \boldsymbol{v} = \boldsymbol{A}_S\boldsymbol{x}_S + \boldsymbol{A}_C\boldsymbol{x}_C + \boldsymbol{v}, \tag{2.4}$$

where $\boldsymbol{y} \in \mathbb{R}^{n\times 1}$ is the vector of observations, $\boldsymbol{x}_S \in \mathbb{R}^{p_S\times 1}$ and $\boldsymbol{x}_C \in \mathbb{R}^{p_C\times 1}$ are the unknowns of interest, and $\boldsymbol{v} \in \mathbb{R}^{n\times 1}$ is the vector of noise. The observations are generated from the features in $\boldsymbol{A}_S \in \mathbb{R}^{n\times p_S}$ and $\boldsymbol{A}_C \in \mathbb{R}^{n\times p_C}$. These two matrices together are referred to as the matrix $\tilde{\boldsymbol{A}}$, which is constructed as

$$\tilde{\boldsymbol{A}} = \begin{bmatrix}\boldsymbol{A}_S & \boldsymbol{A}_C\end{bmatrix} \in \mathbb{R}^{n\times\tilde{p}}, \tag{2.5}$$

where $\tilde{p} = p_S + p_C$ denotes the total number of unknowns which generate the vector of observations. The unknowns in $\boldsymbol{x}_S$ and $\boldsymbol{x}_C$ are arranged in the following corresponding fashion,

$$\tilde{\boldsymbol{x}} = \begin{bmatrix}\boldsymbol{x}_S \\ \boldsymbol{x}_C\end{bmatrix} \in \mathbb{R}^{\tilde{p}\times 1}. \tag{2.6}$$

In an ordinary linear regression setting, the training data would typically consist of the observations in $\boldsymbol{y}$ together with $\tilde{\boldsymbol{A}} = \begin{bmatrix}\boldsymbol{A}_S & \boldsymbol{A}_C\end{bmatrix}$, such that the unknowns $\boldsymbol{x}_S$ and $\boldsymbol{x}_C$ could be estimated. However, we instead consider a setting where a matrix $\boldsymbol{A}_F \in \mathbb{R}^{n\times p_F}$ of *fake features* is included in the training data together with the features in $\boldsymbol{A}_S$, and the feature matrix $\boldsymbol{A}_C$ is missing from the training data. The fake features in $\boldsymbol{A}_F$ are unrelated to the observations $\boldsymbol{y}$ and the other features in the underlying system. Hence, we consider a setting where the assumed model used for estimation is formulated based on the assumption that $\boldsymbol{A}_F$ and $\boldsymbol{A}_S$ should be included, and that corresponding unknowns $\boldsymbol{x}_F \in \mathbb{R}^{p_F \times 1}$ and $\boldsymbol{x}_S$ should be estimated, thus the assumed model is written as

$$\boldsymbol{y} = \bar{\boldsymbol{A}}\bar{\boldsymbol{x}} + \bar{\boldsymbol{v}} = \boldsymbol{A}_F\boldsymbol{x}_F + \boldsymbol{A}_S\boldsymbol{x}_S + \bar{\boldsymbol{v}}, \tag{2.7}$$

where the matrix $\bar{\boldsymbol{A}} \in \mathbb{R}^{n \times \bar{p}}$ and the stacked vector $\bar{\boldsymbol{x}} \in \mathbb{R}^{\bar{p} \times 1}$ are constructed as

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A}_F & \boldsymbol{A}_S \end{bmatrix}, \ \bar{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}_F \\ \boldsymbol{x}_S \end{bmatrix} \tag{2.8}$$

such that $\bar{p} = p_F + p_S$. The matrix $\boldsymbol{A}_F$ with fake features, included in the training data, is assumed to be uncorrelated with the data of the underlying system, i.e., $\boldsymbol{y}$, $\boldsymbol{A}_S$, $\boldsymbol{A}_C$ and $\boldsymbol{v}$, as well as the unknowns $\boldsymbol{x}_S$ and $\boldsymbol{x}_C$. We denote the noise vector in the assumed model by $\bar{\boldsymbol{v}} \in \mathbb{R}^{n \times 1}$, to distinguish it from the noise $\boldsymbol{v}$ in the underlying system. The noise in the assumed model is assumed to be zero-mean and its covariance matrix is denoted by $\hat{\boldsymbol{K}}_{\bar{\boldsymbol{v}}} \in \mathbb{R}^{n \times n}$.

To summarize, there are three distinct types of features in the problem, $\boldsymbol{A}_F$, $\boldsymbol{A}_S$ and $\boldsymbol{A}_C$, which we refer to as

- **Fake features**: The features in $\boldsymbol{A}_F$ are irrelevant to the vector of observations $\boldsymbol{y}$, but included in the misspecified model, hence we refer to $\boldsymbol{A}_F$ as *fake features*.
- **Included underlying features**: The features in $\boldsymbol{A}_S$ are relevant to the observations $\boldsymbol{y}$ and they are included in the training data, hence $\boldsymbol{A}_S$ is referred to as *included underlying features*.
- **Missing features**: The features in $\boldsymbol{A}_C$ are relevant to the observations $\boldsymbol{y}$, but not included in the training data, hence they are referred to as *missing features*.

The training data thus consists of $(\boldsymbol{y}, \boldsymbol{A}_F, \boldsymbol{A}_S)$, and the training consists of creating estimates $\hat{\boldsymbol{x}}_F \in \mathbb{R}^{p_F \times 1}$ and $\hat{\boldsymbol{x}}_S \in \mathbb{R}^{p_S \times 1}$ of the unknowns $\boldsymbol{x}_F$ and $\boldsymbol{x}_S$ in the assumed model (2.7).

In order to evaluate the generalization properties of the resulting estimates, we consider test data which are independent and identically distributed (i.i.d.) with the samples of training data. In particular, we consider an observation generated as the training data, i.e.,

$$y = \boldsymbol{a}_S^\mathsf{T}\boldsymbol{x}_S + \boldsymbol{a}_C^\mathsf{T}\boldsymbol{x}_C + v, \tag{2.9}$$

and the corresponding predicted output, computed using the assumed model, i.e.,

$$\hat{y} = \boldsymbol{a}_F^\mathsf{T}\hat{\boldsymbol{x}}_F + \boldsymbol{a}_S^\mathsf{T}\hat{\boldsymbol{x}}_S. \tag{2.10}$$

Here, $\boldsymbol{a}_F \in \mathbb{R}^{p_F \times 1}$, $\boldsymbol{a}_S \in \mathbb{R}^{p_S \times 1}$, and $\boldsymbol{a}_C \in \mathbb{R}^{p_C \times 1}$ are feature vectors i.i.d. with the rows of $\boldsymbol{A}_F$, $\boldsymbol{A}_S$, $\boldsymbol{A}_C$, respectively, and the noise $v \in \mathbb{R}$ is i.i.d. with the entries of the noise vector $\boldsymbol{v}$.

Papers I – III are based on the problem formulation described here. The presented results characterize the generalization error in terms of the prediction error $y - \hat{y}$, as well as the respective errors related to the different unknown vectors, i.e., $\boldsymbol{x}_S - \hat{\boldsymbol{x}}_S$, $\boldsymbol{x}_C - \hat{\boldsymbol{x}}_C$ and $\boldsymbol{x}_F - \hat{\boldsymbol{x}}_F$. We discuss the contributions of these papers in more detail in Section 2.2.2.

2.2.2 Generalization Error under Model Mismatch

Paper I and II

Paper I and II study the problem described in Section 2.2.1 under the Bayesian setting of linear minimum mean squared error (LMMSE) estimation.

We consider a vector of observations $\boldsymbol{y} \in \mathbb{R}^{n\times 1}$ generated from the noisy underlying system in (2.4), i.e., $\boldsymbol{y} = \boldsymbol{A}_S\boldsymbol{x}_S + \boldsymbol{A}_C\boldsymbol{x}_C + \boldsymbol{v}$. The unknowns of the underlying system, $\boldsymbol{x}_S$ and $\boldsymbol{x}_C$, and the noise vector $\boldsymbol{v}$ are zero-mean random vectors with covariance matrices $\boldsymbol{K}_{\boldsymbol{x}_S} \in \mathbb{R}^{p_S\times p_S}$, $\boldsymbol{K}_{\boldsymbol{x}_C} \in \mathbb{R}^{p_C\times p_C}$ and $\boldsymbol{K}_{\boldsymbol{v}} \in \mathbb{R}^{n\times n}$, respectively.

The training dataset consists of the vector $\boldsymbol{y}$, the features $\boldsymbol{A}_S$ from the underlying system and a set of fake features $\boldsymbol{A}_F$. The features $\boldsymbol{A}_C$ are missing. The assumed model is formulated as in (2.7), i.e., $\boldsymbol{y} = \bar{\boldsymbol{A}}\bar{\boldsymbol{x}} + \bar{\boldsymbol{v}} = \boldsymbol{A}_F\boldsymbol{x}_F + \boldsymbol{A}_S\boldsymbol{x}_S + \bar{\boldsymbol{v}}$, where $\bar{\boldsymbol{A}} = \begin{bmatrix}\boldsymbol{A}_F & \boldsymbol{A}_S\end{bmatrix}$ and $\bar{\boldsymbol{x}} = \begin{bmatrix}\boldsymbol{x}_F^\intercal & \boldsymbol{x}_S^\intercal\end{bmatrix}^\intercal$. There is a stochastic prior on the vector of unknowns $\bar{\boldsymbol{x}}$ as well as on the noise $\bar{\boldsymbol{v}}$. In particular, the vector of unknowns $\bar{\boldsymbol{x}}$ and the noise vector $\bar{\boldsymbol{v}}$ are assumed to be uncorrelated and zero-mean, and their assumed covariance matrices are denoted as $\hat{\boldsymbol{K}}_{\bar{\boldsymbol{x}}} = \mathbb{E}_{\bar{\boldsymbol{x}}}[\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^\intercal] \in \mathbb{R}^{\bar{p}\times\bar{p}}$, and $\hat{\boldsymbol{K}}_{\bar{\boldsymbol{v}}} = \mathbb{E}_{\bar{\boldsymbol{v}}}[\bar{\boldsymbol{v}}\bar{\boldsymbol{v}}^\intercal] \in \mathbb{R}^{n\times n}$, respectively. The hat-notation $\hat{\boldsymbol{K}}$ on the covariance matrices is here used to emphasize that these covariance matrices are assumptions made for the model. Note that these assumed covariance matrices are not necessarily equal to the covariance matrices in the underlying system (2.4), e.g., we may have $\hat{\boldsymbol{K}}_{\bar{\boldsymbol{x}}} \neq \boldsymbol{K}_{\bar{\boldsymbol{x}}}$.

Paper I and II consider linear estimation of $\bar{\boldsymbol{x}}$, i.e., the estimate $\hat{\bar{\boldsymbol{x}}}$ of $\bar{\boldsymbol{x}}$ is a linear function of the vector of observations $\boldsymbol{y}$, and can be written in the following form

$$\hat{\bar{\boldsymbol{x}}} = \boldsymbol{W}\boldsymbol{y}, \tag{2.11}$$

where the matrix $\boldsymbol{W} \in \mathbb{R}^{\bar{p}\times n}$ is referred to as the estimator. The LMMSE estimator is the linear estimator which minimizes the mean squared error in the unknowns of the assumed model, and is given by

$$\boldsymbol{W} = \arg \min_{\boldsymbol{W} \in \mathbb{R}^{\bar{p}\times n}} \mathbb{E}_{\bar{\boldsymbol{x}},\boldsymbol{y}} \|\bar{\boldsymbol{x}} - \boldsymbol{W}\boldsymbol{y}\|_2^2 . \tag{2.12}$$

Note that the expectation is taken with respect to the assumed prior distributions on $\bar{\boldsymbol{x}}$ and the assumed model of $\boldsymbol{y}$ in (2.7). With the assumed model and priors, the LMMSE estimator is given by

$$\boldsymbol{W} = \hat{\boldsymbol{K}}_{\bar{\boldsymbol{x}}\boldsymbol{y}}\hat{\boldsymbol{K}}_{\boldsymbol{y}}^{-1}, \tag{2.13}$$

and the corresponding estimates by

$$\hat{\bar{\boldsymbol{x}}} = \begin{bmatrix}\hat{\boldsymbol{x}}_F \\ \hat{\boldsymbol{x}}_S\end{bmatrix} = \hat{\boldsymbol{K}}_{\bar{\boldsymbol{x}}\boldsymbol{y}}\hat{\boldsymbol{K}}_{\boldsymbol{y}}^{-1}\boldsymbol{y}. \tag{2.14}$$

The assumed covariance matrix $\hat{\boldsymbol{K}}_{\boldsymbol{y}} = \mathbb{E}_{\boldsymbol{y}}[\boldsymbol{y}\boldsymbol{y}^{\mathsf{T}}] \in \mathbb{R}^{n\times n}$ and the cross-covariance matrix $\hat{\boldsymbol{K}}_{\bar{\boldsymbol{x}}\boldsymbol{y}} = \mathbb{E}_{\bar{\boldsymbol{x}},\boldsymbol{y}}[\bar{\boldsymbol{x}}\boldsymbol{y}^{\mathsf{T}}] \in \mathbb{R}^{\bar{p}\times n}$ are given by the assumed model and the priors on $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{v}}$, i.e.,

$$\hat{\boldsymbol{K}}_{\bar{\boldsymbol{x}}\boldsymbol{y}} = \mathbb{E}_{\bar{\boldsymbol{x}},\bar{\boldsymbol{v}}}\left[\bar{\boldsymbol{x}}(\bar{\boldsymbol{A}}\bar{\boldsymbol{x}}+\bar{\boldsymbol{v}})^{\mathsf{T}}\right] = \hat{\boldsymbol{K}}_{\bar{\boldsymbol{x}}}\bar{\boldsymbol{A}}^{\mathsf{T}}, \tag{2.15}$$

$$\hat{\boldsymbol{K}}_{\boldsymbol{y}} = \mathbb{E}_{\bar{\boldsymbol{x}},\bar{\boldsymbol{v}}}\left[(\bar{\boldsymbol{A}}\bar{\boldsymbol{x}}+\bar{\boldsymbol{v}})(\bar{\boldsymbol{A}}\bar{\boldsymbol{x}}+\bar{\boldsymbol{v}})^{\mathsf{T}}\right] = \bar{\boldsymbol{A}}\hat{\boldsymbol{K}}_{\bar{\boldsymbol{x}}}\bar{\boldsymbol{A}}^{\mathsf{T}} + \hat{\boldsymbol{K}}_{\bar{\boldsymbol{v}}}. \tag{2.16}$$

In Paper I, the misspecified model only contains the included underlying features $\boldsymbol{A}_S$, hence the features $\boldsymbol{A}_C$ are missing. Thus, the model mismatch considered in Paper I is only due to missing features, as fake features are not considered there. Paper II considers fake and missing features together. Thus, Paper I offers a more focused analysis on the effect of missing features, while Paper II treats fake features and missing features together.

Paper I and II analytically characterize the generalization error for their respective settings. The results give analytical expressions of the generalization error as a function of the problem dimensions, i.e., number of samples n and the respective numbers of included, missing, and fake features, p_S, p_C and p_F. While the power levels $\mathrm{tr}(\boldsymbol{K}_{\boldsymbol{x}_S})$ and $\mathrm{tr}(\boldsymbol{K}_{\boldsymbol{x}_C})$ affect the generalization error, our results show that under isotropic Gaussian features, the respective structures of the covariance matrices $\boldsymbol{K}_{\boldsymbol{x}_S}$ and $\boldsymbol{K}_{\boldsymbol{x}_C}$ do not. The expressions capture the double-descent phenomenon, which here occurs if the number of features in the misspecified model $\bar{p} = p_S + p_F$, i.e., the combined number of included underlying features p_S and fake features p_F, is close to the number of samples in the training data, i.e., if $\bar{p} \approx n$, or equivalently, if $\bar{\boldsymbol{A}} \in \mathbb{R}^{n\times\bar{p}}$ is approximately square. To gain insights into this, consider the following scenario. If the noise level in the assumed model is low, e.g., $\hat{\boldsymbol{K}}_{\bar{\boldsymbol{v}}} = \hat{\sigma}_{\bar{v}}^2\boldsymbol{I}_n$ with $\hat{\sigma}_{\bar{v}}^2$ small, then the large peak in generalization error at $\bar{p} \approx n$ occurs because there is a high probability that there are singular values of $\bar{\boldsymbol{A}}$ which are close to zero, but not exactly zero. Due to these small nonzero singular values, low noise level assumption, and the inversion of the matrix $\hat{\boldsymbol{K}}_{\boldsymbol{y}}$, the generalization error becomes large. Nonetheless, a sufficiently large noise level assumption explicitly regularizes the problem, which dampens or even removes the peak in generalization error.

The results in Paper II show that the model mismatch induced by the presence of the fake features in the assumed model can decrease the error, improving the estimation performance, even though the fake features are uncorrelated with the true features. The results suggest that the performance improvement may be a result of the fake features providing implicit regularization to the problem, rather than additional explanatory power.

Paper III

The problem formulation in Paper III considers a deterministic prior on the unknowns. The estimates $\hat{\boldsymbol{x}}_F$ and $\hat{\boldsymbol{x}}_S$ are found by minimizing the ridgeregularized least-squares cost function,

$$\min_{\hat{\bar{\boldsymbol{x}}},\, \hat{\boldsymbol{y}}=\bar{\boldsymbol{A}}\hat{\bar{\boldsymbol{x}}}} \|\boldsymbol{y}-\hat{\boldsymbol{y}}\|_2^2 + \lambda \left\|\hat{\bar{\boldsymbol{x}}}\right\|_2^2 \tag{2.17}$$

$$= \min_{\hat{\boldsymbol{x}}_F, \hat{\boldsymbol{x}}_S} \|\boldsymbol{y} - \boldsymbol{A}_F\hat{\boldsymbol{x}}_F - \boldsymbol{A}_S\hat{\boldsymbol{x}}_S\|_2^2 + \lambda \|\hat{\boldsymbol{x}}_F\|_2^2 + \lambda \|\hat{\boldsymbol{x}}_S\|_2^2 . \tag{2.18}$$

The solution is given by

$$\hat{\bar{\boldsymbol{x}}} = \begin{bmatrix} \hat{\boldsymbol{x}}_F \\ \hat{\boldsymbol{x}}_S \end{bmatrix} = \left(\bar{\boldsymbol{A}}^\mathsf{T}\bar{\boldsymbol{A}} + \lambda \boldsymbol{I}_{\bar{p}}\right)^{-1} \bar{\boldsymbol{A}}^\mathsf{T}\boldsymbol{y}, \tag{2.19}$$

where $\lambda > 0$ is the ridge regularization parameter. We recall that $\bar{\boldsymbol{A}}$ is constructed as $\bar{\boldsymbol{A}} = \begin{bmatrix}\boldsymbol{A}_F & \boldsymbol{A}_S\end{bmatrix}$, where $\boldsymbol{A}_F$ denotes the fake features. This paper further investigates the implicit regularization provided by fake features, and the trade-offs between this implicit regularization and the explicit ridge regularization provided by λ. The main result of Paper III is a high-probability bound on the generalization error. The bound is guaranteed to hold if the ridge parameter is large enough with respect to the difference between the number of included and fake features $p_S + p_F$ and the number of samples n.

This result gives insight into the trade-off between the ridge parameter and the fake features, which was observed in Paper II. In particular, if the model without fake features is close to the interpolation threshold, then the generalization error will be large if the ridge parameter λ is too small. Similarly as in (2.14), the matrix inversion in (2.19) with small nonzero singular values of $\bar{\boldsymbol{A}}$ and small λ, causes the peak in generalization error at $\bar{p} = p_S + p_F \approx n$. On the other hand, if a large number of fake features is added, so that the model is sufficiently far away from $\bar{p} \approx n$, then the peak in generalization error is avoided regardless of the ridge parameter value λ. Hence, the performance improvement due to the presence of fake features in the assumed model can be interpreted as an implicit regularization effect.

3. Distributed Learning

Machine learning problems can be distributed over a network of computational nodes, such that the computational load is lower per node, compared to the load experienced by a single node performing the task alone. Such machine learning algorithms fall into the paradigm of *distributed learning* algorithms. By sharing the computational load over the network, the distributed learning framework is suitable for large-scale machine learning tasks, while it also facilitates emerging needs of data privacy and security [48, 49]. The increasing demand of large-scale machine learning solutions [50–52] for applications such as edge computing [53, 54], has lead to rapid development of the distributed learning field.

A range of terms are used to distinguish what setting within distributed learning is considered. The following terms categorize different settings based on network topology [55]: *centralized* networks, where one central machine orchestrates the learning and aggregation of the computations performed in the network; *decentralized* networks, where intermediate aggregation is performed; *fully distributed* networks, where nodes are fully independent and aggregation is performed directly between the nodes. The term *federated learning* was introduced to put emphasis on distributed learning methods for centralized systems with mobile and edge device applications [56,57]. Distributed learning methods may also be categorized depending on if the samples of data are being distributed, e.g., as in [58], or if parts of the model are being distributed among the learners, e.g., as in [59, 60].

Distributed learning has been studied from a wide range of perspectives, such as privacy protection [49], constraints which vary over time [61], adaptive clustering of nodes [62], and communication efficiency [63–65]. Other than these problem aspects, literature on distributed learning typically focuses on convergence guarantees of the training error, while there is a lack of theoretical characterizations of the generalization error. Nonetheless, we note that distributed learning on minimum mean squared error estimation frameworks have more specifically studied the generalization error. For example, distributed Kalman filters [59, 66] distributed least-mean squares algorithms [58, 67], affine projection algorithms [68], and distributed linear discriminant analysis [69]. In this line of work it is typically the training data samples that are distributed over the network, rather than parts of the model. In contrast, our work in Paper IV–VII studies the generalization error in the setting with distributed model parameters. This setting can be interpreted as the local models being misspecified with respect to the full model, i.e., there is a model mismatch. The presented results thus reveal how the network structure, which is directly connected to the local models' degrees of mismatch, affects the generalization error.

We consider the distributed optimization algorithm CoCoA [70], which we describe in Section 3.1. This setting falls under the category of centralized networks, as the algorithm relies on global aggregates of the estimates sent by the nodes to a central node. CoCoA is a development from its predecessor in CoCoA-v1 [71] and has been further generalized to CoCoA$^+$ in [72] and CoLa in [73]. CoCoA has been an impactful distributed optimization framework, and has been accredited as a component in the 46-fold improvement of the training time of logistic regression performed by IBM and NVIDIA [74].

3.1 The CoCoA Framework

This section provides an introduction to the distributed optimization framework CoCoA [70]. CoCoA is a general distributed optimization framework for minimization problems which can be mapped to the following form

$$\min_{\hat{\boldsymbol{x}} \in \mathbb{R}^{p\times 1}} f\left(\boldsymbol{A}\hat{\boldsymbol{x}}\right) + g\left(\hat{\boldsymbol{x}}\right), \tag{3.1}$$

where $\hat{\boldsymbol{x}} \in \mathbb{R}^{p\times 1}$ is a parameter vector, $\boldsymbol{A} \in \mathbb{R}^{n\times p}$ is a data matrix. Note that CoCoA can also be run on the dual form of this problem, as described in [70], along with the primal form as included here in (3.1). In this thesis, CoCoA is applied to the ridge regression problem,

$$\min_{\hat{\boldsymbol{x}} \in \mathbb{R}^{p\times 1}} \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}\right\|_2^2 + \frac{\lambda}{2}\left\|\hat{\boldsymbol{x}}\right\|_2^2, \tag{3.2}$$

i.e., (3.1) with

$$f\left(\boldsymbol{A}\hat{\boldsymbol{x}}\right) = \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}\right\|_2^2, \tag{3.3}$$

$$g\left(\hat{\boldsymbol{x}}\right) = \frac{\lambda}{2}\left\|\hat{\boldsymbol{x}}\right\|_2^2, \tag{3.4}$$

where $\boldsymbol{y} = [y_1, \cdots, y_n]^\intercal \in \mathbb{R}^{n\times 1}$ is the vector of outputs, $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n]^\intercal \in \mathbb{R}^{n\times p}$ is the matrix of regressors, and $\lambda \geq 0$ is the regularization parameter. Hence, the training dataset consists of the n input-output pairs $(y_i,\, \boldsymbol{a}_i) \in \mathbb{R} \times \mathbb{R}^{p\times 1}$, $i = 1,\, \ldots,\, n$. We continue this section by describing our implementation of CoCoA for solving (3.2), which is given in Algorithm 1 and visually represented in Figure 3.1.

CoCoA operates in a distributed and iterative fashion over a network of learners. The problem of producing an estimate $\hat{\boldsymbol{x}} \in \mathbb{R}^{p\times 1}$ is distributed over the network such that each node governs an exclusive subset of the entries of the unknowns of interest $\hat{\boldsymbol{x}}$, denoted by $\hat{\boldsymbol{x}}_{[k]} \in \mathbb{R}^{p_k\times 1}$, and the corresponding columns of the regressor matrix $\boldsymbol{A} \in \mathbb{R}^{p\times 1}$, denoted by $\boldsymbol{A}_{[k]} \in \mathbb{R}^{n\times p_k}$. Here,

Algorithm 1: CoCoA implementation for solving the regression problem in equation (3.2).

```
1 Input: Training data $(\boldsymbol{y}, \boldsymbol{A})$, with the feature matrix partitioned as
    $\boldsymbol{A} = [\boldsymbol{A}_{[1]}, \ldots, \boldsymbol{A}_{[K]}]$. Initialize with $\hat{\boldsymbol{x}}^{(0)}$.
2 Compute $\boldsymbol{v}_{[k]}^{(0)}$ corresponding to initialization of $\hat{\boldsymbol{x}}^{(0)}$:
    $\boldsymbol{v}_{[k]}^{(0)} = \bar{\varphi} K \boldsymbol{A}_{[k]} \hat{\boldsymbol{x}}_{[k]}^{(0)}$, $k = 1, \ldots, K$.
3 for $i = 1, \ldots, T_c$ do
4     $\bar{\boldsymbol{v}}^{(i)} = \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{v}_{[k]}^{(i-1)}$
5     for $k \in \{1, 2, \ldots, K\}$ do
6         $\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)} = \left(\sigma' \boldsymbol{A}_{[k]}^{\mathsf{T}} \boldsymbol{A}_{[k]} + \lambda \boldsymbol{I}_{p_k}\right)^{+} \left(\boldsymbol{A}_{[k]}^{\mathsf{T}} \left(\boldsymbol{y} - \bar{\boldsymbol{v}}^{(i)}\right) - \lambda \hat{\boldsymbol{x}}_{[k]}^{(i)}\right)$
7         $\hat{\boldsymbol{x}}_{[k]}^{(i)} = \hat{\boldsymbol{x}}_{[k]}^{(i-1)} + \bar{\varphi} \Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}$
8         $\boldsymbol{v}_{[k]}^{(i)} = \bar{\boldsymbol{v}}^{(i)} + \bar{\varphi} K \boldsymbol{A}_{[k]} \Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}$
9 Output: $\hat{\boldsymbol{x}}^{(T_c)}$
```

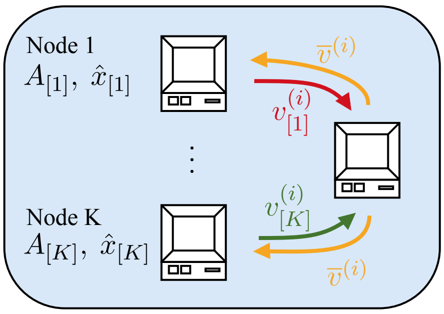

Figure 3.1. Visual representation of distributed learning with CoCoA.

the subscript $[k]$, $k = 1, \ldots, K$, denotes the subset of column indices governed in node k, where K denotes the number of nodes. The number of columns, and hence the number of features, in node k is denoted by p_k, where $\sum_{k=1}^{K} p_k = p$. Without loss of generality, we assume that the columns corresponding to the K partitions come sequentially from the columns of $\boldsymbol{A}$, i.e., $\boldsymbol{A} = \begin{bmatrix}\boldsymbol{A}_{[1]} & \cdots & \boldsymbol{A}_{[K]}\end{bmatrix}$ and $\hat{\boldsymbol{x}} = \begin{bmatrix}\hat{\boldsymbol{x}}_{[1]}^{\mathsf{T}} & \cdots & \hat{\boldsymbol{x}}_{[K]}^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}}$. With this partitioning, we can write

$$\boldsymbol{A}\hat{\boldsymbol{x}} = \begin{bmatrix}\boldsymbol{A}_{[1]} & \cdots & \boldsymbol{A}_{[K]}\end{bmatrix} \begin{bmatrix}\hat{\boldsymbol{x}}_{[1]} \\ \vdots \\ \hat{\boldsymbol{x}}_{[K]}\end{bmatrix} = \sum_{k=1}^{K} \boldsymbol{A}_{[k]}\hat{\boldsymbol{x}}_{[k]}. \tag{3.5}$$

In CoCoA, the nodes iteratively produce estimates for their respective unknown components $\hat{\boldsymbol{x}}_{[k]}$. The iteration number is denoted as (i), $i = 1, \ldots, T_c$, and the corresponding estimates as $\hat{\boldsymbol{x}}_{[k]}^{(i)}$. The nodes collaborate by creating local estimates of the output vector $\boldsymbol{y}$, denoted by $\boldsymbol{v}_{[k]}^{(i)} \in \mathbb{R}^{n\times 1}$, which are collected by a central node which computes and broadcasts the average of them,

$$\bar{\boldsymbol{v}}^{(i)} = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{v}_{[k]}^{(i)} \in \mathbb{R}^{n\times 1}. \tag{3.6}$$

The update that node k makes to its estimate in iteration (i) is denoted by $\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}$, and is computed by solving the following minimization problem

$$\begin{aligned}\min_{\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}} \; & \frac{1}{K} f\left(\bar{\boldsymbol{v}}^{(i)}\right) + \nabla_{\bar{\boldsymbol{v}}^{(i)}} f\left(\bar{\boldsymbol{v}}^{(i)}\right)^{\mathsf{T}} \boldsymbol{A}_{[k]}\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)} \\ & + \frac{\sigma'}{2\tau}\left\|\boldsymbol{A}_{[k]}\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}\right\|_2^2 + g\left(\hat{\boldsymbol{x}}_{[k]}^{(i-1)} + \Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}\right),\end{aligned} \tag{3.7}$$

where the coefficient $\frac{1}{\tau}$ is related to the smoothness of $f(\cdot)$, and the term $\frac{\sigma'}{2\tau}\left\|\boldsymbol{A}_{[k]}\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}\right\|_2^2$ penalizes large changes in $\boldsymbol{v}_{[k]}^{(i)}$, where the coefficient σ' controls the importance of this penalization.

To use CoCoA for solving the ridge regression problem in (3.2), i.e., using $f(\cdot)$ and $g(\cdot)$ from (3.3) and (3.4), corresponds to the following substitutions in (3.7),

$$f\left(\bar{\boldsymbol{v}}^{(i)}\right) = \frac{1}{2}\left\|\boldsymbol{y} - \bar{\boldsymbol{v}}^{(i)}\right\|_2^2, \tag{3.8}$$

$$\nabla_{\bar{\boldsymbol{v}}^{(i)}} f\left(\bar{\boldsymbol{v}}^{(i)}\right) = \bar{\boldsymbol{v}}^{(i)} - \boldsymbol{y}, \tag{3.9}$$

$$g\left(\hat{\boldsymbol{x}}_{[k]}^{(i-1)} + \Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}\right) = \frac{\lambda}{2}\left\|\hat{\boldsymbol{x}}_{[k]}^{(i-1)} + \Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}\right\|_2^2. \tag{3.10}$$

Thus, we have

$$\begin{aligned}\min_{\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}} \frac{1}{2K}\left\|\boldsymbol{y}-\bar{\boldsymbol{v}}^{(i)}\right\|_2^2 &+ \left(\bar{\boldsymbol{v}}^{(i)}-\boldsymbol{y}\right)^{\mathsf{T}}\boldsymbol{A}_{[k]}\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)} \\ &+ \frac{\sigma'}{2}\left\|\boldsymbol{A}_{[k]}\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}\right\|_2^2 + \frac{\lambda}{2}\left\|\hat{\boldsymbol{x}}_{[k]}^{(i-1)}+\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}\right\|_2^2,\end{aligned} \tag{3.11}$$

where we have set $\tau = 1$ as $f(\boldsymbol{A}\hat{\boldsymbol{x}})$ is 1-smooth here, see [70, Def. 3]. This is a convex problem in $\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}$, and is solved by taking the partial derivative, setting it to zero and simplifying, obtaining

$$\left(\sigma'\boldsymbol{A}_{[k]}^{\mathsf{T}}\boldsymbol{A}_{[k]}+\lambda\boldsymbol{I}_{p_k}\right)\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)} = \boldsymbol{A}_{[k]}^{\mathsf{T}}\left(\boldsymbol{y}-\bar{\boldsymbol{v}}^{(i)}\right)-\lambda\hat{\boldsymbol{x}}_{[k]}^{(i)}. \tag{3.12}$$

We solve this set of linear equations by using the Moore-Penrose pseudoinverse, denoted by $(\cdot)^+$, obtaining

$$\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)} = \left(\sigma'\boldsymbol{A}_{[k]}^{\mathsf{T}}\boldsymbol{A}_{[k]}+\lambda\boldsymbol{I}_{p_k}\right)^{+}\left(\boldsymbol{A}_{[k]}^{\mathsf{T}}\left(\boldsymbol{y}-\bar{\boldsymbol{v}}^{(i)}\right)-\lambda\hat{\boldsymbol{x}}_{[k]}^{(i)}\right). \tag{3.13}$$

Note that we use the pseudoinverse here, rather than the standard matrix inverse, as we allow the regularization coefficient to be set to zero, i.e., $\lambda = 0$, for which the existence of the matrix inverse is not guaranteed.

The node then updates its local estimate $\hat{\boldsymbol{x}}_{[k]}^{(i)}$ of $\boldsymbol{x}_{[k]}$ and its local estimate $\boldsymbol{v}_{[k]}^{(i)}$ of the output $\boldsymbol{y}$,

$$\hat{\boldsymbol{x}}_{[k]}^{(i)} = \hat{\boldsymbol{x}}_{[k]}^{(i-1)} + \bar{\varphi}\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}, \tag{3.14}$$

$$\boldsymbol{v}_{[k]}^{(i)} = \bar{\boldsymbol{v}}^{(i)} + \bar{\varphi}K\boldsymbol{A}_{[k]}\Delta\hat{\boldsymbol{x}}_{[k]}^{(i)}, \tag{3.15}$$

where the coefficient $\bar{\varphi}\in(0,1]$ controls how the updates from each node are combined. In particular, $\bar{\varphi}=1$ corresponds to addition of the nodes' solutions, and $\bar{\varphi}=\frac{1}{K}$ corresponds to averaging. The local estimate vectors $\boldsymbol{v}_{[k]}^{(i)}$ are then collected by the central node, which calculates the aggregate $\bar{\boldsymbol{v}}^{(i+1)}$ using (3.6). We have thus described one iteration of CoCoA. As the latest aggregate $\bar{\boldsymbol{v}}^{(i+1)}$ is communicated to the nodes, the algorithm is iterated by the nodes repeating the steps in (3.13)–(3.15) based on the latest aggregate.

3.2 Contributions

In Paper IV – VII, we have studied the distributed learning problem using the CoCoA optimization framework. Paper IV and V consider the standard regression problem where the model is trained on one set of training data, which is referred to as a single task. We describe our contributions to the single-task setting in Section 3.2.1. In contrast, Paper VI and VII study the *continual learning* problem with distributed learning. Here, multiple regression problems, i.e., multiple tasks, are to be solved sequentially. We provide an overview of continual learning in Section 3.2.2 together with a summary of our contributions to the continual learning problem in the distributed learning setting.

3.2.1 Distributed Learning with a Single Task

Contributions

Paper IV and V together reveal that the generalization error in the distributed learning setting can heavily depend on how the data is partitioned over the network. In Paper IV, we study the distributed linear regression problem with isotropic Gaussian regressors. This paper gives closed form expressions of the generalization error after the first iteration of CoCoA. This result together with numerical simulations, shows how the double-descent phenomenon affects the distributed learning setting, showing itself as a peak in the generalization error for certain partitioning schemes. In particular, for a fixed model size, the generalization error peak, related to the double-descent in the centralized setting, occurs as a function of the data partitioning over the network. Such a peak in the error occurs when there is at least one node with the number of unknowns close to the number of samples in the training data. Paper V extends the scope of the regressor distributions to correlated Gaussian distributions, as well as sub-gaussian distributions. The main results in this paper are formulated as high-probability bounds on the generalization error. Paper V also shows that for massively overparameterized models partitioned such that the number of parameters in each node is larger than the number of training data samples, the algorithm converges in the first iteration. This further motivates the relevance of the one-step setting of Paper IV. The results of Paper IV and V together describe how the generalization error varies with the problem and network dimensions, such as the number of samples, the number of unknowns per node and the number of nodes.

3.2.2 Distributed Continual Learning

Overview

Paper VI and VII focus on performing *continual learning* using distributed learning. Continual learning is a paradigm of machine learning in which learning from a sequence of data is considered. Each distinct set of data in the sequence corresponds to a *task*, i.e., a machine learning problem. The aim of continual learning is to train models such that can learn to perform new tasks while they keep performing well on all the previously seen tasks, i.e., not forgetting what was learnt previously.

A task can be represented by any type of learning problem, and has been considered in a range of applications, such as wireless system design [75] and image and gesture classification [76]. Due to variations between tasks in a sequence, the continual learning problem is related to the field of adaptive filtering [77], where the model continuously undergoes adaptation to a changing environment through iterative optimization methods.

A key performance metric in continual learning is that of *forgetting*, which measures the model's performance degradation on previous tasks as it trains on new tasks in the sequence. If a model abruptly loses the information learnt about a previous task, while training on a new task, it is said to exhibit *catastrophic forgetting* [78], which is a fundamental problem to overcome in order to achieve continual learning [79].

Continual learning has gained increasing attention in recent years [80]. Although continual learning has been mainly studied in the centralized learning setting, a limited number of studies on continual learning in distributed learning settings have recently been presented [81, 82]. This thesis contributes to the line of work on continual learning in distributed settings by investigating the possibility of using the distributed optimization framework CoCoA for the continual learning problem.

Contributions

The continual learning setting in this thesis considers tasks which correspond to linear regression problems, where each task has individual sets of training data, consisting of pairs of input and output data. In this setting, the sequence of regression problems should be solved using a common set of model parameters. During training, only the training data from one regression problem is available at a time, and the training is performed sequentially over the linear regression problems, i.e., the tasks.

We now summarize the continual learning setting we consider together with the distributed optimization framework of CoCoA. We use $t = 1, \dots, T$, to denote the indices of the tasks that come in the sequence. For each task, we are given a set of training data denoted by $(\boldsymbol{y}_t, \boldsymbol{A}_t)$. Here, $\boldsymbol{A}_t \in \mathbb{R}^{n_t \times p}$ is a features matrix with i.i.d. standard Gaussian entries, and $\boldsymbol{y}_t \in \mathbb{R}^{n_t \times 1}$ is the corresponding vector of observations generated by the following noisy linear underlying system

$$\boldsymbol{y}_t = \boldsymbol{A}_t \boldsymbol{x}_t^* + \boldsymbol{z}_t, \tag{3.16}$$

where $\boldsymbol{x}_t^* \in \mathbb{R}^{p \times 1}$ is the vector of unknowns for task t, and $\boldsymbol{z}_t \in \mathbb{R}^{n_t \times 1}$ is an unknown noise vector uncorrelated with the features, where the entries of $\boldsymbol{z}_t$ are i.i.d. with zero mean and their variance denoted by σ_t^2. The noise level may vary between between tasks. The matrices $\boldsymbol{A}_t$, $t = 1, \dots, T$ are statistically independent, and so are the noise vectors $\boldsymbol{z}_t$, $t = 1, \dots, T$.

Each task corresponds to finding an estimate $\hat{\boldsymbol{x}} \in \mathbb{R}^{p \times 1}$ which can generalize for new samples from the underlying system in (3.16), i.e., $y_{t,\text{new}} \approx \boldsymbol{a}_{t,\text{new}}^{\mathsf{T}} \hat{\boldsymbol{x}}$,

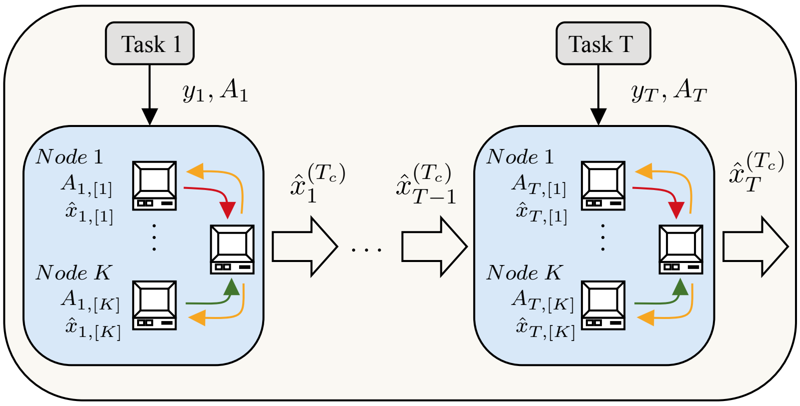

Figure 3.2. Distributed continual learning with CoCoA.

where $y_{t,\text{new}}$ and $\boldsymbol{a}_{t,\text{new}}$ denotes a new sample from task t, unseen during training. For task t, we solve this problem by using CoCoA to minimize the squared loss for the training data of task t, i.e., $\|\boldsymbol{A}_t\hat{\boldsymbol{x}}_t - \boldsymbol{y}_t\|_2^2$. The estimate $\hat{\boldsymbol{x}}$ is updated as the task sequence progresses, and we denote that version of $\hat{\boldsymbol{x}}$ after training on task t by $\hat{\boldsymbol{x}}_t \in \mathbb{R}^{p\times 1}$. We use the notation $\boldsymbol{A}_{t,[k]} \in \mathbb{R}^{n_t\times p_k}$ and $\hat{\boldsymbol{x}}_{t,[k]} \in \mathbb{R}^{p_k\times 1}$ to denote the partitions of task t's feature matrix and the corresponding parameter estimates during the training of task t, where $k = 1, \dots, K$, denotes the indices of the K nodes in the network. Because CoCoA is an iterative algorithm, we write $\hat{\boldsymbol{x}}_t^{(i)} \in \mathbb{R}^{p\times 1}$ to denote the estimate in the i^{th} CoCoA iteration of training on task t, where $i = 1, \dots, T_c$. The final estimate for task t, i.e., $\hat{\boldsymbol{x}}_t^{(T_c)}$, is used as initialization for the training on task $t+1$. The resulting algorithm is illustrated in Figure 3.2. The generalization error is measured over the whole sequence of tasks, and for any estimate $\hat{\boldsymbol{x}}$, it is given by

$$\frac{1}{T}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{y_{t,\text{new}},\boldsymbol{a}_{t,\text{new}}}\left[\left(\boldsymbol{a}_{t,\text{new}}^{\mathsf{T}}\hat{\boldsymbol{x}} - y_{t,\text{new}}\right)^2\right] = \frac{1}{T}\sum_{t=1}^{T}\|\hat{\boldsymbol{x}} - \boldsymbol{x}_t^*\|_2^2 + \sigma_t^2. \quad (3.17)$$

Our analysis focuses on the expectation of the generalization error, over the distribution of training data.

Paper VI focuses on the setting where $\boldsymbol{x}_t^* = \boldsymbol{x}^*, t = 1, \dots, T$, i.e., that the unknowns of the underlying systems are the same for all tasks, and the tasks are noise-free, i.e., $\sigma_t = 0$ for all t. Under these assumptions, Paper VI shows that CoCoA can perform continual learning and achieve reasonable performance, even in the case where the tasks only come once in the sequence. The effect of the number of samples per task and the number of tasks are investigated numerically. This paper also investigates the data fit under repeating sequences of tasks. The results reveal that extreme performance degradation can occur depending on the total number of samples in the task sequence, in relation to the number of unknowns. In particular, we observe that even though the total number of samples $\sum_{t=1}^{T} n_t$ increases with the number of tasks T, a larger T can actually make the performance much worse than for a small T.

Paper VII considers the more general setting where all $\boldsymbol{x}_t^*$ may be distinct vectors, and with $\sigma_t > 0$. The main result of Paper VII is a closed form analytical expression of the generalization error under isotropic Gaussian features and noise. The expression holds for the overparameterized setting where the number of unknowns each node governs is larger than the number of samples in the tasks, i.e., $p_k > n_t$ for all nodes k and tasks t. For other cases where $p_k > n_t$ does not hold, the expression is valid for CoCoA with $T_c = 1$, i.e., if CoCoA is run for one iteration per task. The presented expression is given as a function of the number of samples per task, the number of tasks, the number of unknowns/features per node, the number of nodes, and the similarity between tasks.

In addition to showing that continual learning can be performed using the distributed learning algorithm CoCoA, the results of Paper VI and VII reveal that the network structure may affect the generalization error in a significantly different way than what is reported for single-task learning scenarios. In particular, for a fixed model size, the most favorable network structure in terms of generalization properties depends on the task similarities as well as the number of tasks.

4. Outlook

4.1 Conclusions

In this thesis, we have discussed the works of a series of papers which have investigated the generalization error performance of linear models. We have categorized these papers into two groups, the conclusions of which we discuss below.

In the first group of papers, we proposed a systematic approach to the study of the generalization error by considering linear regression with fake and missing features. We have presented closed form expressions for the generalization error and its decompositions into the respective components corresponding to the included, missing, and fake features, for the unregularized solution. These expressions are practically computable functions of the number of features and the number of samples. The results show that the inclusion of fake features in the model can decrease the generalization error, compared to when there are no fake features in the model. In particular, the error component related to the included underlying features, i.e., the features that are present both in the model and the underlying system, can be smaller when there is a large number of fake features, rather than none. We noted that this performance improvement could be interpreted as implicit regularization provided by the presence of the fake features. To study this effect, we considered our model mismatch framework with fake and missing features in the ridge regression setting, for which we have provided high-probability bounds on the generalization error. Our results show that a performance improvement as a result of an increased number of features does not necessarily imply that these features contain explanatory power: the improvement may be a result of the regularizing effect of the fake features.

The second group of papers considers linear regression in a distributed learning setting. We first considered the setting with one single task, i.e., where the model parameters are estimated using one single set of training data. This setting was then extended to that of continual learning, where sets of training data corresponding to different tasks are presented to the model in a sequential fashion.

In the distributed learning setting, we have characterized the generalization error as a function of how the data is partitioned over the network. Here, the number of features governed per node is a central parameter to our analysis. The setting where each node is overparameterized has been of particular interest here, as it corresponds to the learning of a large-scale model. Under isotropic Gaussian features and overparameterized nodes, we have presented closed form expressions for the generalization error, which are valid after the distributed algorithm's convergence, for both the continual learning setting and the single-task setting. In the single-task setting, in addition to the standard Gaussian feature distribution, we have considered correlated Gaussian distributions and sub-gaussian distributions. We have presented high-probability bounds on the generalization error for these settings. The expressions for the continual learning setting show how the generalization error varies with the task similarities, as well as the problem dimensions of the individual tasks. In particular, the most favorable network size depends on the level of task similarity as well as the number of tasks.

Our findings have highlighted a relationship between the training and generalization error in distributed learning that is generally overlooked. In particular, the partitioning of data over the network can dramatically increase the gap between the training and the generalization error, compared to when the same problem is solved by one single node. Our results have provided guidelines on how to partition the model over the network in order to avoid potential pitfalls. In particular, the results show that one should avoid partitioning the data such that the number of features in any node is close to the number of samples of the task. The local subproblem of a node which violates this rule will be close to the interpolation threshold, which amplifies the gap between training and generalization error, by the same mechanisms which give the double-descent curve its distinctive peak at the interpolation threshold. We have discussed that regularization may decrease the error gap in such scenarios. However, the regularization should be chosen carefully as the convergence may be slow if the regularization is too weak.

4.2 Future Work

The recent successes of massively overparameterized deep neural networks defy the conventional wisdom of statistical learning textbooks. Answering the question of what makes these models generalize well is an important line of work as it could lead to more reliable model design. Despite recent efforts, the generalization properties of neural networks are not well-understood, and the low generalization error of large nonlinear models needs further analytical investigation. This thesis has focused on linear models, which have received significant attention in the literature on overparameterized models. A natural extension of our work to nonlinear scenarios is the investigation of the setting where the data and/or the assumed model contains nonlinear input-output relationships which are not present in the other one.

This thesis has focused on the setting where all nodes have perfect communication channels and use the same local solution strategies with perfect accuracy. On the other hand, heterogeneous networks, where nodes have different communication qualities and different non-perfect local solvers, are often encountered in practice. Hence, it is natural to ask what can happen if we have noise or dropouts in the communication, so that the central node cannot receive updates from the nodes reliably, or what if the computational resources at the nodes are limited, such that the local updates are suboptimal. These practical aspects are expected to affect the solution and the associated generalization error. Hence, extending our analysis to include these aspects of distributed learning is a natural line of future work.

In order to make algorithms adaptive to time-varying real-world environments, developing methods of continual learning is considered an important step forward within the field of machine learning and artificial intelligence. Continual learning draws inspiration from how humans and animals are able to perform an increasing number of tasks over time. In distributed learning, the nodes learn from each other via communication over the network. This may be compared to how humans learn from interacting and communicating with others. Additionally, continual learning applications may also benefit from inherent advantages offered by distributed learning frameworks, such as computational scalability and privacy protection. Hence, the potential of using distributed learning in achieving continual learning poses a natural line of research, which has been explored in this thesis.

We have studied the distributed continual learning setting where the nodes collaborate to solve the same task, and where that task changes over time. Studying the setting where the nodes are to perform continual learning on a local sequence of tasks, where the tasks are related to the tasks of other nodes, is of interest.

Data privacy is important in various applications, such as scenarios where the nodes represent user devices or healthcare institutions, as the data may not be allowed to be shared across the network. Here, the communicated updates over the network must keep the local data private, while still enabling the nodes to learn a common goal together. While we have focused on the trade-offs of the distribution of the data over the network, focusing on the aspects of privacy protection of distributed continual learning is another practically relevant line of research.

Bibliography

[1] M. Hellkvist and A. Özçelikkale, “Model mismatch trade-offs in LMMSE estimation,” in *29th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 2045–2049.

[2] M. Hellkvist, A. Özçelikkale, and A. Ahlén, “Estimation under model misspecification with fake features,” *IEEE Transactions on Signal Processing*, vol. 71, pp. 47–60, 2023.

[3] M. Hellkvist, A. Özçelikkale, and A. Ahlén, “Regularization trade-offs with fake features,” *Accepted to the 31st European Signal Processing Conference (EUSIPCO)*, 2023.

[4] M. Hellkvist, A. Özçelikkale, and A. Ahlén, “Generalization error for linear regression under distributed learning,” in *IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2020, pp. 1–5.

[5] M. Hellkvist, A. Özçelikkale, and A. Ahlén, “Linear regression with distributed learning: A generalization error perspective,” *IEEE Transactions on Signal Processing*, vol. 69, pp. 5479–5495, 2021.

[6] M. Hellkvist, A. Özçelikkale, and A. Ahlén, “Continual learning with distributed optimization: Does CoCoA forget?” *Conference submission*, 2022.

[7] M. Hellkvist, A. Özçelikkale, and A. Ahlén, “Distributed continual learning with CoCoA,” *Journal submission*, 2023.

[8] M. Li, Y. Li, and Q. Wan, “Performance bound for target localization under model misspecification,” in *IEEE 5th International Conference on Computer and Communications (ICCC)*, 2019, pp. 410–414.

[9] L. T. Thanh, K. Abed-Meraim, and N. L. Trung, “Misspecified Cramer–Rao bounds for blind channel estimation under channel order misspecification,” *IEEE Transactions on Signal Processing*, vol. 69, pp. 5372–5385, 2021.

[10] C. Ren, M. N. El Korso, J. Galy, E. Chaumette *et al.*, “Performance bounds under misspecification model for mimo radar application,” in *23rd European Signal Processing Conference (EUSIPCO)*, 2015, pp. 514–518.

[11] A. Roy, H. Xu, and S. Pokutta, “Reinforcement learning under model mismatch,” *Advances in neural information processing systems*, vol. 30, 2017.

[12] D. Lederman and J. Tabrikian, “Constrained MMSE estimator for distribution mismatch compensation,” in *Fourth IEEE Workshop on Sensor Array and Multichannel Processing, 2006.*, 2006, pp. 439–443.

[13] D. Zachariah, N. Shariati, M. Bengtsson, M. Jansson *et al.*, “Estimation for the linear model with uncertain covariance matrices,” *IEEE Transactions on Signal Processing*, vol. 62, no. 6, pp. 1525–1535, 2014.

[14] R. Mittelman and E. L. Miller, “Robust estimation of a random parameter in a Gaussian linear model with joint eigenvalue and elementwise covariance uncertainties,” *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1001–1011, 2010.

[15] L. El Ghaoui and H. Lebret, “Robust solutions to least-squares problems with uncertain data,” *SIAM Journal on Matrix Analysis and Applications*, vol. 18, no. 4, pp. 1035–1064, 1997.

[16] Y. C. Eldar, A. Ben-Tal, and A. Nemirovski, “Robust mean-squared error estimation in the presence of model uncertainties,” *IEEE Transactions on Signal Processing*, vol. 53, no. 1, pp. 168–181, 2005.

[17] X. Liu, D. Zachariah, and P. Stoica, “Robust prediction when features are missing,” *IEEE Signal Processing Letters*, vol. 27, p. 720–724, 2020.

[18] S. Foucart and H. Rauhut, “A mathematical introduction to compressive sensing,” in *An Invitation to Compressive Sensing*. Springer, New York, 2013, pp. 1–39.

[19] Y. Chi, L. L. Scharf, A. Pezeshki, and A. R. Calderbank, “Sensitivity to basis mismatch in compressed sensing,” *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2182–2195, 2011.

[20] Z. Tan, P. Yang, and A. Nehorai, “Joint sparse recovery method for compressed sensing with structured dictionary mismatches,” *IEEE Transactions on Signal Processing*, vol. 62, no. 19, pp. 4997–5008, 2014.

[21] S. Bernhardt, R. Boyer, S. Marcos, and P. Larzabal, “Compressed sensing with basis mismatch: Performance bounds and sparse-based estimator,” *IEEE Transactions on Signal Processing*, vol. 64, no. 13, pp. 3483–3494, 2016.

[22] S. M. Stigler, “Gauss and the inventions of least squares,” *The Annals of Statistics*, vol. 9(3), pp. 465–474, 1981.

[23] V. Muthukumar, K. Vodrahalli, V. Subramanian, and A. Sahai, “Harmless interpolation of noisy data in regression,” *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 67–83, 2020.

[24] M. Belkin, D. Hsu, and J. Xu, “Two models of double descent for weak features,” *SIAM J. Math. Data Sci.*, vol. 2, no. 4, pp. 1167–1180, 2020.

[25] T. Hastie, A. Montanari, S. Rosset, and R. J. Tibshirani, “Surprises in high-dimensional ridgeless least squares interpolation,” *Annals of statistics*, vol. 50, no. 2, p. 949, 2022.

[26] P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler, “Benign overfitting in linear regression,” *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30 063–30 070, 2020.

[27] M. Belkin, “Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation,” *Acta Numerica*, vol. 30, p. 203–248, 2021.

[28] P. L. Bartlett, A. Montanari, and A. Rakhlin, “Deep learning: a statistical viewpoint,” *Acta Numerica*, vol. 30, p. 87–201, 2021.

[29] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, ser. Springer series in statistics. Springer, 2009.

[30] C. Zhang, S. Bengio, M. Hardt, B. Recht *et al.*, “Understanding deep learning requires rethinking generalization,” *arXiv:1611.03530*, Nov. 2016.

[31] M. Belkin, D. Hsu, S. Ma, and S. Mandal, “Reconciling modern machine-learning practice and the classical bias–variance trade-off,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15 849–15 854, 2019.

[32] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[34] M. Belkin, S. Ma, and S. Mandal, "To understand deep learning we need to understand kernel learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 541–549.

[35] P. Nakkiran, P. Venkat, S. M. Kakade, and T. Ma, "Optimal regularization can mitigate double descent," in *International Conference on Learning Representations*, 2021.

[36] S. d'Ascoli, M. Refinetti, G. Biroli, and F. Krzakala, "Double trouble in double descent: Bias and variance (s) in the lazy regime," in *International Conference on Machine Learning*. PMLR, 2020, pp. 2280–2290.

[37] D. Holzmüller, "On the universality of the double descent peak in ridgeless regression," in *International Conference on Learning Representations*, 2020.

[38] Z. Liao, R. Couillet, and M. W. Mahoney, "A random matrix analysis of random fourier features: beyond the gaussian kernel, a precise phase transition, and the corresponding double descent," *Journal of Statistical Mechanics: Theory and Experiment*, no. 12, p. 124006, 2021.

[39] D. Bosch, A. Panahi, and A. Özcelikkale, "Double descent in feature selection: Revisiting LASSO and basis pursuit," *Inter. Conf. on Machine Learning (ICML) 2021 Workshop on Overparameterization: Pitfalls & Opportunities*, 2021.

[40] D. Bosch, A. Panahi, A. Özçelikkale, and D. P. Dubhashi, "Random features model with general convex regularization: A fine grained analysis with precise asymptotic learning curves," in *International Conference on Artificial Intelligence and Statistics*, vol. 206. PMLR, 2023, pp. 11 371–11 414.

[41] S. d'Ascoli, L. Sagun, and G. Biroli, "Triple descent and the two kinds of overfitting: Where & why do they appear?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 3058–3069, 2020.

[42] L. Chen, Y. Min, M. Belkin, and A. Karbasi, "Multiple descent: Design your own generalization curve," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8898–8912, 2021.

[43] D. Richards, J. Mourtada, and L. Rosasco, "Asymptotics of ridge (less) regression under general source condition," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 3889–3897.

[44] D. Kobak, J. Lomond, and B. Sanchez, "The optimal ridge penalty for real-world high-dimensional data can be zero or negative due to the implicit ridge regularization," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 6863–6878, 2020.

[45] D. LeJeune, H. Javadi, and R. Baraniuk, "The implicit regularization of ordinary least squares ensembles," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 3525–3535.

[46] T. Liang and A. Rakhlin, "Just interpolate: Kernel "ridgeless" regression can generalize," *The Annals of Statistics*, vol. 48, no. 3, pp. 1329–1347, 2020.

[47] L. Breiman and D. Freedman, "How many variables should be entered in a regression equation?" *Journal of the American Statistical Association*, vol. 78,

no. 381, pp. 131–136, 1983.

[48] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 46–51, 2020.

[49] X. Wang, H. Ishii, L. Du, P. Cheng *et al.*, "Privacy-preserving distributed machine learning via local randomization and ADMM perturbation," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4226–4241, 2020.

[50] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM review*, vol. 60, no. 2, pp. 223–311, 2018.

[51] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba *et al.*, "Towards federated learning at scale: System design," *Proceedings of machine learning and systems*, vol. 1, pp. 374–388, 2019.

[52] J. Dean, G. Corrado, R. Monga, K. Chen *et al.*, "Large scale distributed deep networks," *Advances in neural information processing systems*, vol. 25, 2012.

[53] H. Chen, Y. Ye, M. Xiao, M. Skoglund *et al.*, "Coded stochastic ADMM for decentralized consensus optimization with edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5360–5373, 2021.

[54] S. Wang, T. Tuor, T. Salonidis, K. K. Leung *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[55] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg *et al.*, "A survey on distributed machine learning," *ACM computing surveys*, vol. 53, no. 2, pp. 1–33, 2020.

[56] B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[57] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1-2, pp. 1–210, 2021.

[58] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.

[59] U. A. Khan and J. M. F. Moura, "Distributing the Kalman filter for large-scale systems," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4919–4935, 2008.

[60] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[61] S. Paternain, S. Lee, M. M. Zavlanos, and A. Ribeiro, "Distributed constrained online learning," *IEEE Transactions on Signal Processing*, vol. 68, pp. 3486–3499, 2020.

[62] X. Zhao and A. H. Sayed, "Distributed clustering and learning over networks," *IEEE Transactions on Signal Processing*, vol. 63, no. 13, pp. 3285–3300, 2015.

[63] D. Alistarh, D. Grubic, J. Li, R. Tomioka *et al.*, "QSGD: Communication-efficient SGD via gradient quantization and encoding," *Advances in neural information processing systems*, vol. 30, 2017.

[64] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff

between computation and communication in distributed computing," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109–128, 2017.

[65] S. Magnússon, C. Enyioha, N. Li, C. Fischione *et al.*, "Communication complexity of dual decomposition methods for distributed resource allocation optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 4, pp. 717–732, 2018.

[66] H. Zhang, J. M. F. Moura, and B. Krogh, "Dynamic field estimation using wireless sensor networks: Tradeoffs between estimation error and communication cost," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2383–2395, 2009.

[67] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, 2008.

[68] L. Li and J. A. Chambers, "Distributed adaptive estimation based on the APA algorithm over diffusion networks with changing topology," in *15th Workshop on Statistical Signal Processing*. IEEE, 2009, pp. 757–760.

[69] K. R. Varshney, "Generalization error of linear discriminant analysis in spatially-correlated sensor networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 3295–3301, 2012.

[70] V. Smith, S. Forte, C. Ma, M. Takáč *et al.*, "CoCoA: A general framework for communication-efficient distributed optimization," *Journal of Machine Learning Research*, vol. 18, no. 230, pp. 1–49, 2018.

[71] M. Jaggi, V. Smith, M. Takác, J. Terhorst *et al.*, "Communication-efficient distributed dual coordinate ascent," *Advances in neural information processing systems*, vol. 27, 2014.

[72] C. Ma, J. Konečnỳ, M. Jaggi, V. Smith *et al.*, "Distributed optimization with arbitrary local solvers," *Optimization Methods and Software*, vol. 32, no. 4, pp. 813–848, 2017.

[73] L. He, A. Bian, and M. Jaggi, "Cola: Decentralized linear learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[74] "News/algorithms in the wild," https://www.epfl.ch/labs/mlo/, march 2018, accessed: 2023-08-30.

[75] H. Sun, W. Pu, X. Fu, T.-H. Chang *et al.*, "Learning to continuously optimize wireless resource in a dynamic environment: A bilevel optimization perspective," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1900–1917, 2022.

[76] M. De Lange, R. Aljundi, M. Masana, S. Parisot *et al.*, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3366–3385, 2022.

[77] A. H. Sayed, *Adaptive filters*. John Wiley & Sons, 2008.

[78] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[79] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[80] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," *arXiv preprint arXiv:2302.00487*,

2023.

[81] J. Yoon, W. Jeong, G. Lee, E. Yang *et al.*, “Federated continual learning with weighted inter-client transfer,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 073–12 086.

[82] D. Qi, H. Zhao, and S. Li, “Better generative replay for continual federated learning,” *arXiv preprint arXiv:2302.13001*, 2023.

Distribution: publications.uu.se
urn:nbn:se:uu:diva-513254

ACTA UNIVERSITATIS
UPSALIENSIS
2023