Machine Learning for Mass Production and Industrial Engineering

Dissertation
zur Erlangung des Grades eines Doktors
der Naturwissenschaften
der Fakultät für Mathematik und Physik
der Eberhard-Karls-Universität zu Tübingen

vorgelegt von
Jens Tobias Pfingsten
aus Neuss
2007

Tag der mündlichen Prüfung: 01.02.2007

Dekan: Prof. Dr. N. Schopohl

1. Berichterstatter: Prof. Dr. B. Schölkopf

2. Berichterstatter: Ph.D. C. E. Rasmussen

Zusammenfassung

In modernen Entwicklungs- und Fertigungsprozessen spielt die Analyse von Messdaten und Simulationsergebnissen eine herausragende Rolle: Während der Entwicklungsphase eines Produktes gilt es, die Eignung des Designs für die Massenfertigung durch Simulationen und Experimente abzusichern. Zudem stellt eine Vielzahl von Messungen eine gleichbleibend hohe Qualität in der automatisierten Fertigungsumgebung sicher.

Bei steigender Anzahl von Messgrößen und wachsenden Datenmengen stößt die konventionelle, manuelle Datenanalyse an ihre Grenzen. Die vorliegende Arbeit untersucht den Nutzen der Anwendung maschinellen Lernens auf typische Fragestellungen in der Datenauswertung von der Entwicklung eines Produktes bis hin zur Massenproduktion. Die Herstellung integrierter Schaltkreise und mikromechanischer Sensoren auf Siliziumbasis dient als Fallbeispiel. Im Rahmen der Arbeit sind konkrete Lösungen zu einigen relevanten Problemen in der industriellen Anwendung entstanden, bei denen die Ausbeute die zentrale Größe darstellt:

Die parametrische Ausbeute wird durch die Empfindlichkeit eines Designs gegenüber Prozessschwankungen bestimmt. In dieser Arbeit wird ein Konzept zur Auswertung von Simulationsergebnissen in einer statistischen Sensitivitätsanalyse entwickelt, die durch den Einsatz nichtparametrischer Regression mit Gaußprozessen effizient durchgeführt werden kann. Der Ansatz ermöglicht eine neue Methode zur robusten Optimierung, die für rechenaufwändige Simulationen erst durch den vorgestellten Ansatz durchführbar wird.

Gaußprozesse ermöglichen eine effektive Nutzung vorhandener Simulationsergebnisse, machen als statistische Modelle aber zudem eine optimale Planung neuer Simulationen möglich, wodurch die Zahl der nötigen Simulationen signifikant reduziert werden kann. Ein neuer Ansatz für aktives Lernen mit Gaußprozessen wird in dieser Arbeit vorgestellt und experimentell validiert.

Neben statistischen Ausfällen, die in der parametrischen Ausbeute erfasst werden, können in der Fertigung systematische Fehler auftreten. Der Ursprung solcher Probleme kann in komplexen Fertigungsanlagen allerdings nur schwer lokalisiert werden, da physikalische Zusammenhänge kaum nachvollziehbar sind und man mit einer großen Zahl möglicher Ursachen konfrontiert ist. Durch Anwendung von Merkmalsselektion ist es im Rahmen dieser Arbeit gelungen, Daten aus Qualitätskontrolle und Fertigung zu kombinieren und die Fehlerlokalisierung zu automatisieren.

Abstract

The analysis of data from simulations and experiments in the development phase and measurements during mass production plays a crucial role in modern manufacturing: Experiments and simulations are performed during the development phase to ensure the design's fitness for mass production. During production, a large number of measurements in the automated production line controls a stable quality.

As the number of measurements grows, the conventional, largely manual data analysis approaches its limits, and alternative methods are needed. This thesis studies the value of machine learning methods for typical problems faced in data analysis from engineering to mass production. In a case-study, the production of integrated circuits and micro electro-mechanical systems in silicon technology is discussed in detail. A number of approaches to salient problems in industrial application have been developed in the presented work, addressing the yield as the central figure of batch processes in silicon manufacturing:

The parametric yield is governed by a design's robustness against process tolerances. This work develops a framework for doing statistical sensitivity analysis, and robust optimization which accounts for process tolerances. Using nonparametric Gaussian process regression, the sensitivity analysis can be performed efficiently. For computationally demanding simulations a robust optimization is eventually only made feasible through the presented approach.

Being probabilistic models, Gaussian processes allow for an optimal experimental design, thus significantly reducing the number of required simulation runs. A novel approach to active learning for Gaussian process regression is proposed in this thesis, and validated experimentally.

Besides random failures, as captured by the parametric yield, systematic errors in the production can lead to additional losses. It is hard to localize the root cause for previously unseen losses, as physical interrelations can hardly be reconstructed in complex manufacturing facilities, and as there is usually a large number of potential sources for the error. This work shows that, using feature selection, data from quality checks can be combined with data from manufacturing to construct an automated localization mechanism.

Machine Learning for Mass Production and Industrial Engineering

Introduction **11**
Motivation and objective of this thesis 11
A guide through this thesis 13

I. Machine learning for semiconductor manufacturing **15**
1. Semiconductor products & manufacturing 15
1.1. Micro systems and integrated circuits 15
1.2. Semiconductor foundries 16
1.3. The “smart” factory 18
1.4. Applications of machine learning in production 19
2. Computer-aided design 22
2.1. Computer experiments 22
2.2. Robust designs for mass production 23
2.3. Machine learning for design analysis 23

II. Bayesian methods in machine learning **25**
1. Bayesian inference 26
1.1. Probabilistic models 26
1.2. The posterior distribution 27
2. Decision theory and model selection 28
2.1. Model selection 29
2.2. Decision theory 31
3. Gaussian process priors 31
3.1. Gaussian process regression 32
3.2. Covariance functions 33

III. Robust designs for mass production **39**
1. Process tolerances and robust designs 40
1.1. Fluctuations and specifications 40
1.2. Approaches to design analysis 43
2. Sensitivity analysis for design validation 46
2.1. Sensitivity measures 46
2.2. Interpretation and use in practice 48
3. Bayesian Monte Carlo for design analysis and optimization . . . 50
3.1. Monte Carlo methods and classical quadrature 50

3.2. Bayesian Monte Carlo . . . 51
3.3. Robust design optimization . . . 53
4. Experiments . . . 54
4.1. Analytical benchmark problems . . . 54
4.2. Case studies from industrial engineering . . . 57
5. Discussion . . . 61

IV. Active Learning for nonparametric regression **63**
1. Experimental design and active learning . . . 64
1.1. Historical development . . . 64
1.2. From experimental design to active learning . . . 66
1.3. The fundamental drawback of active learning . . . 67
1.4. Bounds for learning rates . . . 68
2. Active learning for GP regression . . . 68
2.1. Information-based objectives . . . 69
2.2. The ML-II approximation . . . 72
2.3. Nonstationary GP priors . . . 73
2.4. The greedy A-optimal scheme for active learning . . . 74
3. Evaluation and use in practice . . . 75
3.1. Artificial benchmark functions . . . 75
3.2. Examples from development . . . 78
4. Discussion . . . 79

V. Feature selection for troubleshooting **81**
1. Data preprocessing . . . 82
1.1. Detection of systematic errors . . . 82
1.2. Features and root causes . . . 84
2. Feature selection . . . 85
2.1. Objective in feature selection . . . 85
2.2. Wrappers, filters, and embedded methods . . . 86
2.3. The troubleshooting approach . . . 90
3. Case study . . . 94
3.1. Datasets . . . 94
3.2. Interpretation of the results . . . 95
4. Discussion . . . 100

Conclusions **101**

Appendix **105**
A. Mean square differentiability . . . 105
B. Bayesian Monte Carlo . . . 106
C. Estimates of entropy and (conditional) mutual information . . . 110

Bibliography **111**

... zu selbständiger wissenschaftlicher Arbeit ...

Laut Promotionsordnung[1] muss die Dissertation

> „... *die Fähigkeit des Bewerbers zu selbständiger wissenschaftlicher Arbeit in einem der in der Fakultät für Physik vertretenen Fachgebiete nachweisen*".

Geht es in einer Dissertation auch um selbstständige wissenschaftliche Arbeit, so ist diese doch kaum zu bewältigen, ohne durch Zusammenarbeit und Diskussionen in einer Arbeitsgruppe immer wieder motiviert und inspiriert zu werden. Ich schätze mich glücklich, dass ich die Möglichkeit hatte, in Mannschaften wie der Abteilung CR/ARY „Mikrosystemtechnik" der Robert Bosch GmbH und der Abteilung „Empirische Inferenz für maschinelles Lernen und Wahrnehmung" am Max Planck Institut für biologische Kybernetik zu arbeiten.

Zunächst gilt mein Dank natürlich Hans-Peter Trah, Stefan Finkbeiner, Reinhard Neul und Matthias Maute dafür, die Arbeit bei der CR/ARY ermöglicht und unterstützt zu haben.
Daniel Herrmann hat das Thema maschinelles Lernen bei der CR/ARY ins Leben gerufen und meine Arbeit bei Bosch betreut, wofür ihm mein besonderer Dank gebührt. Er stand mir bei Bosch immer als Ansprechpartner zur Seite.

Vielen Dank an Bernhard Schölkopf dafür, die Kooperation mit dem Max Planck Institut ermöglicht, mich als Mitglied in seiner Arbeitsgruppe aufgenommen und mich auch fachlich substantiell unterstützt zu haben.
Meine wissenschaftliche Arbeit hat Carl Rasmussen als Betreuer wohl am wesentlichsten beeinflusst. Über meine gesamte Zeit als Doktorand hinweg hat er mich in meinen Projekten bekräftigt und unterstützt und mir immer wieder neue Impulse gegeben. Für seine exzellente Betreuung gilt ihm mein ganz besonderer Dank.

Die Kooperation mit AE/EST4 und RtP1/MFW5 war entscheidend für diese Arbeit. Vielen Dank insbesondere an Lars Tebje, Jochen Franz und Johannes Classen sowie Thomas Schnitzler und Andreas Feustel.

Besonders danke ich Benjamin Sobotta für die großartige Zusammenarbeit und den wichtigen Beitrag zu dieser Arbeit.

Ein Arbeitsumfeld, wie ich es bei Bosch und am MPI vorgefunden habe, möchte ich auch in Zukunft nicht missen. Ganz besonders danken möchte ich dafür: Karsten Glien und Jochen Schmähling, mit denen ich beim Bier nicht nur nach dem Klettern nicht nur über interdisziplinäre Forschung reden konnte. Denis Gugel, mit dem ich die Zusammenarbeit auf engstem Raum sehr geschätzt habe. Dilan Görür und Malte Kuss, die mich in fachlichen Diskussionen entscheidend beeinflusst haben.

Mein besonderer Dank gilt nicht zuletzt Dieter Kern für die Betreuung der Promotion an der Fakultät für Physik der Universität Tübingen.

[1] Promotionsordnung der Universität Tübingen für die Fakultät für Physik vom 23. Januar 2002, §7 Absatz 1.

Introduction

Motivation and objective

This thesis has resulted from a project to bring together novel developments in machine learning and substantial applications in modern industrial engineering and mass production. The project was done at the Corporate Sector Research and Advance Engineering of Robert Bosch GmbH, Stuttgart, and was supported by the Max Planck Institute for Biological Cybernetics in Tübingen—opening the opportunity to address both, practical and conceptual issues.

Mass production. The great success of mass production consists in manufacturing items in a large number of copies, by breaking their construction down to a number of well-defined manipulations. Semiconductor manufacturing is a prime example for such mass production, creating large numbers of highly complex devices in standardized batch processes.

To ensure a stable quality, each manufacturing step needs to be repeatable by keeping it within defined specifications. For technologically advanced products these requirements become tighter, and inevitable fluctuations might be of the same scale as the allowed tolerances. To effectively control a complex manufacturing environment, it is therefore necessary to collect a large number of data to monitor its stability. Rigorous quality checks at the end of the production line are indispensable to identify and reject sporadic failures.

Industrial engineering. While one objective is to increase the control over the processes in mass production, another lever to increase quality is to design a product to be robust against fluctuations in the first place. However, the high complexity of the manufactured devices often makes it impossible to foresee the actual impact of fluctuations on their functionality, making robust design a formidable task: one is usually dealing with tens or even hundreds of fluctuating parameters which jointly determine a nonlinear response.

Powerful numerical simulation software is now widely available, making it possible to model the complete behavior of a device. However, such complex models can hardly mediate an intuitive understanding of the system, and are usually too time-consuming to be used in an automated, robust design optimization: they are used to perform individual computer experiments, replacing expensive experimental specimens.

Machine learning. When the underlying physical reality is too complex to be described by manageable models—as it is the case for complex devices such as

integrated circuits or processes in semiconductor manufacturing—it becomes impossible to directly interpret observations or to understand the effect of specific changes.

Machine learning addresses this problem by replacing deduction through physical modeling by extracting the underlying relations directly from observed data: Statistical nonparametric models, for example, do not assume a fixed functional dependency, and the complexity of the model is adjusted as more data are observed. These models serve as a machinery to handle high dimensional data, being very flexible and thus applicable to many different problems: They are based on elementary assumptions about the structure of the data, not the underlying physical reality.

In other fields, such as biotechnology, where the physical reality can not yet be modeled, machine learning is already successfully applied in the interpretation of experimental measurements. Thinking of complex manufacturing lines, the identification of the root cause of an observed error is similar to the identification of a gene, responsible for some phenotypic feature: directly investigating the underlying mechanism is a hopeless endeavor, and the machine learning approach is to identify the root cause (gene) by statistically measuring its impact on the error (phenotypic feature).

Machine learning for mass production and industrial engineering. The goal of this thesis is to make novel results from machine learning accessible to open problems in data analysis from product design to mass production, and to introduce industrial manufacturing as an interesting and mainly uncovered field in machine learning research. The development and production of integrated circuits and micro electro-mechanical systems in silicon technology is used as an exemplary case study throughout this thesis.

The thesis starts with a thorough study of a product's cycle from design to mass production, where the emphasis is placed on the analysis of experimental measurements, computer simulations and data collected during production. A number of valuable applications for machine learning could be identified in this survey. These have been worked out and ultimately cast into software tools in collaboration with various departments at Bosch, where they are now routinely used:

The contribution of this thesis to design analysis is a statistically justified approach to compute the robustness and predominant structure of a design from computationally demanding, high dimensional computer models. The approach has lead to a novel scheme for design optimization, which explicitly accounts for process tolerances—while being tractable through an extremely efficient use of simulation runs. Using nonparametric Gaussian process regression, the proposed method is significantly different from previous approaches to robust optimization.

Since Gaussian processes are probabilistic models, which provide a notion of uncertainty, they can be used to plan simulation runs using statistical experimental design—increasing the informativeness of each computer experiment. In this work a novel active learning scheme is derived from Bayesian decision

theory to make sensitivity analysis applicable to highly expensive simulations. In contrast to previous approaches, the learning scheme avoids expensive numerical approximations and directly minimizes the generalization error in a given region of interest.

While design optimization aims at the reduction of failures due to process tolerances, another approach of this thesis addresses systematic failures due to malfunctioning processes. The objective of the troubleshooting task is to identify the root cause of systematic failures, which are identified in final quality checks. Since it is impossible to build a physical model of the complete manufacturing environment, the root cause needs to be filtered from an extremely large number of characteristics of the manufacturing line. In analogy to previous approaches e.g. in bioinformatics, this work proposes a novel scheme for troubleshooting, locating errors using feature selection. Since the method approaches the problem from a phenomenological viewpoint, it allows to relate measurements from quality control to any type of data collected during production.

A guide through this thesis

A successful application of machine learning requires a deep insight into the addressed field, and a broad overview over basic principles and available methods in machine learning. On the one hand, this thesis introduces modern manufacturing as a relatively new field to machine learning, identifying challenging and relevant tasks. On the other hand, novel methods to approach these tasks have been developed and are described in detail.

Therefore, a wide range of topics from semiconductor technology to statistical inference is covered, and the thesis is aimed at both, practioners and machine learning experts. The following overview is intended as a guide to the reader, indicating topic, background and addressee of individual sections—it can be seen as an annotated table of contents.

Chapter I introduces **semiconductor manufacturing** as a prime-example for modern mass production, where data analysis plays an important role: Section 1 comments on the organization of mass-production and section 2 covers engineering for robust designs. Each section is divided into an overview over basic terms and open questions related to machine learning, outlining our approaches in relation to previous work. Besides an introduction to modern manufacturing and design, the chapter serves as a case study for the practioner to exemplify where an application of machine learning can improve off-the-shelf methods.

A very basic fact in machine learning is that no information can be extracted from observations without a suitable model. **Chapter II** outlines the concept of **Bayesian statistics** as a formal way to encode information in the framework of probability theory. Section 1 introduces probabilistic models as the basic ingredient for inference, i.e. the analysis of data in the light of prior information. Statistical model selection and decision theory are covered

by section 2. For a reader who is familiar with Bayesian analysis, these sections might serve as a résumé to introduce concepts and notation used in the following chapters. Illustrating theoretical considerations on the analysis of experimental data, the short survey is also intended to encourage experimenters to use the Bayesian framework as a principled way to handle measurement results. Gaussian processes are nonparametric probabilistic models for regression and classification, which are used extensively in this thesis. Section 3 introduces Gaussian process regression in detail, providing the basis for the proposed approaches to design analysis and active learning.

The approach to **design analysis and robust optimization**, covered by **chapter III**, is a cornerstone of this thesis. Section 1 addresses the fundamental concept of robust designs, and discusses previous work on sensitivity analysis and design optimization in relation to the proposed approach. The measures which are used to analyze the response of a system to fluctuations in mass production are introduced in section 2, their interpretation being exemplified in a case study. While the above sections motivate the approach, section 3 covers the actual implementation, which ultimately makes computations feasible for realistic computer models. These details, as well as the empirical verification in section 4, might be of more relevance to a reader interested in machine learning than for a potential user of the provided software package.

How to plan simulation runs efficiently when the models are nonlinear and high dimensional is covered by **chapter IV**, where a novel **active learning** scheme is developed for Gaussian process regression. Section 1 reviews the historical development of statistical experimental design and active learning. The proposed learning scheme for Gaussian process regression is developed in section 2, where a number of illustrations exemplify the results. The section is mostly aimed at the experienced reader and includes a thorough discussion of the conceptual background. The experimental section 3 serves as a verification of the approach's performance.

Chapter V on **troubleshooting** is largely self contained, as the feature selection approach does not rely on the same methods as the remaining chapters. Section 1 discusses the used data, including the preprocessing which ultimately makes feature selection applicable to this problem. This work's main contribution to the troubleshooting problem has been to identify its analogy to other fields studied in machine learning. The generic concept of feature selection is covered by section 2, where the chosen implementation is motivated in comparison to previous work. To verify the viability of the proposed troubleshooting scheme, it has been tested on historic and recent data from production. The results of the case study are presented in section 3.

The outlined approaches are discussed again in a **concluding section** with regard to the presented results.

I. Machine learning for semiconductor manufacturing

Machine learning provides a large variety of methods to extract information from data. The aim of this thesis is to assess where machine learning promises to be valuable to analyze processes in production and engineering, and to develop tools for the application in practice. Production in semiconductor technology involves a particularly automated manufacturing environment where a lot of data is collected on the way from the design to the finished product. Therefore we concentrate on this field to identify beneficial applications for machine learning. However, we believe that the approaches which we have developed in this thesis apply also to other fields of modern production.

We devote this chapter to the description of semiconductor manufacturing, typical products, their design and fabrication. Note that we slightly abuse the term *semiconductor manufacturing* in this thesis, also using it to refer to the technology to produce silicon-based *micro electro-mechanical systems* (MEMS).

semiconductor manufacturing

MEMS

We start this chapter with a brief description of MEMS in section 1, which are produced on a large scale by Robert Bosch GmbH for sensor applications. The production of such miniaturized systems in semiconductor *foundries* is quite different from traditional manufacturing or assembly. We introduce the basic processes and give an overview on the typical organization of semiconductor mass production, where hundreds of machines and production steps might be involved. In section 2 we discuss the design process, which relies heavily on computer experiments since experimental specimens are typically very expensive and time-consuming in their fabrication. Computer models are used in industrial engineering to validate the functionality of a design, and to assess its robustness with respect to process tolerances.

foundry

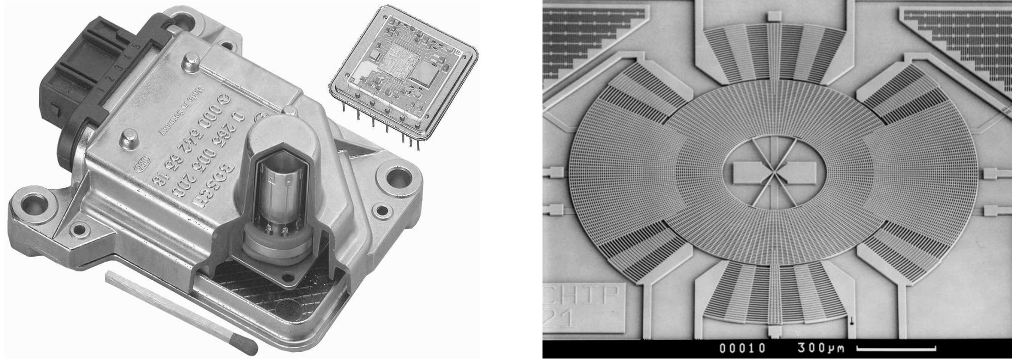
We close each section with a discussion of potential uses of machine learning, introducing related previous approaches and the solutions which we have developed in this work.

1. Semiconductor products & manufacturing

1.1. Micro systems and integrated circuits

In 1965, Moore predicted an exponential growth of the number of transistors per *integrated circuit* (IC) for the next decade (Moore, 1965). The prophecy has remained true until today, and up to 10^9 transistors are now combined in a single processing unit while prices remain stable. Low costs per device

integrated circuit



(a) Yaw rate sensor in fine mechanics and equivalent component in MEMS technology (top right).

(b) Mechanically active structure of the yaw rate sensor in MEMS technology. Size: $2 \times 2 \times \frac{1}{10}\text{mm}^3$.

Figure I.1.: A modern yaw rate sensor in MEMS technology together with a predecessor model constructed in fine mechanics (a). The mechanically active structure of the MEMS sensor is shown in (b). The basic principle is to exploit the coriolis force caused by the rotating frame of reference.

wafer

in combination with increasing performance is mainly due to miniaturization: ICs are produced as batches on so-called *wafers*, silicon discs with a diameter of up to 300mm. Thus we obtain the price for a single device by dividing the costs by the number of units which can be crammed onto a wafer.

The active structures of ICs are generated in an alternating series of processes which deposit or remove thin layers of material on the wafer, where lithographic procedures are used to define fine structures on the layers. These processes are basically the same for all integrated systems, and different products are obtained by using different lithographic masks and a diverse succession of hundreds of processing stages.

yaw rate sensor

The processes for processing silicon have long been restricted to electronic circuits. Robert Bosch GmbH has been a pioneer in transferring the technology to the production of miniaturized "micro" electro-mechanical systems, where partly self-supporting mechanical structures form the active structures. The response of active structures to external forces is measured by means of integrated electrical modules and analyzed by an application specific IC. MEMS are used in sensor applications to measure for example pressure or inertial forces due to acceleration or rotation. Figure I.1 shows mechanical structures, which are used to measure the yaw rate in automotive applications. Compare in panel (a) the degree of miniaturization which could be achieved by the replacement of fine-mechanics by micro structures in MEMS technology.

1.2. Semiconductor foundries

Assembly lines for mass production of most bulk articles are dominated by specially tailored machines, which handle the devices one-by-one and carry out product specific manipulations. In contrast, standard-sized wafers are used

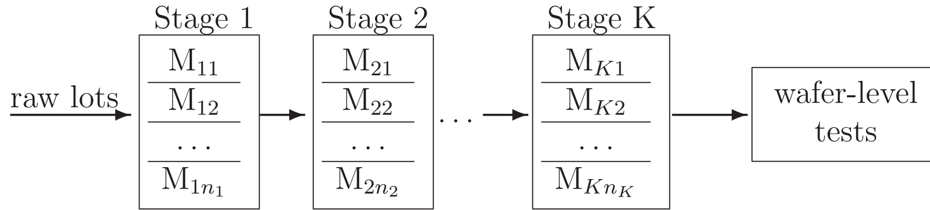


Figure I.2.: Serial-group manufacturing line. The raw lots undergo manipulations in K production stages, and in each stage ℓ we have several machines $M_{\ell 1} \dots M_{\ell n_\ell}$ to choose from. Most lots take different paths through the machinery, as the allocation is designed to maximize the plant utilization. After processing, the functionality of each unit is tested on wafer-level.

in semiconductor foundries for all devices, and manipulations can be reduced to the repetition of a couple of standard processes. While it is expensive and time-consuming to adjust a specialized mass production, it is therefore possible to manufacture a large variety of products simultaneously in one facility, using shared resources. Hence, foundries are not organized in serial manufacturing lines: The machinery consists instead of a large number of multi-purpose machines which are used interchangeably for various processing steps of different products. Single *lots*[1] are guided through the machinery by a computerized system which ensures the correct succession of processing steps. Several equivalent machines are available for most steps, and the typical way of a lot through the machinery can be seen as a serial-group manufacturing scheme, as shown in figure I.2.

lot

In the examples which we describe in chapter V, we are dealing with as many as 500 production stages in which one can typically choose from five equivalent machines. As the allocation to machines is usually designed to maximize the utilization of the machinery, it will therefore hardly happen that two lots share a common history.

Manufacturing in standard chemical processes, such as etching and epitaxial growth, has the advantage that expensive machinery can be used for a large family of products. Unfortunately, using these methods the dimensions of interest can hardly be controlled directly. In epitaxy, for example, the resulting layer thickness would be controlled via deposition time and temperature, in contrast to a direct control in milling metal workpieces. Furthermore, the very idea of using batch processes on wafer-level implies that control measurements are not done for each device. The process stability needs therefore to be controlled on the basis of few measurements on the wafer, which are referred to as *in-line measurements*. Hence, even though the processes in mass production are usually well understood and under tight control, it might happen that previously unseen errors pass the control mechanisms.

in-line measurements

Due to the imperfect coverage of in-line measurements, a conclusive decision regarding the quality of the devices can only be made in *final tests* at the very end of the production chain. However, it is highly important from the

final tests

[1]A lot is a group of wafers which is handled simultaneously.

Stage ID	Equip. ID	Equip. Type	Track in	Track out	Queue
P1_5X	8Q1	WLANT	XX/XX/2004 XX:XX:11	XX/XX/2004 12:XX:08	XX/XX/2004 XX:XX:11
NP_086X	103	WLANT	XX/14/2004 XX:XX:22	XX/XX/2004 10:XX:27	XX/XX/2004 0X:XX:55
NE_087S	QIM	MISKO	XX/XX/2004 XX:XX:45	XX/XX/2004 16:XX:45	XX/XX/2004 XX:X4:35
PAC57X	Q5_3	CRO_AC	XX/XX/2004 13:XX:17	XX/XX/2004 18:XX:14	XX/XX/2004 13:35:03
...	...	...	...	...	...

Table I.1.: The complete processing history is stored in a database for each lot. Recorded are the equipment ID and timestamps when the lot was put into the queue, when processing started and ended.

economical point of view to detect changes in target parameters early in the line in order to be able to react fast to changes in the processes.

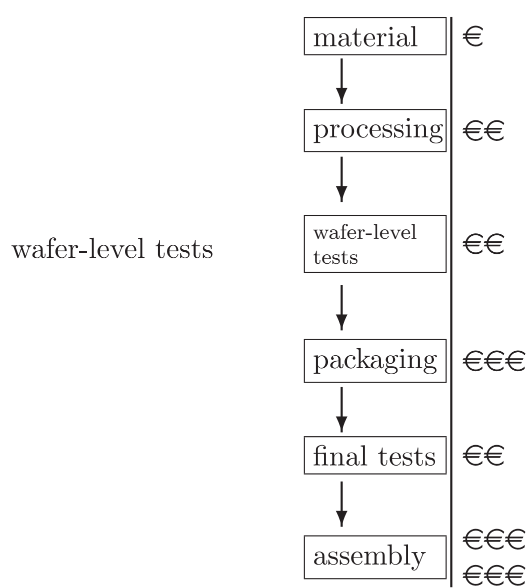


Figure I.3.: Added value.

Especially with regards to the cost structure of batch processes it is is desirable to discard malfunctioning devices before separation: Batch processing makes handling cheap, and a large fraction of the costs for a finished device can therefore be assigned to the last stages after the devices have been separated. Each device is therefore tested rigorously on *wafer-level* before the units are separated. A schematic view on the structure of semiconductor production is shown in figure I.3.

wafer-level tests

The tests on wafer-level are an important source of information for the manufacturing line: As we have outlined, in-line measurements are usually fragmentary and the finishing tests on wafer-level are the first to control each device. Following tests after separation are of restricted significance to the manufacturing line, as the packaging has a significant influence on the results. This makes it hard to relate the results to processes in the wafer foundry.

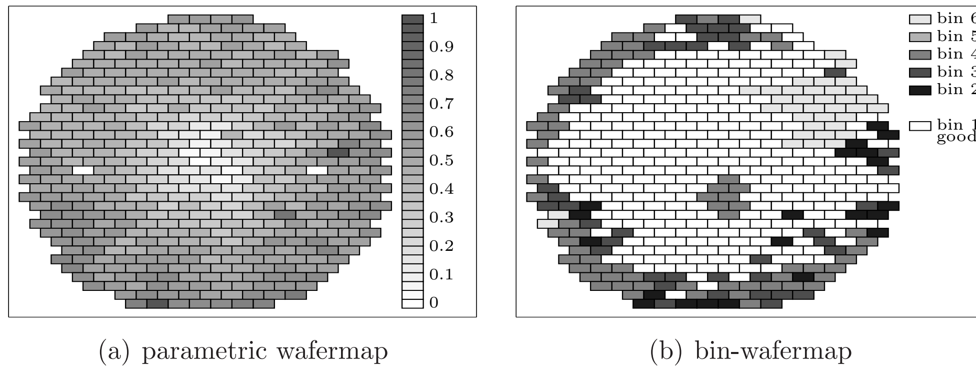
1.3. The "smart" factory

For serial manufacturing lines it is sufficient to stack the workpieces in front of the machine which is next in the process chain. As we have seen, however, the organization of a semiconductor foundry is more intricate. Most lots which are waiting in the queue require different processing and it is necessary to keep track of the actual state of each lot.

lot-history

For this reason the complete *lot-history* is stored in a database, which can also be accessed for further analysis. An exemplary excerpt from such a lot-history is shown in table I.1[2]: The database stores the time when each lot enters

[2]Note that the data has been alienated for reasons of nondisclosure.



(a) parametric wafermap (b) bin-wafermap

Figure I.4.: The results of on-wafer tests are typically shown as wafermaps, where the results are arranged according to the units' positions on the wafer. Panel (a) shows a parametric wafermap with numerical test results. Panel (b) shows a pass/fail wafermap, where several tests are combined by classifying the units into several failure bins.

the queue for a production stage, the beginning and ending of processing, as well as the ID of the used equipment.

Our running example is semiconductor manufacturing. However, many other modern shop floors provide similar structures to store similar information, and the methods which we propose in this thesis can therefore also be valuable there.

To ensure process stability, a great variety of in-line measurements are performed during production. In contrast to the lot-history, these data are not necessarily stored in the database, as it is often sufficient to install a control mechanism locally at the machine. The in-line data is therefore only partly available for subsequential analysis and might have to be collected manually.

The results of the electrical on-wafer tests are tens of numerical values or characteristic curves, which are recorded for each die. Units which fail these tests are *inked* and separated out. Different types of failures are pooled in several *error bins*, and their distribution on wafers is often inspected in form of so-called wafermaps (figure I.4[2]).

ink
error bins
wafermaps

The data which are collected in the final measurements after separation are similar to that from on-wafer tests. However, as we have mentioned above, the units' characteristics are usually well defined by the on-wafer tests, and deviations from the final tests are dominated by effects from separation and packaging.

1.4. Applications of machine learning in production

Having introduced the structure of semiconductor manufacturing and related problems, we will now discuss how machine learning might help to use available data to ease the manufacturing process.

optical inspection

Optical inspection. A prominent application of machine learning methods in manufacturing is the automation of visual tests, which are generally very expensive and little effective if done manually. Tobin et al. (2001) discuss a procedure which automatically compares shots of single dies to reference pictures and counts deviations as defects. Such algorithms are now commonly used, their output listing single defects together with their position on the wafer. This allows for a visualization on *defect maps*, which are similar to wafermaps.

defect maps

SPC

Sheward chart

SPC. *Statistical process control* (SPC) is commonly used to control processes with a large number of inspection variates. A detailed description is given by Wieringa (1999). The basic tool in SPC are *Sheward control charts*, where each relevant parameter is plotted together with its specifications against the time (figure I.5). Since single outliers are easily detected automatically, the main challenge is the detection of drifts or sudden shifts.

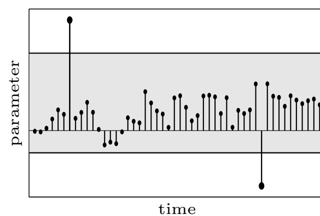


Figure I.5.: Sheward control chart. Shown are measurements (•) and the tolerance window (—).

Wiel (1996); Rao et al. (1996); Leang et al. (1996); Kohlmorgen and Lemm (2001); Chinnam (2002) report on automatic methods to detect changes in the time series. A common idea for the analysis is to use a regression model to predict future measurements. Such models can foresee a drift out of the tolerance window and detect sudden changes by strong deviations of predictions and measurements.

SPC relies heavily on manual inspections and simple models which do not reflect the interplay between different parameters. Therefore, manufacturing can only be controlled by inspecting the temporal evolution of the variates.

process modeling

Process modeling. The goal of advanced methods is to instead model the behavior of complete processes, representing the interrelation of several parameters. Machine learning methods have already been used to build such models for engineering tasks where new processes are evaluated:

Braha and Shmilovici (2002), for example, report to have optimized a novel cleaning process under the constraint that only a small amount of data was available. The picture is different when it comes to the control of established processes. Fenner et al. (2005) have developed a process control based on a simultaneous examination of several parameters, which is in use in an experimental clean room. However, such a mechanism seems not to be applicable in mass production, where processes are tightly controlled and errors occur mostly due to previously unseen causes:

A process model which is trained on data from the normal run can hardly give reasonable predictions to control atypical situations, and it seems more natural for mass production to aim at the construction of automatic mechanisms for troubleshooting. As Agosta and Gardos put it, “by their nature, breakdown events are rare, and a troubleshooting model cannot be built solely by a data-driven approach”. In (Agosta and Gardos, 2004) they describe how

to solve this problem by using Bayesian networks which allow to incorporate expert knowledge in the analysis[3]. Manago and Auriol (1996) and Cheetham et al. (2001) present earlier work on data-based expert systems. The technique is referred to as "case-based reasoning", and is commercially used to guide engineers through the troubleshooting process.

case-based reasoning

Troubleshooting. The aim in troubleshooting is fundamentally different from the approach to reproduce foundry processes in empirical models. As mentioned above, such models necessarily represent processes under normal conditions and can hardly generalize well to previously unseen, atypical situations. In contrast, troubleshooting approaches directly address abnormal observations. These approaches are really what is needed in mass production, as processes are already well understood under normal conditions. One is not primarily interested in modeling *why* things go wrong, but in detecting whether something *is* going wrong and isolating the root cause. Once the cause is located, the maintenance can be pointed to the right machine and solve the problem.

troubleshooting

The expert knowledge, which is necessary to detect errors, can be encoded by a precise definition of error patterns. Bergeret and Gall (2003), for example, describe a simple approach to find the root cause for a repeated observation of some error: In the serial-group structure of wafer processing (figure I.2) lots are usually mixed after each processing stage, and Bergeret and Gall locate the responsible machine as the one which has handled several conspicuous lots consecutively. The assumptions in this approach are that lots mix during production and that a single machine causes the error repeatedly over a period of time.

We present a related approach based on *feature selection* in chapter V, which is also described in (Pfingsten et al., 2005). Similarly to the above approach, we combine the lot-history with a list of detected errors. However, our method is not restricted to the above assumptions.

feature selection

Error detection. Troubleshooting methods are necessarily based on the combination of the lot-history with a list of workpieces which show similar abnormal characteristics. Such lists might be created manually, but it is certainly worthwhile to think of ways to automatically identify systematic errors based on a vague description of what might go wrong. The most obvious target figure in wafer foundries is the yield of flawless units. Bensch et al. (2005) correlate the yield and in-line measurements to identify early indicators for yield losses.

yield

The spatial distribution of failures on a wafer can give additional insight into the source of the problem. The wafermap shown in figure I.4, for example, shows a significantly increased failure rate at the rim of the wafer. As failures which are caused by particle contamination tend to be uniformly distributed over the wafer, patterns on wafermaps indicate that a process does

[3]For introductory texts on graphical models and Bayesian networks see (Buntine, 1996; Jordan et al., 1999).

not run optimally. Defect-maps from optical inspection can be analyzed with similar methods to detect spatial correlations. Several works discuss methods to attribute failures to different failure patterns. See (Duvivier, 1999; Fountain et al., 2002; Nicolao et al., 2003; Riordan et al., 2005). We discuss a number of real-world examples from the Bosch foundry, where we were able to localize root causes for errors which were detected on the basis of such patterns (chapter V).

2. Computer-aided design

The above section covers the potential benefit of machine learning approaches to mass production, in particular semiconductor manufacturing. In the following section we describe modern development, where experiments and especially computer simulations are used to assess the fitness of a design for mass production. We outline the designing process using computer experiments in section 2.1, and discuss in 2.2 how they are used to determine the robustness of designs against process tolerances. We outline our approaches to these problems in 2.3.

2.1. Computer experiments

The construction of simple mechanical systems can usually be done manually and without the aid of special software. However, as quality requirements and the system's complexity increase, the designer can no longer master the system's characteristics, and a valid design can hardly be obtained at a single try. Simulation software can help in those cases to speed up necessary iterations in the design cycle and may allow for an automatic design optimization.

Before computational power was widely available, only relatively simple models were used. Such simple models need to be based on very specific assumptions and one has to identify beforehand what question they are to answer. Therefore they require a deep understanding of the system in consideration.

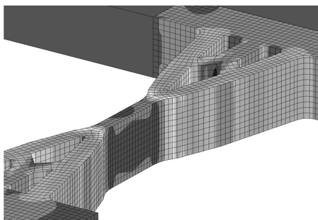


Figure I.6.: Exemplary plot of an FEM computer experiment to determine the tensile strength of a silicon structure.

Today, however, simulation techniques have matured to the point where computer models can describe all relevant features of a system, including geometrical properties, electrical and thermal aspects, and circuit simulations. Typically used simulation tools are based on *finite element methods* (FEM) or *equivalent network models*. Such models are constructed as one-to-one emulations of a device and mimic their behavior (as illustrated in figure I.6). The purpose of modern computer models is thus to replace experimental specimens where possible, as the latter may be extremely expensive and time-consuming in their fabrication and measurement.

2.2. Robust designs for mass production

Figure I.1(b) shows a modern yaw rate sensor which is produced in silicon-based MEMS technology with complex structures on the scale of micrometers. Using photolithographic processes, these extremely small structures can be manufactured within tolerance windows below micrometers, which can hardly be achieved by traditional cutting machining.

However, while typical fluctuations are small on an absolute scale, they may be large compared to the dimensions of the structures, and the performance of the manufactured devices may vary significantly from unit to unit. High quality standards require the characteristics of a product to lie within a small tolerance window. A main task of simulation is therefore to assess and optimize the robustness of the design with respect to random variations which are inherent to the used processes.

Computer models which simulate the behavior of a device can be seen as a deterministic mapping from geometrical and material parameters of the system (input parameters) to its characteristics (output parameters). The main task is thus to estimate the variation in the output by combining the deterministic computer model and the distributions of the input parameters. The corresponding techniques are embraced by the terms *uncertainty* or *sensitivity analysis*. The challenge is that the system's response needs to be studied over a range of parameter settings using a limited number of simulation runs.

uncertainty analysis

sensitivity analysis

2.3. Machine learning for design analysis

In simple, low dimensional problems it is common to evaluate the computer model on a regular grid of parameter settings and to interpolate to new parameter settings using linear or polynomial interpolation. Once the grid has been computed, the interpolation schemes make it possible to avoid expensive software and long computation times when asking for the simulation results at new input parameters. The possibly large number of settings on the grid can be calculated as a batch and the engineer does therefore not have to wait for the results of single runs.

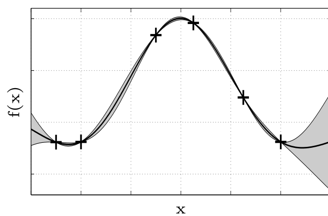


Figure I.7.: GP fit. Training instances (+), predictive mean (—), and 2σ-confidence interval (gray).

Now it is possible to run computer experiments with many parameters, which might show strong nonlinear effects. Simple interpolation schemes are not applicable for these cases as the size of the grid explodes with the number of dimensions. *Gaussian processes* (GPs) are a common regression model for high dimensional, nonlinear problems, which have been used as fast surrogates of expensive computer code in a number of previous works[4]. GPs can efficiently solve the interpolation task for high dimensional problems, and can combine given function

Gaussian process

[4]See for example (Sacks et al., 1989a; Bernardo et al., 1992; Welch et al., 1992; Morris et al., 1993; Rasmussen, 2003).

values and gradient information.

probabilistic models

In contrast to most other regression methods, GPs are *probabilistic models*, i.e. they not only deliver point estimates of the underlying function, but also come with a notion of uncertainty. For a simple example see figure I.7, where we have plotted a GP fit with the 2σ confidence interval for its predictions.

In this thesis we use GPs to approach two topics, which are related to the problems we face in the design of micro systems: We propose a methodology to analyze the robustness of a design with respect to process fluctuations, and to automate design optimization. The probabilistic nature of GPs can be used to define experimental designs, which enhance the value of each simulation run when exploring a region of interest.

Design analysis and optimization. As we have argued above, an important purpose of computer experiments is to study the robustness of designs with respect to fluctuations in production and to find influential parameters. Sensitivity analysis is traditionally done using the "brute force" Monte Carlo method or using restricted linear models. However, when expensive computer experiments are used, the number of simulation runs is crucial, and we need to obtain a certain accuracy using as few runs as possible. As suggested by Haylock and O'Hagan (1996) we replace the Monte Carlo method by one that uses a GP prior. We show in extensive experiments that the method reduces the number of necessary runs by an order of magnitude, thus saving time, resources, and expensive software licenses. The GP meta-model can be used to do an efficient gradient-based design optimization to maximize the robustness with respect to fluctuations in manufacturing. Chapter III covers design analysis in detail. We have also described the approach in (Pfingsten et al., 2006a).

Bayesian experimental design

Active learning. Using GPs we can reduce the number of necessary simulation runs further if we carefully plan at which parameter settings the computer code should be run. In the context of probabilistic models, which include a notion of uncertainty, *Bayesian experimental design* provides a formalism to determine optimal designs once an objective function is defined. We develop optimal designs for sensitivity analysis and show that the approach is significantly more efficient than the state-of-the-art methods for passive learning. We discuss our approach in chapter IV. The results have been published in (Pfingsten, 2006).

II. Bayesian methods in machine learning

The Bayesian interpretation of probability provides a principled way to reason under incomplete knowledge. By interpreting probability as a measure for subjective belief, the mathematical framework of probability theory is applied to quantify the a-posteriori knowledge as a combination of a-priori beliefs and information from observed data. In this chapter we outline the generic principles of Bayesian analysis, which are used throughout this thesis.

An overview on Bayesian methods is given by MacKay (1999, Chap. 4), who presents them in relation to other approaches in machine learning. The textbook of Berger (1985) addresses the subject from the statistical viewpoint, and discusses Bayesian analysis in more detail. Jaynes's (2003) book contains many illustrative examples, and in particular examines the differences between the frequentist and the Bayesian interpretation of probability. An article by Cox (1946) deserves special attention: the author deduces the algebra of probability theory from universal rules for *reasonable expectation*.

The historical background of Bayesian statistics is covered by Jaynes (1985) and Fienberg (2006). The connections to statistical physics are particularly interesting. Jaynes (1957a,b), for example, showed that the principles of statistical mechanics can be seen as an instance of Bayesian inference under the maximum entropy principle. Related to this, the entropy, which first appeared in thermodynamics, can be reinterpreted as a measure for information (Shannon, 1948).

Gaussian processes (GPs) are a common class of models for high dimensional, nonparametric regression and classification. In this chapter we discuss GP models for regression, to apply them in our approaches to sensitivity analysis (chapter III) and active learning (chapter IV) for computer experiments in industrial development.

Gaussian process
GP

A recent textbook on GPs is (Rasmussen and Williams, 2006), Stein (1999) approaches the subject from a more theoretical perspective. GP priors have a long history as *kriging* in spatial statistics (Matheron, 1963), and they are discussed in a number of works by Sacks and Ylvisaker (1966, 1968, 1978) to describe correlated deviations from parametric models. O'Hagan (1978) introduces GPs as localized linear models, and Neal (1996) shows that they can, in special cases, also be seen as neural networks with an infinite number of hidden neurons.

We outline the rules for Bayesian inference in section 1, the basic ideas of Bayesian model selection and decision theory are discussed in section 2. We discuss GPs in some more detail in section 3.

1. Bayesian inference

The following section outlines the basic concepts of Bayesian inference. While introducing the general formalism, we illustrate the procedure on a simple analysis of experimental data. The example is based on an experiment by Glien et al. (2004) on the reliability of MEMS devices. A detailed description of the analysis has been published in (Pfingsten and Glien, 2006).

Our presentation starts with an overview on the nature of probabilistic models (section 1.1). In section 1.2 we introduce the formal way of doing inference in the Bayesian framework, where we briefly introduce Markov Chain Monte Carlo methods as a tool for numerical approximation.

1.1. Probabilistic models

The first ingredient of Bayesian analysis is a model

$$\mathcal{M}(\boldsymbol{\alpha}) \, , \tag{1.1}$$

which encodes our belief about the system we analyze. While the structure of the model $\mathcal{M}$ remains fixed, it is specified by uncertain parameters $\boldsymbol{\alpha}$ which need to be inferred from observations.

prior

The knowledge we have about the parameters $\boldsymbol{\alpha}$ before the analysis is encoded in a *prior distribution*

$$p(\boldsymbol{\alpha}|\mathcal{M}) \, , \tag{1.2}$$

which might contain information from previous measurements or theoretical considerations. Prior distributions are well-known from statistical physics, where the maximum entropy principle is the foundation for the analysis of systems with a large number of degrees of freedom (Jaynes, 1957a,b).

Once we have observed some data $\mathcal{D}$ which are related to the model, the prior beliefs can be updated according the new information. The probability distribution of the observations,

$$\mathcal{L}(\boldsymbol{\alpha}) \quad = \quad p(\mathcal{D}|\mathcal{M}, \boldsymbol{\alpha}) \, , \tag{1.3}$$

likelihood function

is part of the model, and specifies how the data relates to the model and its parameters $\boldsymbol{\alpha}$. As a function of the parameters, $\mathcal{L}(\boldsymbol{\alpha})$ is referred to as the *likelihood function*. It might contain additional constants, such as a noise level, which are part of the model $\mathcal{M}$. According to the *likelihood principle*, the likelihood expresses all information which the data contain about the parameters $\boldsymbol{\alpha}$. It should therefore be the only place where the data enter the analysis.

Example (part 1): Our illustrative example analyzes an experiment to determine the parameters of a model for slow crack growth. To keep the presentation clear, we simplify the notation and leave aside all technical details (see (Glien et al., 2004)). In the experiment one measures the fracture strength y for several specimens at different loading rates x.

The ys scatter around a characteristic fracture strength $f_{\mathrm{FS}}(x, \boldsymbol{\alpha})$, which depends on x and three characteristic parameters $\boldsymbol{\alpha} = (a, b, c)$:

$$\log\left[f_{\mathrm{FS}}(x, \boldsymbol{\alpha})\right] = \begin{cases} ax + b & \text{for } x \leq c \\ ac + b & \text{for } x > c \end{cases} \tag{1.4}$$

The fracture strength of single experimental specimens can differ widely, and the deviations from the characteristic strength are described by a Weibull distribution, $p(y(x)|f_{\mathrm{FS}}(x, \boldsymbol{\alpha}), d) = \mathrm{WB}(y(x)|f_{\mathrm{FS}}(x, \boldsymbol{\alpha}), d)$ with an extra parameter d. Thus, the likelihood for measurements $\mathcal{D} = \{(x_1, y_1) \ldots (x_N, y_N)\}$ is given by

$$\mathcal{L}(\boldsymbol{\alpha}) = \prod_{\ell=1}^{N} \mathrm{WB}\big(y_\ell | f_{\mathrm{FS}}(x_\ell, \boldsymbol{\alpha}), d\big) \ . \tag{1.5}$$

We have practically no prior knowledge about the model parameters, and therefore we set the prior $p(\boldsymbol{\alpha}|\mathcal{M})$ to be uniform in the physically sensible range.

1.2. The posterior distribution

Once we have specified our prior beliefs and the likelihood function, we can infer the parameters $\boldsymbol{\alpha}$ from observed data $\mathcal{D}$. *Bayes' theorem* provides the formal rule to combine prior and likelihood:

Bayes' theorem

$$\underbrace{p(\boldsymbol{\alpha}|\mathcal{D}, \mathcal{M})}_{\text{posterior}} = \underbrace{p(\boldsymbol{\alpha}|\mathcal{M})}_{\text{prior}} \ \underbrace{p(\mathcal{D}|\boldsymbol{\alpha}, \mathcal{M})}_{\text{likelihood}} \times \underbrace{[p(\mathcal{D}|\mathcal{M})]}_{\text{evidence}}^{-1} \ . \tag{1.6}$$

The updated, or *posterior* distribution represents the a-posteriori belief about $\boldsymbol{\alpha}$. The rightmost term, the *marginal likelihood* or *evidence*,

posterior
marginal likelihood
evidence

$$p(\mathcal{D}|\mathcal{M}) = \int \mathrm{d}\boldsymbol{\alpha}\ p(\mathcal{D}|\boldsymbol{\alpha}, \mathcal{M})\ p(\boldsymbol{\alpha}|\mathcal{M}) \ , \tag{1.7}$$

is independent of $\boldsymbol{\alpha}$ and normalizes the posterior.

The posterior distribution $p(\boldsymbol{\alpha}|\mathcal{D}, \mathcal{M})$ contains all information we have about the parameters $\boldsymbol{\alpha}$—including the remaining uncertainty which is often displayed in the form of confidence intervals. Assume we are instead interested in the posterior distribution $p(f|\mathcal{D}, \mathcal{M})$ of some function f which depends on $\boldsymbol{\alpha}$. Using the product and sum rule of probability we obtain

$$p(f|\mathcal{D}, \mathcal{M}) = \int \mathrm{d}\boldsymbol{\alpha}\ p(\boldsymbol{\alpha}|\mathcal{D}, \mathcal{M})\ p(f|\boldsymbol{\alpha}, \mathcal{D}, \mathcal{M}) \ , \tag{1.8}$$

which is the corresponding average over the posterior $p(\boldsymbol{\alpha}|\mathcal{D}, \mathcal{M})$.

Markov Chain Monte Carlo (MCMC) approximations. Integrals of the type (1.8) can in general not be solved analytically. A standard technique to solve the integrals numerically are Monte Carlo methods, which approximate

Monte Carlo

the average by the empirical mean

$$p(f|\mathcal{D},\mathcal{M}) \approx \sum_{\ell=1}^{M} p(f|\boldsymbol{\alpha}_\ell,\mathcal{D},\mathcal{M}) , \tag{1.9}$$

where the $\boldsymbol{\alpha}_1 \ldots \boldsymbol{\alpha}_M$ are samples from the posterior distribution $p(\boldsymbol{\alpha}|\mathcal{D},\mathcal{M})$. As it is generally impossible to draw the samples directly, MCMC methods are used to construct Markov Chains, which approach $p(\boldsymbol{\alpha}|\mathcal{D},\mathcal{M})$ as an equilibrium distribution.

Markov Chain Monte Carlo MCMC

MCMC techniques are guaranteed to converge in the limit $M \to \infty$ under mild conditions. However, they can be computationally expensive and require manual inspection. Hence, MCMC is usually avoided where possible, but it is often the only alternative to approach the exact solution for (1.8). MCMC methods for Bayesian inference are covered by MacKay (2003, Chap.IV), more details are given by Neal (1993, 1996, 1997). A discussion of the practical usage of MCMC can be found in (Kass et al., 1998).

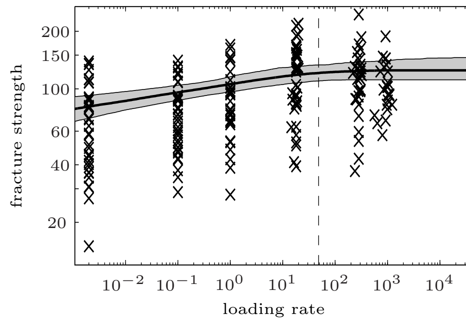


Figure II.1.: Analysis of a dynamic loading experiment.

EXAMPLE (PART 2): The relation in (1.4) is an example for a simple parametric model. The adjoining figure shows the posterior estimate of the characteristic fracture strength as a function of the loading rate x. The solution was obtained using an MCMC technique. Figure II.1 shows the measured data ($\times$), the mean estimate (—) and the 95% confidence interval (gray) for the characteristic fracture strength. Note that the data scatter widely around characteristic fracture strength as the Weibull module d for this material is very small. Confidence intervals for the parameters $\boldsymbol{\alpha} = (a, b, c)$ can directly be obtained from their posterior distribution.

2. Decision theory and model selection

In the previous section we have outlined how to do inference under a given model, combining prior beliefs and information from observed data. However, experiments are often performed to verify the model assumptions themselves, and the aim in machine learning is to find a model which best reflects the structure of the data to generalize to new instances. The problem of comparing the fitness of models under the light of given measurements is covered by Bayesian model selection, which we discuss in the following section 2.1. The overview includes a discussion of the common *maximum likelihood approximation of type II* (ML-II) and the much-cited principle of *Occam's razor*.

Model selection is an instance of decision theory, which addresses the quantitative rating of actions in the presence uncertainty. We discuss the generic formalism in section 2.2, as it builds the foundation for Bayesian active learning, covered by chapter IV.

2.1. Model selection

Model comparison. Assume we compare the plausibility of a set of models $\mathcal{M}_\ell$, $\ell \in \{1, 2 \ldots\}$ on the basis of our prior beliefs $p(\mathcal{M}_\ell)$ and observed data $\mathcal{D}$. According to Bayes' rule the posterior probability for a model $p(\mathcal{M}_\ell|\mathcal{D})$ is given—up to a constant factor—by $p(\mathcal{M}_\ell)p(\mathcal{D}|\mathcal{M}_\ell)$. Two models $\mathcal{M}_1$ and $\mathcal{M}_2$ are thus compared via their *posterior odds*:

posterior odds

$$\underbrace{\frac{p(\mathcal{M}_1|\mathcal{D})}{p(\mathcal{M}_2|\mathcal{D})}}_{\text{posterior odds}} = \underbrace{\frac{p(\mathcal{M}_1)}{p(\mathcal{M}_2)}}_{\text{prior odds}} \underbrace{\frac{p(\mathcal{D}|\mathcal{M}_1)}{p(\mathcal{D}|\mathcal{M}_2)}}_{\text{Bayes' factor}} . \tag{2.10}$$

As the prior odds are given independently of observations, the data enters the comparison only through the *Bayes' factor* (Kass and Raftery, 1995), which is the ratio of the evidences (1.7).

Bayes' factor

Model selection is formally solved by considering the Bayes' factors (2.10). Note, however, that the computation of the involved evidences is often very hard even in numerical approximation. The fully Bayesian approach to model selection is therefore only rarely used in practice.

Maximum likelihood type-II. It can be convenient to define a class of models $\mathcal{M}_{\boldsymbol{\theta}}$ via a set of parameters $\boldsymbol{\theta}$, which are referred to as *hyperparameters*[1].

hyperparameters

In contrast to the parameters $\boldsymbol{\alpha}$ within the model, the hyperparameters parameterize the model itself. However, the difference is only of interpretative nature. Just as the parameters $\boldsymbol{\alpha}$ are integrated out according to (1.8), the hyperparameters need to be treated by averaging over their posterior distribution:

$$p(f|\mathcal{D}, \mathcal{M}) \quad = \quad \int \mathrm{d}\boldsymbol{\theta}\; p(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M})\; p(f|\mathcal{D}, \mathcal{M}_{\boldsymbol{\theta}}) \,. \tag{2.11}$$

The corresponding integral is usually hard to solve analytically or numerically, the only feasible approaches often being MCMC methods.

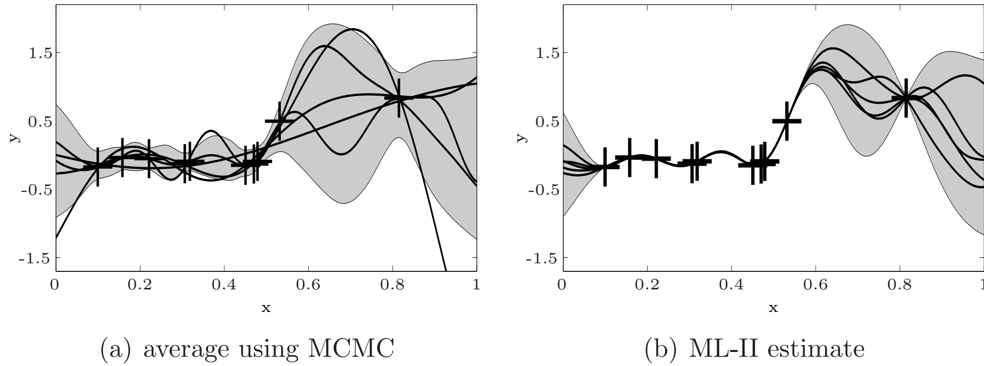
The average is therefore often replaced by model selection: A common assumption is that no set of hyperparameters is to preferred, i.e. that $p(\mathcal{M}_{\boldsymbol{\theta}})$ is flat[2]. The best model according to the Bayes' factors (2.10) is in this case found by optimizing the marginal likelihood $p(\mathcal{D}|\mathcal{M}_{\boldsymbol{\theta}})$ with respect to $\boldsymbol{\theta}$. The method is therefore called *maximum likelihood of type-II* (ML-II). For more details see (Berger, 1985, Chap 3).

Maximum likelihood of type-II

ML-II

[1]In the last section's example we have already introduced a hyperparameter in the form of a parameter in the likelihood function. The likelihood (1.5) has a parameter d, which is a property of the analyzed material.

[2]Note that this assumption is not invariant under variable transformation.



(a) average using MCMC (b) ML-II estimate

Figure II.2.: Illustration of the posterior distribution (2.11) and its ML-II approximation. Both panels show the predictive distribution of Gaussian process regression with observations (+). The gray area indicates the 95% confidence interval for the latent function, and the solid lines are random samples from the posterior. The MCMC average over hyperparameters (a) includes smoother samples, which interpret parts of the variation as noise. In contrast, the ML-II estimate (b) chooses only one set of hyperparameters and overfits the data, effectively interpolating between the observations.

ML-II can be interpreted as an optimistic approximation to the posterior $p(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M})$, which is replaced by a sharp δ-distribution:

$$p(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M}) \approx \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*) \qquad \text{with} \quad \boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\; p(\mathcal{D}|\mathcal{M}_{\boldsymbol{\theta}}) \, . \tag{2.12}$$

An illustrative example can be found in figure II.2, where we show an ML-II estimate (2.12) in comparison to the MCMC approximation to the exact average (2.11). The example is a simple regression problem, where we have used a flexible Gaussian process prior[3]. Gaussian processes are only introduced in the following section 3, however, leaving the details aside, the example illustrates how the overly confident ML-II estimate can lead to overfitting in flexible model classes.

Occam's razor. Fully Bayesian inference automatically "smooths" extreme predictions of overly complex models by averaging over the hyperparameters (2.11). This can be seen as an automatic implementation of what is often informally called *Occam's razor* (MacKay, 1992a). The ML-II approximation (2.12) can, in contrast, lead to overfitting:

Occam's razor

On the one hand, optimizing the hyperparameters instead of averaging over all plausible possibilities as suggested by (2.11), can lead to poor results when doing inference in a flexible class of models. On the other hand, ML-II is often used to estimate the evidence of model classes, to compare them using Bayes' factors (2.10). ML-II is used replacing the averaged evidence,

[3]For the example in figure II.2 we have assumed a Gaussian process prior as introduced in section 3. We used the squared exponential covariance function (3.20) with ARD-distance (3.18).

$\int d\boldsymbol{\theta}\ p(\mathcal{D}|\mathcal{M}_{\boldsymbol{\theta}})\ p(\boldsymbol{\theta}|\mathcal{M})$, by the maximized $p(\mathcal{D}|\mathcal{M}_{\boldsymbol{\theta}^*})$. When optimized ML-II evidences are used, flexible models are necessarily favored, possibly in spite of poor generalization. Bayesian model selection for nonparametric regression is analyzed in detail in (Pfingsten and Rasmussen, 2006).

2.2. Decision theory

Model selection is an example for decision problems, where we seek to take an optimal action a out of a set of alternatives $\mathcal{A}$, based on our prior beliefs and the available information. In order to cast a decision problem into the mathematical framework, one needs to specify a *utility function* U to reflect the optimality criterion which the action is supposed to maximize:

utility function

$$\mathrm{U}(a, \boldsymbol{\alpha}, \boldsymbol{\theta}|\mathcal{M}, \mathcal{D}) \ . \tag{2.13}$$

The utility function naturally depends on prior information, which are reflected by assuming a class of models $\mathcal{M}$ and observed data $\mathcal{D}$. Data and prior can be considered fixed, which we indicate by conditioning U on $\mathcal{M}$ and $\mathcal{D}$.

The action a is to be chosen to maximize U, however, the utility might also depend on uncertain parameters $\boldsymbol{\alpha}$ and the hyperparameters $\boldsymbol{\theta}$ which correspond to $\mathcal{M}$. Such uncertain parameters are integrated out by averaging over the posterior distribution:

$$\mathrm{U}(a|\mathcal{M}, \mathcal{D}) = \int d\boldsymbol{\alpha} \int d\boldsymbol{\theta} \quad p(\boldsymbol{\alpha}, \boldsymbol{\theta}|\mathcal{M}, \mathcal{D}) \quad \mathrm{U}(a, \boldsymbol{\alpha}, \boldsymbol{\theta}|\mathcal{M}, \mathcal{D}) \ . \tag{2.14}$$

Hence, the *Bayesian expected utility* is the averaged utility function with a weighting according to the posterior belief in the uncertain parameters.

Bayesian expected utility

3. Gaussian process priors

The running example in the previous section was a parametric model to describe the nature of slow crack growth. Such physically motivated models cover a very broad class of inference problems, where the generative process is well understood. In contrast, *nonparametric models* are formulated avoiding such specific assumptions to generalize from observations to unseen instances—possibly without understanding the underlying mechanisms.

nonparametric models

Gaussian processes are nonparametric models to encode a prior distribution on a very broad class of functions, directly specifying their characteristics without a detour via parameterization. In combination with various likelihood functions, GPs can be used e.g. for classification or regression. In this thesis we present applications of GP regression, whose concepts are outlined in section 3.1. GPs can be seen as a kernel method, where the covariance function is re-interpreted as a kernel. In section 3.2 we analyze the assumptions, which are implicit to the choice of the covariance function.

3.1. Gaussian process regression

Assume we model a mapping f from input parameters $\mathbf{x} \in \mathbb{R}^D$ to an output $f(\mathbf{x}) \in \mathbb{R}$. GPs extend parametric models by allowing for systematic deviations from a (parameterized, e.g. linear) mean function

$$\mu(\mathbf{x}) = \mathrm{E}[f(\mathbf{x})] \,. \tag{3.15a}$$

The simplest way to deal with deviations is to interpret them as pure noise. In this case we have $f(\mathbf{x}) = \mu(\mathbf{x}) + \epsilon$, where the ϵ are independent identically-distributed random variables.

If the parametric model $\mu(\mathbf{x})$ does not perfectly fit the data, the deviations are systematic, and might be similar especially at neighboring inputs $\mathbf{x}$ and $\bar{\mathbf{x}}$. The simplest way to model the similarity of the deviations is to use a covariance function, which models their dependence as a function of their position:

$$k(\mathbf{x}, \bar{\mathbf{x}}) = \mathrm{cov}\left[f(\mathbf{x}), f(\bar{\mathbf{x}})\right] \,. \tag{3.15b}$$

Formally speaking, GPs are an instance of random processes which are defined as follows:

Definition 1 (Random process) *A random process is a collection of random variables $X(\mathbf{x})$ with $\mathbf{x} \in T$, where the index set T can be finite or continuous.*

For regression we are interested in the continuous case, where $T \subset \mathbb{R}^D$. A Gaussian process is completely defined by the mean the covariance function (3.15):

Definition 2 (Gaussian process) *A Gaussian random process is a random process, where the joint distribution of all finite collections $X(\mathbf{x}_1) \ldots X(\mathbf{x}_N)$ with $\mathbf{x}_1 \ldots \mathbf{x}_N \in T$ is multivariate normal. A Gaussian process is therefore defined by a mean function $\mu(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \bar{\mathbf{x}})$.*

In the following we focus on the structure of GPs, which is governed by the covariance function. The mean function only defines a fixed offset function, and therefore we set it to zero to keep the following derivations simple.

While we assume through parametric models that the mapping f is defined by some parameters, GPs directly encode a prior on a function space. The covariance function specifies the structure of the model, and we collect its parameters as hyperparameters in a vector $\boldsymbol{\theta}$.

In most setups we do not directly measure f for some input $\mathbf{x}$, and instead observe a $y(\mathbf{x})$, which is in some way related to the latent function value $f(\mathbf{x})$. The likelihood function $\mathcal{L}(f(\mathbf{x})) = p(y(\mathbf{x})|f(\mathbf{x}), \boldsymbol{\theta})$ defines the relation of the latent function f and the observations y[4]. As before we add possible

[4]Recall the definition of the likelihood: In contrast to the definition in (1.3), $p(y, \mathbf{x}|f, \boldsymbol{\theta})$, we use $p(y(\mathbf{x})|f(\mathbf{x}), \boldsymbol{\theta})$. As $p(y, \mathbf{x}|f, \boldsymbol{\theta}) = p(y|\mathbf{x}, f, \boldsymbol{\theta})\, p(\mathbf{x}|f, \boldsymbol{\theta})$, this implies that we do not model the distribution of the inputs $\mathbf{x}$, setting $p(\mathbf{x}|f, \boldsymbol{\theta})$ to a constant in the likelihood function.

parameters of the likelihood function to the vector of hyperparameters $\boldsymbol{\theta}$. A common assumption for regression is that observations are corrupted by normal noise. The likelihood function is thus $\mathcal{L}(f(\mathbf{x})) = \mathcal{N}(y(\mathbf{x})|f(\mathbf{x}), \sigma^2)$.

With normal noise inference is particularly simple, since the posterior process (1.6) is again Gaussian and can be derived analytically. Assume we have observed N targets $\mathbf{y} = (y_1 \dots y_N)^T$ at inputs $\mathbf{X} = (\mathbf{x}_1 \dots \mathbf{x}_N)^T$, collecting both in the dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$. The posterior process is given by

$$p(f|\mathcal{D}, \boldsymbol{\theta}) \propto p(f|\boldsymbol{\theta}) p(\mathcal{D}|f, \boldsymbol{\theta}) , \tag{3.16}$$

and for an input $\mathbf{x}^*$ the posterior predictive distribution is normal

$$\begin{aligned} p(f^*|\mathbf{x}^*, \mathcal{D}, \boldsymbol{\theta}) &= \mathcal{N}\left(f^*|m(\mathbf{x}^*), v(\mathbf{x}^*)\right) & (3.17a) \\ \text{with mean} \quad m(\mathbf{x}^*) &= \mathbf{k}(\mathbf{x}^*)^T Q^{-1} \mathbf{y} & (3.17b) \\ \text{and variance} \quad v(\mathbf{x}^*) &= k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^T Q^{-1} \mathbf{k}(\mathbf{x}^*) . \end{aligned}$$

To abbreviate notation we have defined $Q = K + \text{diag}[\sigma^2, \dots, \sigma^2]$, and used $\mathbf{k}(\mathbf{x}^*) \in \mathbb{R}^N$ and $K \in \mathbb{R}^{N \times N}$ with $[\mathbf{k}(\mathbf{x}^*)]_\ell = k(\mathbf{x}_\ell, \mathbf{x}^*)$ and $K_{i\ell} = k(\mathbf{x}_i, \mathbf{x}_\ell)$. A more detailed derivation can be found in (Rasmussen and Williams, 2006, Chap. 2).

3.2. Covariance functions

Gaussian processes can be seen as linear models in a feature space, which is defined by the covariance function as a kernel. In both interpretations $k(\mathbf{x}, \bar{\mathbf{x}})$ defines the similarity of function values at two inputs $\mathbf{x}$ and $\bar{\mathbf{x}}$.

In general, the only constraint for covariance functions is their positive definiteness. However, the class of functions which are commonly used for GPs is very limited. Besides classes $k(\mathbf{x}, \bar{\mathbf{x}}) = k(\mathbf{x} \cdot \bar{\mathbf{x}})$, which depend only on the dot product, covariance functions are mostly assumed to be *stationary*, i.e. $k(\mathbf{x} - \bar{\mathbf{x}})$, and *isotropic*, $k(\|\mathbf{x} - \bar{\mathbf{x}}\|)$[5].

stationary

isotropic

Restricting the covariance to one of the above types implies similar assumptions for the latent function f: Stationary covariances encode that the structure of the function is invariant under translations, i.e. f has a similar structure over the whole input space. Isotropy implies a corresponding invariance over directions in the input space.

Besides linear and polynomial kernels, which are of the dot product type, stationary and isotropic covariance structures are practically exclusively used. Gibbs (1997) describes how non-stationary kernels can be defined while ensuring positive definiteness, however, the flexibility of non-stationary models has to our knowledge only been reported to lead to improved results on few specific, and low dimensional datasets[6].

[5] The definition of isotropy depends on the used norm $\|\cdot\|$.

[6] A special case of non-stationary covariance functions is encoded by mixtures of Gaussian processes, which can be interpreted as GP models with a latent extension of the feature space. Please refer to (Pfingsten et al., 2006b) for further details. Other works to discuss non-stationary kernels include (Schmidt and O'Hagan, 2003; Paciorek and Schervish, 2004; Gramacy et al., 2004). The approach by Goldberg et al. (1998) brakes translational invariance by assuming input dependent noise.

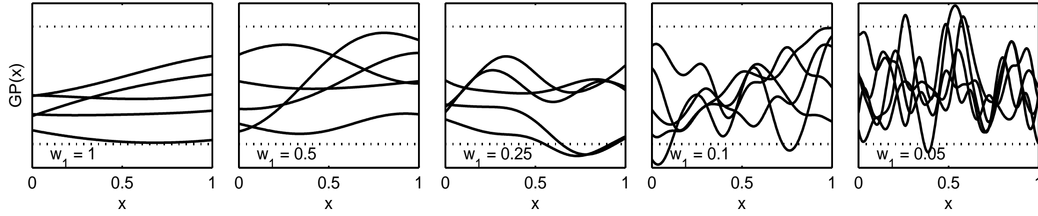


Figure II.3.: Samples from an ARD GP prior with different parameters w_1, which effectively define the scale of the input parameter.

Automatic relevance determination. Isotropic covariance functions are restricted to depend only on the distance $\|\mathbf{x} - \bar{\mathbf{x}}\|$, where the euclidean norm is a natural choice for inputs from $\mathbb{R}^D$. Since the input parameters may live on different scales, the assumption of isotropy might prove too limited. A common choice is to allow for an adequate scaling by considering each dimension d on its typical *length scale* w_d[7] (let $A = \mathrm{diag}(w_1^2 \dots w_D^2)$):

length scale

$$d^2_{\mathrm{ARD}} = \sum_{d=1}^{D} \left(\tfrac{x_d - \bar{x}_d}{w_d}\right)^2 = (\mathbf{x} - \bar{\mathbf{x}})^T A^{-1} (\mathbf{x} - \bar{\mathbf{x}}). \tag{3.18}$$

Find an illustration in figure II.3, where we have plotted samples from a one dimensional GP prior with varying length scale parameters[8].

Distances of the above type are known under the collective term *automatic relevance determination* (ARD), which is due to Neal (1996). However, similar covariance functions have been described earlier by Sacks and Ylvisaker (1966). Welch et al. (1992) already used the length scale parameters to determine the impact of single input parameters: Assume a length scale w_ℓ takes a very large value, so that

automatic relevance determination ARD

$$\frac{|x_\ell - \bar{x}_\ell|}{w_\ell} \ll \frac{|x_i - \bar{x}_i|}{w_i} \qquad \forall i \neq \ell,\ \forall \text{ observed } \mathbf{x}, \bar{\mathbf{x}}\ . \tag{3.19}$$

The dimension ℓ can in this case be neglected in the sum (3.18). Thus it does not enter the covariance function and leaves the posterior process (3.17) invariant. When the length scales are inferred from the data, the ARD mechanism hereby implements an implicit feature selection[9].

[7] We write diag to indicate a diagonal matrix with the elements being the given arguments. In principle we can choose any positive semi-definite matrix A. For instance, Rasmussen and Williams (2006) consider $A^{-1} = \mathrm{diag}(w_1^{-2} \dots w_D^{-2}) + \Lambda^T \Lambda$ with $\Lambda \in \mathbb{R}^{D \times k}$, $k < D$, which they call the *factor analysis distance*. It serves to identify linear combinations of inputs, which are particularly relevant.

[8] We have used a GP with the common squared exponential covariance function (3.20). The dotted lines indicate the prior signal amplitude as a 2σ confidence interval, given by the parameter v. In solid lines we show five samples from the prior.

[9] In chapter V 2.2 we discuss feature selection in more detail.

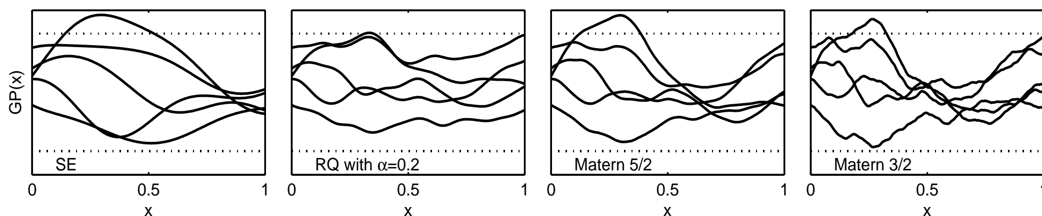


Figure II.4.: Samples from GPs with various kernels. The covariance function defines the characteristics of the corresponding GP, such as its smoothness (i.e. its differentiability). Note that the smoothness is fundamentally different from the typical length scale as in ARD.

Smoothness assumptions. In the above paragraph we have introduced ARD, which implements an automatic scaling of the data to obtain an adequate distance measure d_{ARD}. This measure can be used with a variety of covariance functions which specify how the correlations between function values evolve with the distance.

The structure of the covariance function eventually defines the characteristics of the corresponding GP, such as its differentiability which we refer to as the *smoothness*. In this sense, the ARD length scale parameters do not affect the smoothness of the GP. What the length scale parameters do control is by how much the GP may vary within a given interval, which we distinguish by referring to it as the *variability*.

smoothness

variability

For an overview over common choices for kernel functions see (Stein, 1999) or (Rasmussen and Williams, 2006). A basic result, as given by Abrahamsen (1997), is that the smoothness of a Gaussian process directly corresponds to the smoothness of its covariance function. In short, a Gaussian process is as many times *mean square* (MS) *differentiable* as the covariance function in $d = 0$. The corresponding theorems can be found in appendix A.

mean square differentiable

In the following we introduce the most common stationary covariance functions for GPs, using the symbol d to indicate some distance measure. Figure II.4 illustrates the effect of the kernels, showing samples from the corresponding GP priors[10].

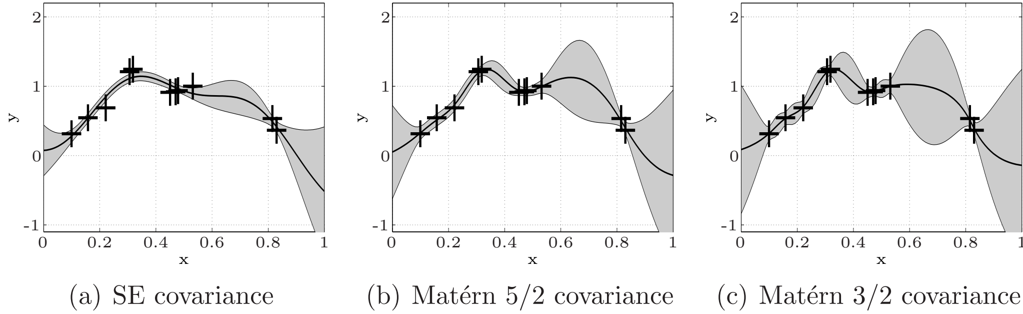
Squared exponential (SE) covariance function.
The SE kernel

squared exponential kernel

$$k_{\mathrm{SE}}(d) = v^2 \exp\left[-\tfrac{1}{2}d^2\right] \tag{3.20}$$

is probably the most commonly used kernel in machine learning. As it is infinitely often differentiable, it leads to GPs which are infinitely often MS differentiable. The power spectrum of the SE kernel has again Gaussian shape, and therefore only covers a restricted band of frequencies. Hence, the SE kernel encodes the assumption that f is very smooth, and the model can be considered little flexible.

[10] We are plotting samples from one-dimensional ARD GP priors, setting $w_1 = 0.25$. The common factor v^2 indicates the signal variance, whose 2σ interval is indicated by dotted lines.



(a) SE covariance (b) Matérn 5/2 covariance (c) Matérn 3/2 covariance

Figure II.5.: Posterior predictive distribution for different covariance functions. Plotted are observations (+), mean (–) and 2σ confidence intervals (gray) of the predictive distribution. The covariances encode priors that include functions which are only once (c) or twice (b) differentiable, or exclusively functions which are infinitely often differentiable (a). The plots show that the mean function is smooth for all cases, and the smoothness mainly influences how fast the uncertainty increases in between observations.

rational quadratic kernel

Rational Quadratic (RQ) covariance function. The RQ kernel is defined as

$$k_{\mathrm{RQ}}(d) = v^2 \left(1 + \frac{d}{2\alpha}\right)^{-\alpha} \tag{3.21a}$$

$$\propto \int \mathrm{d}\tau \; \tau^{\alpha-1} \exp\left(-\alpha\tau\right) \exp\left(-\tfrac{1}{2}\tau d^2\right) \; . \tag{3.21b}$$

As we have indicated in (3.21b), it can be seen as an infinite mixture of SE kernels with different length scales. By changing α, we can adjust the range of contributing length scales, and therefore change he flexibility of the RQ kernel. Nevertheless, for all α the RQ kernel comprises the assumption of MS differentiability to any order. For $\alpha \to \infty$ we effectively constrain the superposition and recover the SE kernel.

Matérn kernels

Covariance functions of the Matérn (MA) form. The Matérn-class of covariance functions is given by

$$k_{\mathrm{MA}}(d) = v^2 \, \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}d\right)^{\nu} K_\nu \left(\sqrt{2\nu}d\right) \; , \tag{3.22}$$

where K_ν is a modified Bessel function (Stein, 1999). The parameter $\nu \in \mathbb{R}^+$ controls the smoothness of the corresponding process: The GP is up to ν times differentiable, and k_{MA} approaches the SE kernel in the limit $\nu \to \infty$. For $\nu = \frac{3}{2}$ and $\frac{5}{2}$ the Matérn kernel can be handled analytically. The corresponding process is once $(\frac{3}{2})$ and twice $(\frac{5}{2})$ MS differentiable.

Illustration: The effect of the ARD parameters is intuitively clear: They implement an automatic normalization, which assigns a "natural length scale" to each input dimension. The effect is shown by figure II.3, where the ARD parameter w_1 effectively zooms into the x-axis.

In comparison, the assumptions which correspond to the choice of a particular covariance function are not so apparent. The standard procedure in machine learning is thus to try a few kernels, comparing their performance in a model selection scheme.

Figure II.4 shows samples from the GP prior with different kernels, to illustrate typical functions covered by the model. To exemplify the models' effect on the results of an inference problem, we show corresponding predictive distributions in figure II.5[11].

We observe two main aspects: On the one hand, fast fluctuations of rough functions in the prior are averaged out in the predictive mean. Thus, the use of highly flexible functions can be sensible even in the case of few observations. The predictive variance, on the other hand, is strongly influenced by the smoothness assumptions. As we allow for more flexibility, the uncertainty in between observations increases faster than for smooth models. Especially the SE kernel is known to underestimate the uncertainty by implying unrealistic strong assumptions about the functions' smoothness (Stein, 1999).

[11] As for figure II.4 we have set $w_1 = 0.25$, the signal and noise level are $v = 1$, $\sigma = 0.05$.

III. Robust designs for mass production

Simulation software is now extensively used to explore the behavior of complete physical systems, replacing expensive experiments in industrial engineering. Even though computer models can be constructed efficiently using commercial tools, the interpretation of the results remains an intricate problem: The models have tens of parameters and often require considerable time for evaluation. Hence, it is hard to grasp the behavior of the system and to explore the model's response interactively.

As we have outlined in chapter I 2, an important motivation for using simulations in the design for mass production is to analyze the design's robustness against process tolerances. When process tolerances cannot be considered small, as it is the case for MEMS and ICs, a device's functionality has to be validated over the whole distribution of parameter settings, which is given by the typical fluctuations in production.

In this chapter we describe the results of a project to create a software package for model-based design analysis and optimization[1]. The starting point of such analysis are known process tolerances and a computer program which simulates the device. Our software provides fast sensitivity analysis for varying parameter settings and an efficient gradient-based design optimization to automate re-specification. The focus of our approach lies on the efficient use of simulation runs to make design analysis feasible for computationally expensive models.

Our procedure is based on the use of an efficient emulator for the simulation software, which eases both, manual exploration and statistical design analysis. We use Gaussian process (GP) regression to ensure sufficient flexibility for nonlinear and high dimensional models, and to make efficient use of expensive simulation runs. We define several measures for the importance of linear and nonlinear effects, which let the designer grasp the global structure of a model at one glance. Design analysis necessarily involves a global average over the response of the computer model, which is usually difficult to compute. However, since these averages can be computed in closed form from the GP emulator, our approach renders possible an efficient gradient-based robust optimization. The accuracy of the results can be assessed using standard validation methods such as cross validation.

GPs have previously been proposed for efficient sensitivity analysis, and the main contribution of this work has been to derive new sensitivity measures for

[1] The software is a joint project of Benjamin Sobotta, Daniel Herrmann and Tobias Pfingsten at CR/ARY together with AE/EST4.

process fluctuations. The robust optimization scheme, which is proposed in this work, is significantly different from previous approaches, and it is the first to explicitly account for the distribution of the process tolerances while being feasible for computationally expensive models. Our approach has previously been described in (Pfingsten et al., 2006a).

In section 1 we introduce the concept of robust designs to attenuate the effect of random fluctuations in production (1.1). We review common approaches to design analysis in 1.2 and discuss them in the light of the requirements for day-to-day use in industrial engineering. In relation to the state-of-the-art we outline our approach and discuss the contribution of this work.

We propose a number of sensitivity measures in section 2, deriving them in 2.1 and exemplifying their interpretation (2.2) on the analysis of a MEMS sensor. To compute the sensitivity measures efficiently, we use the Bayesian Monte Carlo (BMC) approach, which we describe in section 3. Relating BMC to classical quadrature and Monte Carlo (3.1), we explain the BMC scheme in 3.2 and our approach to automated design optimization in 3.3. To keep the argumentation clear, we have moved technical details to the appendix B.

We have validated our approach on a number of benchmark problems from literature and fully featured models of MEMS, presenting the results in section 4. A discussion is given by section 5.

1. Process tolerances and robust designs

Before describing our approach to design analysis, we introduce the problem setup and previous approaches in this section. In section 1.1 we outline the basic terminology and the aim of the analysis, using a simple example to motivate the generic considerations. We review previous works which treat design analysis and optimization in section 1.2, working out where our approach comprises novel results.

1.1. Fluctuations and specifications

Process tolerances. Complex physical systems such as MEMS are specified by a large number of parameters, including material properties and geometrical dimensions. Besides such internal parameters, other quantities such as the temperature might have a substantial influence on the behavior of the device. We collect all parameters in $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^D$ and denote the deterministic response by $f(\mathbf{x}) \in \mathbb{R}$. For simplicity we write a possibly noisy observation of $f(\mathbf{x})$ as y.

As we can hardly know all parameters exactly, we are necessarily left with some uncertainty about the response even if the model is perfect. In the Bayesian framework the uncertainty in the input parameters is described by an *input distribution*

input distribution

$$p(\mathbf{x}), \tag{1.1}$$

which reflects the subjective knowledge about the parameters.

When thinking of mass production, $p(\mathbf{x})$ also reflects a probability in the sense of a repeated experiment: as the processes are subject to random fluctuations, the parameters vary from device to device. The input distribution is usually known for standard processes and describes fluctuations of a process which is not subject to sudden changes or systematic drifts. We refer to these variations as *process tolerances*. Industrial processes are mostly defined by a *nominal parameter* $\hat{\mathbf{x}}$, around which the actual parameters fluctuate.

process tolerances

nominal parameters

The uncertainty distribution. Since the input parameters $\mathbf{x}$ are random variables, so are $f(\mathbf{x})$ and the corresponding measurement y. The distribution of the output—commonly called the *uncertainty distribution*—is given via the mapping $f(\mathbf{x})$ and the input distribution $p(\mathbf{x})$:
The cumulative distribution function (CDF) for y is given by

uncertainty distribution

indicator function

$$p_{\mathbf{x}}(y \leq a) = \int_{\mathcal{X}} \mathrm{d}\mathbf{x}\, p(\mathbf{x})\, \mathbb{I}[f(\mathbf{x}) \leq a] \,, \tag{1.2}$$

where $\mathbb{I}$ denotes the indicator function. If it exists, the density function[2] is given by $p_{\mathbf{x}}(y) = \frac{\partial}{\partial a}\mathrm{CDF}(a)\,|_{a=y}$. We use the subscript $\mathbf{x}$ to indicate that the distribution is induced by the uncertain inputs.

The term *uncertainty analysis* collects the methods to derive $p_{\mathbf{x}}(y)$. It is generally hard to obtain the uncertainty distribution, in particular when the function $f(\mathbf{x})$ is only known via an expensive computer code.

uncertainty analysis

Example: Stone pitch (part 1) To illustrate the idea of uncertainty analysis we consider a simple low-dimensional problem.

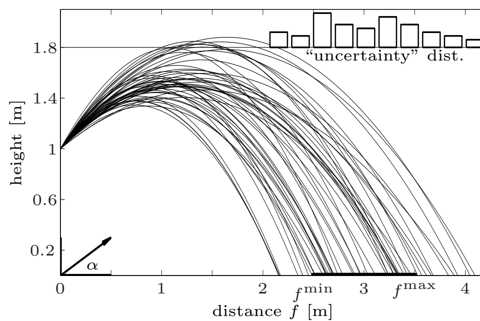


Figure III.1.: Stone pitch. Trajectories for several angles α and initial velocities v_o.

Assume we shoot a projectile, such as a stone or golf ball, with initial velocity v_o and angle α from 1m height. The setup is shown in figure III.1. Neglecting friction, the stone's trajectory can readily be computed, including the distance f where it hits the ground (Demtröder, 1994, Chap. 2). The goal is to hit an interval $[f^{\min}, f^{\max}] = [2.5, 3.5]$. Assume the pitch is not perfect, resulting in Gaussian noise around the chosen α and v_o (standard deviation 4° and $\frac{1}{2}$m/s). The figure shows 50 trajectories around the nominal value $(\hat{\alpha}, \hat{v}_o) = (45^o, 4.8\text{m/s})$.

Accordingly, the target is not always hit and the stone strikes the ground at varying distances. In the plot we show a histogram to indicate the uncertainty distribution of the resulting distances $f(v_o, \alpha)$.

[2]For notational simplicity we assume in the following that the density function $p_{\mathbf{x}}(y)$ exists. However, we only use its first and second moment which are guaranteed to exist since the support of $p(\mathbf{x})$ and f are bounded for all real production process.

Feasibility region and parametric yield. Concerning computer-aided design, the nominal input parameters $\hat{\mathbf{x}} \in \mathcal{X}$ are usually set to ensure that one or several responses $f_1(\mathbf{x}) \dots f_L(\mathbf{x})$ meet specific requirements, such as

$$f_\ell(\mathbf{x}) \in [f_\ell^{\min}, f_\ell^{\max}] \,. \tag{1.3}$$

feasibility region

The region $\mathcal{F} \subset \mathcal{X}$ where all constraints are met is called the *feasibility region*. While it is relatively simple to find some point from $\mathcal{F}$, one has to make sure that most area under $p(\mathbf{x})$ falls within $\mathcal{F}$ when process tolerances cannot be neglected.

parametric yield

The *parametric yield* measures the fraction of devices which meet the requirements despite process tolerances[3]. Using the input distribution one obtains

$$\text{Yield} = \int_{\mathcal{F}} \mathrm{d}\mathbf{x}\, p(\mathbf{x}). \tag{1.4}$$

Maximizing the parametric yield is the most important objective in design optimization. One possibility is to minimize the impact of process tolerances by adjusting the nominal values $\hat{\mathbf{x}}$. Tackling the process tolerances themselves often implies an investment in new machinery or more careful processing. In those cases one has to balance yield gain and resulting costs.

EXAMPLE: STONE PITCH (PART 2) Figure III.2 shows a contour plot of the distance where the stone hits the ground as a function of (α, v_o).

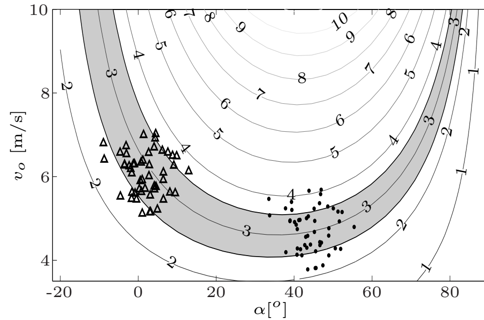


Figure III.2.: Stone pitch. Feasibility region and contours for $f(\alpha, v_o)$ [m].

In our example the angle can be adjusted arbitrarily, and v_o can be chosen between 0 and 10 m/s. The feasibility region $\mathcal{F}$ (gray) indicates when the target $[f^{\min}, f^{\max}]$ is hit. The •s correspond to the trajectories in figure III.1 around $\hat{\alpha} = 45^o$ and $\hat{v}_o = 4.8$ m/s.

Note that some trials miss the target due to the fluctuations, even though the nominal values are set to hit the target at 3.1m. From the shape of the feasibility region one can argue that the shallow region at large angles α leads to low success rates, while the uncertainty in v_o has less impact when α is small. A design optimization has to account for these effects by considering the shape of $p(\mathbf{x})$ and the feasibility region. The optimal setting[4] is $(\hat{\alpha}, \hat{v}_o) = (2.5^o, 6.1\text{m/s})$, where the $\triangle$s indicate 50 additional samples. The yield at the optimal setting is by 7% larger than at the original nominal value.

[3]The parametric yield does not include *catastrophic failures* which are not related to natural process variations. A typical example for catastrophic failures in semiconductor manufacturing is failure due to particle contamination.

[4]We have used the numerical optimization scheme as described in section 3.3.

1.2. Approaches to design analysis

General difficulty. In the preceding section 1.1 we have seen that a design analysis has to account for the shape of the feasibility region as well as for the shape of the input distribution. The yield (1.4), for example, is the integral of $p(\mathbf{x})$ over $\mathcal{F}$.

Designs which are analyzed in modern engineering contain tens or even hundreds of parameters. Hence, the integrals are very high dimensional and generally hard to solve—we discuss common numerical methods in section 3. When dealing with computationally expensive models, such as circuit simulations or geometric finite element models, the bottleneck is given by the number of simulation runs: If one evaluation of the model runs minutes or even hours, the results need to be used efficiently to make integration feasible.

Requirements for industrial engineering. Besides the capability to assess single settings, the circumstances in engineering lead to some additional requirements for a useful approach to design analysis:

1. Running a batch of simulations—e.g. outside working hours—is cheaper than doing so interactively: while computation time is relatively cheap, it is inconvenient to have the user wait for a result during analysis. Also, expensive licenses for simulation software are usually short during the day while unused at night.

2. Simulation runs should be recycled for various analyses: the software should let the engineer explore the design quickly, i.e. without re-running the simulations for each setting. This includes simple features like plotting projections to one or two axes.

3. The results of the analysis need to be reduced to few expressive figures: besides the parametric yield, one is interested in the influence of single parameters, in interactions of inputs and in the degree of linearity.

4. The analysis has to be reliable by allowing for the validation of the SA. Besides, the software should provide the means for validating the computer model itself, e.g. by detecting outliers due to numerical instabilities.

State of the art. Computer-aided optimization of integrated circuits has been described as early as 1967 by Temes and Calahan, and since then network models have been used extensively in the designing process. Brayton et al. (1981), Bandler and Chen (1988) and Director et al. (1993) review several techniques to optimize the parametric yield.

The simplest approach to compute the yield is the *Monte Carlo* (MC) method, where simulations at random samples from $p(\mathbf{x})$ are used to approximate (1.4) directly. An application can be found in (Johnson et al., 1999). The method is exact in the limit of a large number of samples, however, it might require many runs for convergence—in particular in combination with

Monte Carlo

an optimization scheme. Thus, most advanced methods are constructed to reduce the computational costs of the MC method.

geometrical design centering

A popular technique is *geometrical design centering*, as covered by Low and Director (1991) and Antreich et al. (1994). Instead of directly approaching the expression for the parametric yield (1.4), these methods replace the problem by a related geometrical setup: the nominal value $\hat{\mathbf{x}}$ is placed at a maximal distance to the boundary of the feasibility region[5]. These approaches are efficient since they evaluate $f(\mathbf{x})$ only at few points close to the boundary. However, they are mainly suited for situations where the yield is nearly 100%, as the distance only indicates critical directions—geometrical approaches do not provide reliable estimates for the yield.

response surface

Response surface (RS) methods are used to speed up computations by replacing the original code by some simpler approximation. See (Myers and Montgomery, 2002). When used in optimization, RSs are usually local linear or quadratic approximations. Especially for high dimensional input spaces such parametric models fail or become computationally unattractive (Li et al., 2005).

The problem can be solved by using more flexible, nonparametric models: Zaabab et al. (1995) and Rayas-Sánchez (2004) report on the use of artificial neural networks. Sacks et al. (1989a,b) and Currin et al. (1991) propose to use Gaussian processes to interpolate between single runs of computer models. O'Hagan in particular promoted the use of GP "emulators" in a Bayesian analysis of computer experiments[6].

Design centering approaches analyze and optimize given models with respect to their overall robustness. However, to improve the very structure of the design, the engineer needs to understand the influence and interaction of the input parameters. By decomposing the output uncertainty into specific effects, a *sensitivity analysis* (SA) uncovers this information in a compressed form.

sensitivity analysis

Saltelli et al. (2000a,b) review the concept of SA and a number of measures for the impact of input parameters. A large number of sensitivity measures can be found in literature, where a first distinction is made between *local* and *global* sensitivity measures:

local/global sensitivity measures

- Local measures are computed around the nominal value, mostly based on derivatives. They can be used to measure the impact of small disturbances and do not reflect the shape of the input distribution.

- Global measures need to be employed when the support of $p(\mathbf{x})$ cannot be considered small with respect to the variability of $f(\mathbf{x})$. As defined by Saltelli et al. (2000a), global measures need to reflect size and shape of the input distribution and are therefore based on averages over $p(\mathbf{x})$.

[5] The choice of the distance measure needs to reflect the shape of $p(\mathbf{x})$. A Gaussian distribution $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\hat{\mathbf{x}}, B)$, for example, corresponds to the distance $d(\mathbf{x}, \hat{\mathbf{x}}) = \mathbf{x}^T B^{-1}\hat{\mathbf{x}}$. Corresponding distances exist only for a restricted class of distributions.

[6] O'Hagan has been involved in a number of publications on this subject, including a guide for practioners: (O'Hagan et al., 1998; Haylock and O'Hagan, 1996; Oakley and O'Hagan, 2002, 2004; O'Hagan, 2004; Conti et al., 2004; Kennedy and O'Hagan, 2001; Kennedy et al., 2004).

A local linear approximation can be computed with $1 + D$ function evaluations, while the computation of a global average involves the exploration of the support of $p(\mathbf{x})$ and thus requires a large number of samples. The approximations described in literature can be seen as trade-offs between accuracy and complexity (Morris, 2004):

- Local linear measures can be generalized by using local parametric fits (response surfaces) of higher order.
- Assuming that f is an additive function of the input parameters, one can simplify the global analysis by considering projections onto single parameters. See e.g. Classen et al. (2004).
- Using feature selection, SA can be reduced to identifying inactive parameters. Welch et al. (1992) proposes to use the ARD capability of GPs (see chapter II 3.2). These qualitative *screening* methods make with few simulation runs, however, they do not quantify the importance of input parameters.

screening

Contribution of this work. The aim of this work was to create a software tool for design analysis and optimization at Robert Bosch GmbH. To be valuable for regular use by the designers, such tool has to fulfill the requirements mentioned on page 43:

To account for requirements 1 and 2 (fast interactive use and recycling of data) we have chosen a response surface scheme. As process tolerances are generally not small and responses are usually high dimensional nonlinear functions, the RS needs to be a flexible nonparametric fit. The emulator can be tested using standard techniques such as cross validation and thus provides a reliable estimate for the accuracy of the results (requirement 4).

We use Gaussian processes (GPs), which have proven to be efficient models for regression and interpolation. Their structure eases the derivation of analytical expressions for sensitivity measures, and their ARD capability automatically provides a screening mechanism. As the sensitivity measures can be computed analytically from the RS, an automatic optimization becomes extremely efficient, even accounting for process tolerances.

As mentioned above, the use of GPs in the context of sensitivity analysis has previously been reported on. However, they are hardly known outside the statistics and machine learning community, and have not been applied previously in design analysis for mass production. The robust optimization scheme, presented here, is significantly different from previous approaches, analytically incorporating the distribution of process tolerances using a global GP response surface. We believe that only the presented approach makes robust optimization feasible when expensive computer models are used.

As GPs are probabilistic models, they provide a notion of predictive uncertainty. This allows for the use of active learning—also known as experimental design—to explore the support of $p(\mathbf{x})$ efficiently. We discuss this aspect in chapter IV.

GPs encode a rich class of functions, however, it is not clear how they perform on real-world tasks from development. We present an extensive empirical study, where we analyze the convergence rate of the GP-based approach in relation to the Monte Carlo method.

Most work on SA concentrates on input distributions which encode uncertainty in the inputs. This work is the first to explicitly treat process tolerances where the nominal setting plays a salient role, deriving statistically justified global sensitivity measures for this setting. To ease the interpretation of high dimensional models we propose novel measures for the degree of nonlinearity and non-additivity over $p(\mathbf{x})$ (requirement 3).

2. Sensitivity analysis for design validation

Above we have outlined our approach to design analysis, where the efficient use of simulation runs plays a major role. In this section we introduce and motivate a number of sensitivity measures, the actual output of our tool for design analysis. The methods to compute these measures are subject of the following section 3.

We derive a number of sensitivity measures in 2.1, which are constructed to give a compressed overview over the structure of the computer model. Section 2.2 can be seen as a manual for practioners, where we exemplify the interpretation of the results via the characteristics of a pressure sensor.

2.1. Sensitivity measures

Our approach focuses on variance-based measures, which are commonly used for sensitivity analysis. Using the variance implies the assumption that the uncertainty distribution is approximately Gaussian. However, the variance is also a natural measure when considering a single number to characterize the width of a distribution.

Local measures for small disturbances. For models which are approximately linear over the support of $p(\mathbf{x})$,

$$f(\mathbf{x}) \approx f_{\text{lin}}(\mathbf{x}) = a_o + \sum_{\ell} a_\ell x_\ell \,, \tag{2.5}$$

standardized regression coefficients

the *standardized regression coefficients* (SRCs) are a common measure for sensitivity: assume that the inputs are uncorrelated and normally distributed,

$$p(\mathbf{x}) = \prod_{\ell} p_\ell(x_\ell) = \prod_{\ell} \mathcal{N}(x_\ell | \hat{x}_\ell, \sigma_\ell^2). \tag{2.6}$$

The output distribution $p_{\mathbf{x}}(f_{\text{lin}})$ is in this case also normal, and the variance can be decomposed into independent contributions from each input parameter,

$$\text{var}_{\mathbf{x}}\left[f_{\text{lin}}\right] = \sum_{\ell} a_\ell^2 \sigma_\ell^2 \,. \tag{2.7}$$

The SRCs are defined as the shares of the variance due to fluctuations in single parameters:

$$\mathrm{SRC}_\ell = \tfrac{1}{\mathrm{N}_f}\Big(\mathrm{var}_{\mathbf{x}}\Big[f_{\mathrm{lin}}\Big|\text{fix all inputs but } x_\ell\Big]\Big) = \tfrac{1}{\mathrm{N}_f}\Big(a_\ell^2\sigma_\ell^2\Big) \;, \tag{2.8}$$

where N_f is an arbitrary normalizing constant.

Generalization to nonlinear models. The SRCs can easily be generalized to nonlinear models f by defining *local correlation ratios* as

local correlation ratios

$$\mathrm{LCR}_\ell = \tfrac{1}{\mathrm{N}_f}\Big(\mathrm{var}_{\mathbf{x}}\Big[f\Big|\text{fix all inputs but } x_\ell \text{ to their nominal value}\Big]\Big) \;, \tag{2.9}$$

where it is no longer assumed that f is linear. Note that the definition is slightly different from (2.8): As f is no longer assumed to be additive, the value of fixed parameters becomes relevant. While the contributions from each input are independent when the model is additive, mixed terms can in general lead to interactions. Consider for example a term $\mathbf{x}_i\mathbf{x}_\ell$. The impact of $\mathbf{x}_\ell$ is zero for $\mathbf{x}_i = \hat{\mathbf{x}}_i = 0$, while it has nonzero influence for other $\mathbf{x}_i$.

In terms of the input distribution, mean and variance of the uncertainty distribution are given by

$$\mathrm{mean}_{\mathbf{x}}\,[f] \;=\; \int \mathrm{d}\mathbf{x}\; p(\mathbf{x})\; f(\mathbf{x}) \tag{2.10a}$$

$$\mathrm{var}_{\mathbf{x}}\,[f] \;=\; \int \mathrm{d}\mathbf{x}\; p(\mathbf{x})\; f^2(\mathbf{x}) \quad - \mathrm{mean}_{\mathbf{x}}^2[f] \;. \tag{2.10b}$$

From the definition of LCR_ℓ in (2.9) we see that the integral is taken along the axis of input parameter x_ℓ. Hence, the function's behavior off the x_ℓ-axis is disregarded and the LCRs are local measures.

Cross terms: global measures. To account for the cross effects off the axes we use the *correlation ratios*[7],

correlation ratios

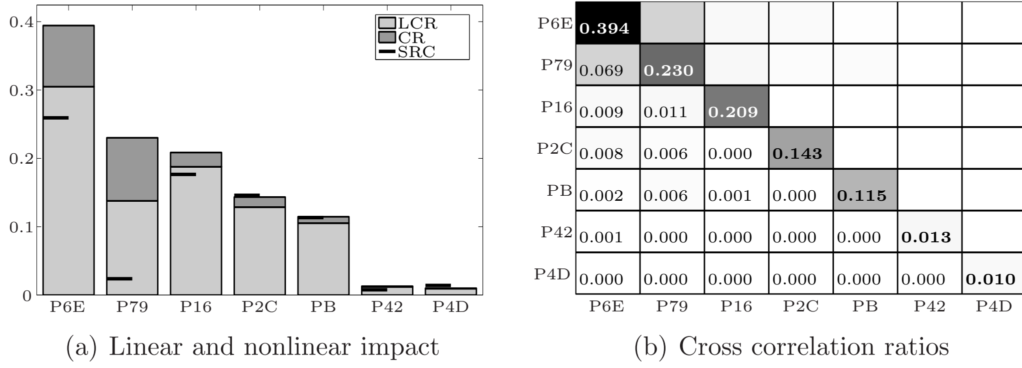
$$\mathrm{CR}_\ell = \tfrac{1}{\mathrm{N}_f}\Big(\mathrm{var}_{\mathbf{x}}\,[f] - \mathrm{var}_{\mathbf{x}}\,[f|x_\ell = \hat{x}_\ell]\Big) \;. \tag{2.11}$$

These are true global measures as the average is taken over the complete support of $p(\mathbf{x})$.

The interpretation of our measures is simple: The local LCR_ℓ expresses the width of the uncertainty distribution when *only* the input x_ℓ fluctuates, with all other parameters being perfectly controlled to their nominal values. In contrast, CR_ℓ measures by how much the fluctuations in the output can be reduced by a perfect control of x_ℓ —correctly considering interactions and fluctuations in all other parameters. Thus, the difference between the two measures the model's degree of non-additivity.

[7]Our correlation ratios CR relate to the expected importance measure $\frac{\mathrm{var}_{\mathbf{x}}[f]-\mathrm{E}_{x_\ell}[\mathrm{var}_{\mathbf{x}}[f|x_\ell]]}{\mathrm{var}_{\mathbf{x}}[f]}$ (Iman and Hora, 1990). The expectation measure is used when nominal values are not naturally given as for process in mass-production.

PSfrag



(a) Linear and nonlinear impact

(b) Cross correlation ratios

Figure III.3.: Design analysis for the pressure sensor (PS model): The most influential design parameters and their impact are shown in panel (a). The correlation ratios CR_ℓ include all effects, their local counterparts LCR_ℓ are computed along the x_ℓ-axis to exclude cross terms. SRC_ℓ is based on a linear approximation. A combination of all three extracts the prevailing structure in the model. The cross correlation ratios CCR are displayed in panel (b) to extract pairwise interactions, where the matrix is given by numbers and gray shades.

cross correlation ratios

To be able to assess the strength of interactions between pairs of parameters we define *cross correlation ratios*

$$\mathrm{CCR}_{i\ell} \;=\; \mathrm{CR}_i - \mathrm{CR}_i(x_\ell = \hat{x}_\ell) \;=\; \mathrm{CCR}_{\ell i}\;, \tag{2.12}$$

which reveal how strongly two parameters are linked. The definition of the cross terms is intuitively clear: they measure the change in the influence of parameter x_ℓ as we fix parameter x_i to its nominal value. Where mixed terms can be neglected, we have $\mathrm{CR}_i = \mathrm{CR}_i(x_\ell = \hat{x}_\ell)$ and the cross terms are zero. If two parameters x_i and x_ℓ are maximally correlated we have $\mathrm{CCR}_{i\ell} = \mathrm{CCR}_{\ell\ell} = \mathrm{CR}_\ell = \mathrm{CCR}_{ii} = \mathrm{CR}_i$.

In our design analysis we use plots which combine all four measures, SRC (2.8), LCR (2.9), CR (2.11) and CCR (2.12) to obtain an overview over the effects which dominate the model. As normalization constant N_f we choose the total variance

$$\mathrm{N}_f = \mathrm{var}_{\mathbf{x}}[f]\;. \tag{2.13}$$

In some applications it might be convenient to choose another scale, e.g. the squared width of a specification interval as in (1.3).

2.2. Interpretation and use in practice

The following case study illustrates the use of the proposed sensitivity measures to extract the basic properties of a simulation model. We have performed the analysis using the Bayesian Monte Carlo method, which is explained in detail in section 3. The case study reproduces the design analysis of an electromechanical pressure sensor, in development at Robert Bosch GmbH:

pressure sensor

PS model: Pressure Sensor. The model stems from the design analysis of a pressure sensor (PS) and covers all relevant mechanical and electrical properties of the system. A finite element model of the mechanical configuration reproduces the deformation of the device due to the applied pressure. The mechanical module has a number of parameters which represent the geometrical dimensions, the deformations are in turn converted into electrical signals. The output of the model is a temperature and pressure dependent electrical signal, for which a last module calculates significant characteristics such as the accuracy of the device. The model has in total 28 parameters, for which typical tolerances are known.

One model run requires several minutes on a modern CPU. An exhaustive MC analysis requires thousands of function evaluations and can therefore not be under consideration for a variety of design alternatives: According to the probabilistic error bound of the MC method we would need 5000 samples for an accuracy of 1.5% in the mean estimate.

Instead, the Bayesian Monte Carlo approach uses a Gaussian process meta-model, which is trained and tested on comparably few simulation runs: We have used a 500 points-Latin Hypercube design (see app. B) from $p(\mathbf{x})$ to obtain training samples and a separate test set of 1000 samples to estimate the model accuracy. The square root of the mean squared error on the test set was 1.3% of the standard deviation of the output, thus ensuring a good accuracy of the GP-meta model. One can easily verify that the accuracy of the estimated CRs is of the same order as the maximal squared error of the regression model.

Note that we have anonymized the model by renaming all input parameters.

Design analysis. The plots in figure III.3 combine all sensitivity measures to give a condensed and comprehensive overview on the nature and impact of the model's parameters. We have restricted the plot to the most influential parameters:

The difference between the CRs and SRCs in panel (a) indicates the importance of nonlinear effects, which are apparently responsible for most variation due to parameter P79. The lower part of the bars, shaded in a lighter gray, shows the local correlation ratios LCR. The difference to the global CRs indicates the part of the correlation which is induced by the joint variation with other parameters, i.e. what we have called cross effects.

These cross effects are broken down by the symmetric CCR matrix, which is shown in plot (b). Shaded in gray we find the CRs on the diagonal and the interdependencies between pairs of parameters off the diagonal. Note, for example, that the first two parameters, P6E and P79, interact strongly.

3. Bayesian Monte Carlo for design analysis and optimization

The concern of this section is how the sensitivity analysis can be performed accurately on the basis of few simulation runs, which basically corresponds to finding efficient numerical approximations to high dimensional integrals: As we have seen, global measures for sensitivity are some kind of average over the joint distribution of input parameters $p(\mathbf{x})$,

$$I[f] \;=\; \int \mathrm{d}\mathbf{x}\, p(\mathbf{x})\; F[f(\mathbf{x})]\,, \tag{3.14}$$

where F denotes some functional of the output f.

Traditionally, classical quadrature rules are used in low dimensions to solve these integrals and (Quasi-) Monte Carlo (MC) methods are applied in higher dimensions. We introduce both in section 3.1. The Bayesian Monte Carlo (BMC) method can be seen as an extension to classical quadrature, where the integrand is modeled using a Gaussian process prior. We introduce the BMC scheme and its application to SA in 3.2. Using the BMC approach, an efficient design optimization can be realized, which correctly incorporates statistical fluctuations. We outline the optimization scheme in 3.3.

3.1. Monte Carlo methods and classical quadrature

Classical quadrature rules show good convergence properties in one dimension, however, they are not applicable for high dimensional integrals: The error of the trapezoidal rule scales as $\mathcal{O}(N^{-2/D})$—for $F \in \mathcal{C}^2$ and N being the number of nodes. As the dimension of the integral in (3.14)—the number of model parameters—is typically very high, this behavior makes classical methods inapplicable for our purposes.

Monte Carlo

Monte Carlo (MC) methods lead to a probabilistic error bound of $\mathcal{O}(N^{-1/2})$ which is independent of the input dimension (Niederreiter, 1992). The basic idea of Monte Carlo methods is to draw a finite number of N samples $\mathbf{x}_1 \ldots \mathbf{x}_N$ from $p(\mathbf{x})$ and to use the empirical mean

$$I[f] \;\approx\; \tfrac{1}{N}\sum_{\ell} F[f(\mathbf{x}_\ell)] \tag{3.15}$$

as an unbiased estimator of the expectation in (3.14)[8]. The average error and the probabilistic bound are guaranteed by the strong law of large numbers and the central limit theorem for any square integrable $F[f]$.

Quasi-Monte Carlo

discrepancy

Latin Hypercube

The simple MC method uses independent random samples from $p(\mathbf{x})$. More sophisticated *quasi-Monte Carlo* methods use sampling schemes which lead to improved space filling, as it can be shown that the convergence rate is related to the *discrepancy*—a measure for space filling. Common designs are Sobol lattices (Sobol, 1993) and the popular *Latin Hypercube* design (see appendix B).

[8] To compute the CRs and LCRs using the MC method one may draw samples from $p(\mathbf{x})$ to compute the complete variance, subsequently recomputing the function with the corresponding parameters being set to their nominal value.

An overview over quasi-MC methods is given by Niederreiter (1992).

The error bound $\mathcal{O}(N^{-1/2})$ for MC methods holds for a very broad class of functions, requiring only square-integrability. While this can be seen as an advantage, it is clear that for highly regular functions fewer nodes should be necessary to approximate the integral than for irregular functions, and it can be worthwhile to reflect this regularity in a quadrature rule.

3.2. Bayesian Monte Carlo

Generic idea. Monte Carlo methods—including improved quasi-MC methods like Latin Hypercube—directly estimate the uncertainty distribution using empirical sums (3.15). The Bayesian MC method uses an indirect estimate, where the underlying function f is modeled using a GP. Using such model, the available simulation runs can be used efficiently to approximate the function, and all measures can subsequently be computed using this approximation:

Algorithm 1 Bayesian Monte Carlo

Require: Generated dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$.

1: **GP regression.** Use the data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ to compute an estimate $p(f|\mathcal{D}, \boldsymbol{\theta}^*)$.

2: **Verification.** Verify the GP assumptions using CV or a separate test set.

3: **Sensitivity analysis.** The posterior distributions to all integrals $I[f]$ can be computed efficiently from $p(f|\mathcal{D}, \boldsymbol{\theta}^*)$.
In particular, all measures LCR, CR, CCR can be computed in closed form.

The underlying idea is relatively general and does also apply to other frameworks: the available data are used to construct a GP emulator of the computer code, which is used for all further analysis.

As we have seen in section 3.1, classical quadrature rules perform well on low dimensional integrals. However, the underlying polynomial interpolation does not generalize well to higher dimensions: to obtain the same error bound as the dimension D is doubled, we need to square the number of nodes N. In contrast, the error bound of the Monte Carlo method is independent of D, as the integrand $F[f]$ is not modeled using any interpolation scheme.

For integrals of dimension higher than $D = 4$ Monte Carlo outperforms classical quadrature, apparently because not assuming any structure on $F[f]$ is more effective than doing polynomial interpolation. However, the essential statement of learning theory is that the error bounds of function estimation methods do not necessarily depend on the dimension of the input space, but rather on the complexity of the function. Thus, as long as a learning algorithm restricts the complexity through regularization, it can generalize well despite high dimensional inputs (Vapnik, 1995, chap. 5).

Gaussian process priors are nonparametric models which implement such regularization and therefore show good generalization performance in high dimensional spaces. We can therefore expect that their use extends the favorable

properties of classical quadrature to higher dimensions, as long as the GP can capture the underlying structure of the data.

The GP prior for numerical quadrature has been proposed by O'Hagan (1991) to replace classical Gauss-Hermite rules for one-dimensional and factorial integration. Rasmussen and Ghahramani (2003) verified that Bayesian Monte Carlo can outperform classical MC in high dimensional applications—showing that the advantage of MC over classical quadrature is due to the inadequacy of the underlying polynomial models for high dimensions.

O'Hagan (1987) entitles his arguments against MC drastically with "Monte Carlo is fundamentally unsound". Besides arguments against importance sampling, his main point against MC methods is that only the function values $f(\mathbf{x}_\ell)$ enter the estimate, not the inputs $\mathbf{x}_\ell$ themselves. Not exploiting the information from the inputs stems from the fact that $F[f]$ is interpreted as a random variable, instead of directly modeling the mapping $F[f(\mathbf{x})]$. Thus, once the input distribution is changed, previous function evaluations cannot be re-used. The Bayesian approach has the advantage that the samples do not have to reflect the input distribution. Other than with MC, with BMC we can recycle the data in further analyses or even reduce the number of function evaluations by choosing an optimal design (see chapter IV).

Bayesian Monte Carlo for SA. For sensitivity analysis Bayesian quadrature has been proposed in (Haylock and O'Hagan, 1996) and (Oakley and O'Hagan, 2002). Having computed the posterior process $p(f|\mathcal{D}, \boldsymbol{\theta}^*)$ from the available simulation runs, we can compute the posterior estimate for mean or variance of the output $f(\mathbf{x})$ under $p(\mathbf{x})$ (3.14) from the GP's predictive distribution (II 3.17b).

The solution involves longish expressions, which we have moved to the appendix B. The quadrature problem can eventually be reduced to integrating products of the input distribution $p(\mathbf{x})$ and the covariance function, and all integrals can be expressed through the following quantities:

$$k_c = \int d\mathbf{x}\, p(\mathbf{x}) \int d\mathbf{x}'\, p(\mathbf{x}')\, k(\mathbf{x}, \mathbf{x}') \tag{3.16a}$$

$$k_o = \int d\mathbf{x}\, p(\mathbf{x})\, k(\mathbf{x}, \mathbf{x}) \tag{3.16b}$$

$$z_\ell = \int d\mathbf{x}\, p(\mathbf{x})\, k(\mathbf{x}, \mathbf{x}_\ell) \tag{3.16c}$$

$$L_{ij} = \ell(\mathbf{x}_i, \mathbf{x}_j) = \int d\mathbf{x}\, p(\mathbf{x})\, k(\mathbf{x}, \mathbf{x}_i) k(\mathbf{x}, \mathbf{x}_j)\ . \tag{3.16d}$$

Note that we can also calculate confidence intervals for the estimated quantities by taking into account the remaining uncertainty in the posterior process $p(f|\mathcal{D}, \boldsymbol{\theta}^*)$.

If the common squared exponential covariance function (II 3.20) is used, the integrals (3.16) can be calculated explicitly for uniform and Gaussian input distributions

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\hat{\mathbf{x}}, B)\ . \tag{3.17}$$

If the input distribution factorizes, $p(\mathbf{x}) = \prod_\ell p_\ell(x_\ell)$, the integrals in (3.16) break down to a product of one dimensional integrals. These are, in contrast to a full integral of the type (3.14), relatively easy to handle using e.g. Gauss-Hermite rules. We provide the details in appendix B.

Integrals such as the parametric yield (1.4) cannot be simplified. However, using the fast GP emulator of the original code, we can use a simple Monte Carlo estimate with a large number of samples.

Nonzero mean functions. Up to this point we have derived BMC for a GP with zero mean function, however, the generalization is straightforward. Assume we add an offset $\mu(\mathbf{x})$ to the GP prediction $m(\mathbf{x})$ in (II 3.17). The expectations over $p(\mathbf{x})$ (2.10) decompose for this sum as

$$\begin{aligned} \mathrm{mean}_{\mathbf{x}}\left[m(\mathbf{x}) + \mu(\mathbf{x})\right] &= \mathrm{mean}_{\mathbf{x}}\left[m(\mathbf{x})\right] + \mathrm{mean}_{\mathbf{x}}\left[\mu(\mathbf{x})\right] && (3.18a) \\ \mathrm{var}_{\mathbf{x}}\left[m(\mathbf{x}) + \mu(\mathbf{x})\right] &= \mathrm{var}_{\mathbf{x}}\left[m(\mathbf{x})\right] + \mathrm{var}_{\mathbf{x}}\left[\mu(\mathbf{x})\right] && (3.18b) \\ &\quad + 2\mathrm{covar}_{\mathbf{x}}\left[m(\mathbf{x}) \,,\, \mu(\mathbf{x})\right] . \end{aligned}$$

Thus, as long as we can compute the integrals over the products of $\mu(\mathbf{x})$, $p(\mathbf{x})$ and $k(\mathbf{x}, \cdot)$, it can easily be incorporated into the analysis. For example, for a polynomial offset in combination with the SE covariance function $k_{\mathrm{SE}}(\mathbf{x}, \bar{\mathbf{x}})$ and Gaussian or uniform input distribution $p(\mathbf{x})$ all integrals are analytically tractable.

3.3. Robust design optimization

General problem. Design centering methods automatically optimize the robustness of a design with respect to maximal deviations from the nominal value. Using our approach such automatic optimization can be done on the global GP response surface while explicitly taking into account the distribution of statistical fluctuations on input parameters. The stone pitch example on page 42 illustrates such optimization.

Design optimization is by nature a difficult task: in a manual adjustment the engineer is usually mindful of a number of design restrictions and opposed objectives. Before an automatic optimization can be performed, these need to be cast into a *utility function* (see section 2.2) and explicit constraints.

utility function

Once the problem has been formalized, the optimization can be done automatically. However, when the utility function takes into account the process tolerances, its evaluation can be computationally expensive. The MC method, for example, requires an independent average for each evaluation and is thus inapplicable. In contrast, the BMC approach provides the averages over $p(\mathbf{x})$ in closed form as an efficient way to replace MC runs on the simulation software.

Yield optimization. The yield (1.4) is usually the main contribution to the utility function for design optimization. Approximating the uncertainty distribution using its first two moments (2.10), the yield for a specification of the

type $f(\mathbf{x}) \in [f^{\min}, f^{\max}]$ (1.3) results in

$$\text{Yield} = \int_{\mathcal{F}} \mathrm{d}\mathbf{x}\, p(\mathbf{x}) = \int_{f^{\min}}^{f^{\max}} \mathrm{d}f\ p_{\mathbf{x}}(f) \tag{3.19a}$$

$$\approx \int_{f^{\min}}^{f^{\max}} \mathrm{d}f\ \mathcal{N}(f|\text{mean}_{\mathbf{x}}[f], \text{var}_{\mathbf{x}}[f]) \tag{3.19b}$$

$$= \Phi\left(\frac{f^{\max}-\text{mean}_{\mathbf{x}}[f]}{\sqrt{\text{var}_{\mathbf{x}}[f]}}\right) - \Phi\left(\frac{f^{\min}-\text{mean}_{\mathbf{x}}[f]}{\sqrt{\text{var}_{\mathbf{x}}[f]}}\right) , \tag{3.19c}$$

where Φ denotes the CDF of the normal distribution (3.20)[9].

Both moments, $\text{mean}_{\mathbf{x}}[f]$ and $\text{var}_{\mathbf{x}}[f]$, can be derived in closed form from the GP emulator as functions of the input distributions' parameters—such as nominal values and process tolerances. Evaluating the expressions for the yield and its derivatives with respect to all parameters is extremely fast. An optimum can therefore be computed efficiently using standard gradient-based optimization schemes as described in (Press et al., 1986, Chap.10). In particular for nonlinear outputs f, the uncertainty distribution might not be approximated well by a normal distribution. However, optimizing the yield approximation (3.19) will in those cases still lead to a type of design centering.

Yield optimization is a major feature of the software tool for engineering, which we have developed on the basis of the presented approach.

4. Experiments

In the following section we present the results of our evaluation of Bayesian Monte Carlo for sensitivity analysis. To assess the convergence properties in comparison to the Monte Carlo method we have used several benchmark problems from literature (section 4.1) and three fully featured models of MEMS devices from current development for mass production (section 4.2).

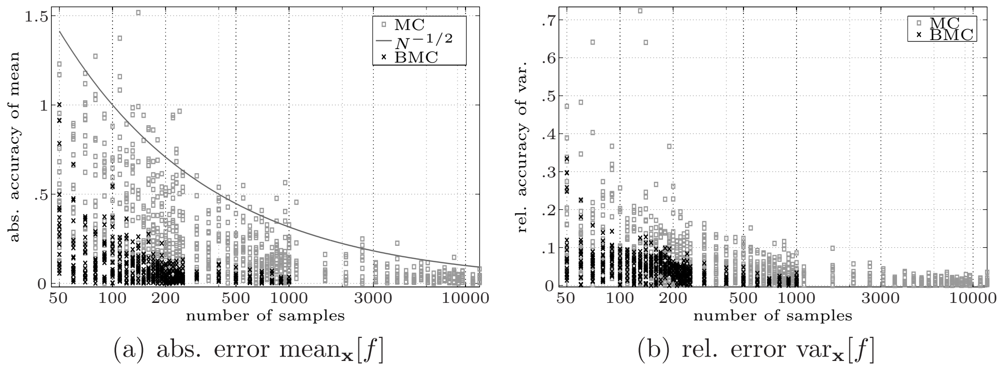
4.1. Analytical benchmark problems

Friedman's function.

Definition. Our first example is a function which was defined by Friedman (1991) as a benchmark problem for regression. It is nonlinear and non-monotonic, and therefore a challenging problem for sensitivity analysis. The function has 10 input parameters, 5 of them having an impact on the output. We use a normal input distribution $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\hat{\mathbf{x}}, B)$ with mean $\hat{x} = (0, 0, \frac{1}{2}, 0 \ldots 0)$ and covariance $B = \text{diag}(\frac{1}{4} \ldots \frac{1}{4})$. As we use a symmeterized version of the original function

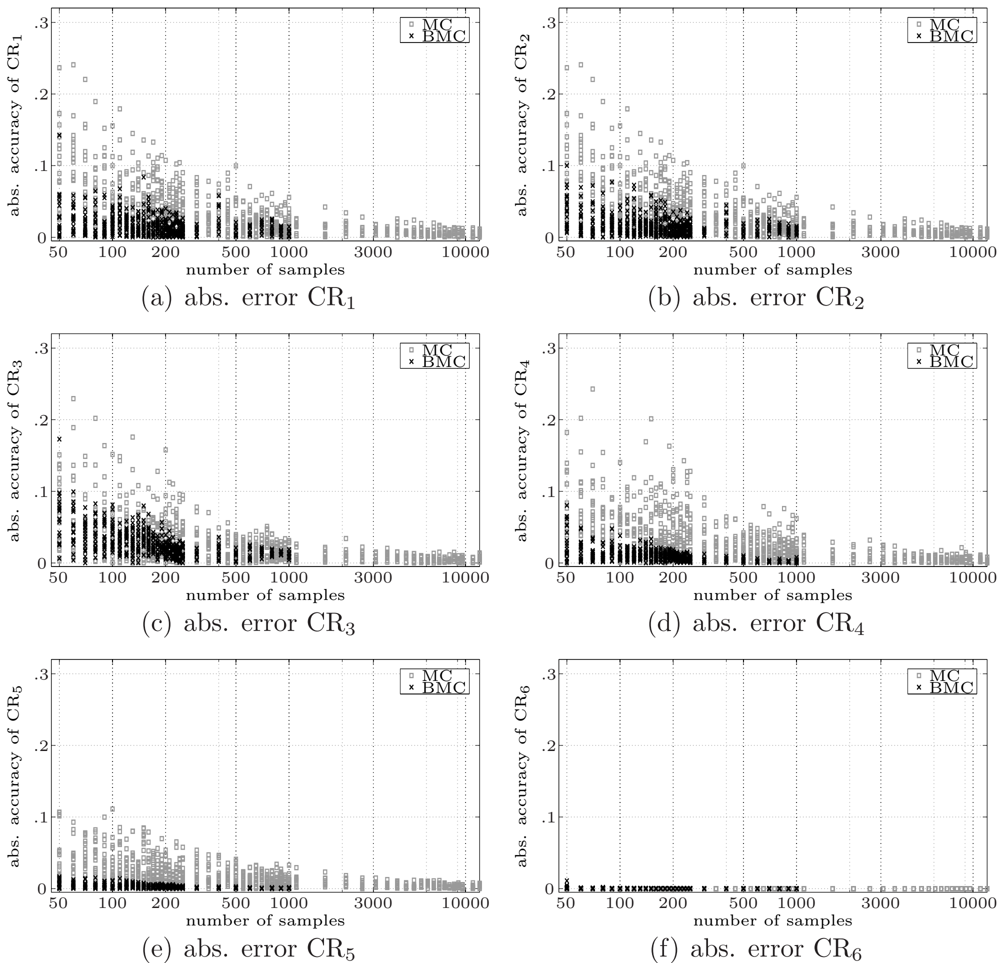
$$f(\mathbf{x}) = 10\sin(\pi x_1 x_2) + 20(x_3 - \tfrac{1}{2})|x_3 - \tfrac{1}{2}| + 10x_4 + 5x_5 , \tag{4.21}$$

[9] The error function is defined as $\Phi(z) = \int_{-\infty}^{z} \mathrm{d}x\, \mathcal{N}(x|0,1)$. (3.20)



(a) abs. error $\text{mean}_{\mathbf{x}}[f]$ (b) rel. error $\text{var}_{\mathbf{x}}[f]$

Figure III.4.: Friedman's benchmark function: Convergence rates for MC and the GP-based BMC scheme. Shown are the errors of the estimates for mean (a) and variance (b) against the number of samples. The true values serve as a reference. The solid curve in (a) indicates the characteristic error bound of the MC estimate for the mean, $\mathcal{O}(N^{-1/2})$.



(a) abs. error CR_1 (b) abs. error CR_2

(c) abs. error CR_3 (d) abs. error CR_4

(e) abs. error CR_5 (f) abs. error CR_6

Figure III.5.: Friedman's benchmark function: Convergence rates for the CRs using MC and BMC. Note that x_6 does not enter $f(\mathbf{x})$, making the screening capability of BMC speed up the convergence of CR_6 drastically.

we ensure that the mean under $p(\mathbf{x})$ is zero. All sensitivity measures can be computed analytically and we compare the estimates to the true values. In this example we do not add noise, as this corresponds to the common situation in computer experiments. Note, however, that the BMC procedure handles noise automatically.

The plots in figures III.4 and III.5 depict the convergence rates of MC and BMC. Each marker represents one of 1 900 experiments using Latin Hypercube designs, which are evaluated using MC and BMC.

Panel III.4(a) shows the convergence rates for the estimate of $\text{mean}_{\mathbf{x}}[f]$, where the MC approximation is governed by the typical convergence rate $\mathcal{O}(N^{-1/2})$. BMC achieves the same accuracy as MC on 10 000 runs, using only 10% of the sample size. The same holds for the variance estimate shown in III.4(b).

The correlation coefficients cannot be computed in only one MC run, as the input distribution is changed for each CR—while no extra simulation run is needed for BMC. In figure III.5 we plot the estimates for the first 6 CRs against the number of simulation runs[10]. One observes that BMC is again an order of magnitude more efficient than MC. The output is independent of $x_6 \dots x_{10}$, and correspondingly $\text{CR}_6 \dots \text{CR}_{10} = 0$. In these cases BMC profits from the GP's screening capability (ARD, eq. (3.18)) and converges on less than 100 samples to the correct value (see panel III.4(f) for CR_6). Nevertheless, also MC detects zero influence on few simulation runs.

SA benchmark problems.

Description. A variety of benchmark problems for SA has been defined by a number of authors. We consider 7 problems in our comparison of BMC and MC, which have been collected by Saltelli et al. (2000a, Chap. 2). A brief description of all problems is given in appendix B. The dimensionality of the problems ranges from 2 to 20 active parameters. Note that model 4 is not differentiable and thus contradicts the smoothness assumptions of BMC.

The results of our experiments are given in table III.1. We have used Latin Hypercube designs of various sizes to obtain an accuracy between 1% and $\frac{1}{10}$% with BMC, and compare it to the MC estimates on the same designs.

The accuracy of the BMC estimates depends strongly on the complexity of the underlying function: While low dimensional mappings can be captured well using only a few hundred samples (model 1, 3, 5), functions with many parameters require more function evaluations. An extreme example is model 2(b), which is defined by 20 equally important inputs. While the screening ability lets the GP ignore inactive parameters—as in Friedman's function—here good accuracy is only obtained using 3 000 samples.

[10]Saltelli (2002) describes a scheme to avoid using a full extra MC run for each CR. Therefore we count only the evaluations of one MC run to keep the comparison fair. However, for simplicity we use extra simulations, setting the corresponding parameters to their nominal values.

BMC clearly outperforms MC in all cases. However, as the number of active parameters increases the difference becomes smaller: on model 1 the accuracy of BMC on $\mathrm{var}_{\mathbf{x}}[f]$ is by a factor 1430 better than MC, while the factor is 2.3 for model 4.

4.2. Case studies from industrial engineering

Pressure sensor.

We have introduced the PS model in section 2.2, page 49, where we show the sensitivity analysis in figure III.3. It turns out that the model is highly non-linear and non-additive, however, only 5 out of 28 parameters have significant impact on the output.

Figure III.6 shows 130 runs on Latin Hypercube designs with 50 to 2 000 samples, where we have computed mean (a) and variance (b) of the output distribution using MC and BMC. As the computation of the CRs would have required extra simulation runs, we only show the BMC results in panel (c) and (d). Note that BMC is only slightly more accurate than MC on the estimate for $\mathrm{mean}_{\mathbf{x}}[f]$. However, BMC estimates the central quantity for SA, $\mathrm{var}_{\mathbf{x}}[f]$, on 200 samples as well as MC on 2 000 and discovers that the minimal CR is zero on only 120 samples. As the CRs are computed analytically from the global fit, the maximum CR converges as fast as the variance.

Accelerometer.

accelerometer

AC model: Our second simulation code models the behavior of a micro electro-mechanical accelerometer which is used e.g. to trigger airbags.

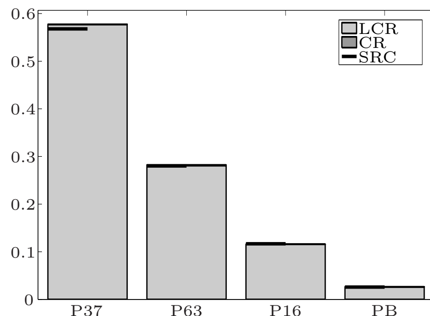


Figure III.7.: Sensitivity analysis for the accelerometer.

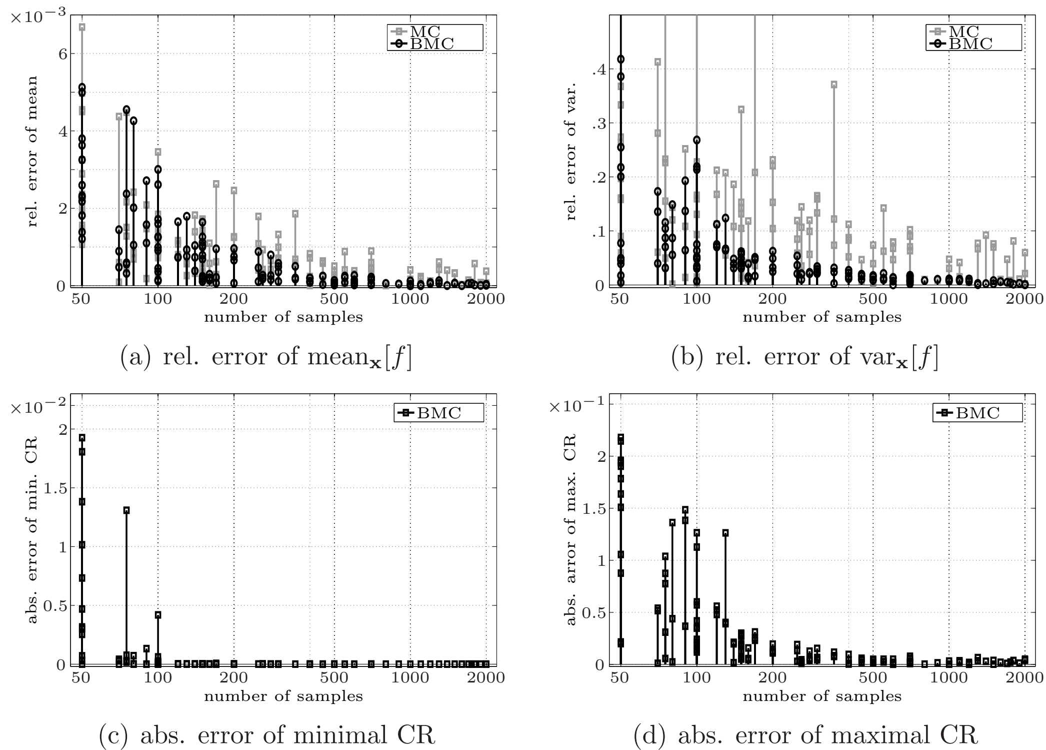
The model has 29 parameters which show variations in the manufacturing process. The predictions of the GP model, trained on 300 points, lead to a root mean squared error of 3.7% relative to the standard deviation on an independent test set of 4 700 instances.

It turns out that this model is dominated by linear effects, as indicated by the sensitivity analysis shown in figure III.7. We find that only 4 parameter have significant influence on the output. Nonlinearities or cross terms can be neglected on the region given by $p(\mathbf{x})$.

The convergence rates for the AC model are shown in figure III.8, where the designs are made of independent, identically distributed samples from $p(\mathbf{x})$. Observe that the MC method's estimate of $\mathrm{var}_{\mathbf{x}}[f]$ converges much slower than that of BMC: On 75 training instances BMC is as accurate as MC on 500 samples. Both methods perform comparably in estimating the $\mathrm{mean}_{\mathbf{x}}[f]$, which can be explained by the great linearity of the model: Due to linearity,

Model	D	N	**MC**	**BMC**	**ratio**
			error $\text{mean}_{\mathbf{x}}[f]$ relative to $\text{std}_{\mathbf{x}}[f]$ in %		means
1	2	100	0.08 [0.0009, 0.24]	0.0020 [0.00009, 0.0065]	**0.0250**
2(a)	6	600	1.66 [0.0213, 4.49]	0.0388 [0.00251, 0.0775]	**0.0233**
2(b)	20	3000	0.72 [0.0501, 1.72]	0.0676 [0.00444, 0.2140]	**0.0938**
3(a)	2	30	1.61 [0.3635, 4.22]	0.0364 [0.00125, 0.1848]	**0.0226**
3(b)	2	300	2.59 [0.2264, 6.80]	0.0124 [0.00194, 0.0410]	**0.0048**
4	8	2000	0.68 [0.0253, 1.57]	0.1515 [0.00651, 0.43864]	**0.2223**
5	3	500	1.98 [0.0651, 6.02]	0.0570 [0.00388, 0.1740]	**0.0287**
			relative error $\text{var}_{\mathbf{x}}[f]$ in %		
1	2	100	8.32 [0.4198, 22.9]	0.0062 [0.00020, 0.0279]	**0.0007**
2(a)	6	600	10.8 [0.2033, 24.1]	0.3976 [0.04713, 0.8913]	**0.0365**
2(b)	20	3000	4.76 [0.1118, 16.5]	1.8373 [0.06968, 3.4314]	**0.3863**
3(a)	2	30	14.7 [1.1521, 42.3]	0.1258 [0.00441, 0.6785]	**0.0086**
3(b)	2	300	13.9 [0.7684, 31.9]	0.1170 [0.00203, 0.3941]	**0.0084**
4	8	2000	2.19 [0.4055, 7.29]	0.9554 [0.38691, 1.8166]	**0.4346**
5	3	500	3.77 [0.0599, 11.6]	0.2684 [0.00505, 0.5694]	**0.0711**

Table III.1.: Convergence on SA benchmark problems (app. B). The figures represent the performance on 20 independent designs (mean [best, worst]). The last column ("ratio") contains the ratio of mean BMC- and MC-error.



(a) rel. error of $\text{mean}_{\mathbf{x}}[f]$

(b) rel. error of $\text{var}_{\mathbf{x}}[f]$

(c) abs. error of minimal CR

(d) abs. error of maximal CR

Figure III.6.: PS model: pressure sensor. Convergence rate of MC and BMC on mean, variance, minimal and maximal CR. As reference we have chosen the mean over all estimates using 2 000 samples.

all effects caused by a deviation from the nominal value cancel in the mean estimate and effectively we only need to estimate the offset $f(\hat{\mathbf{x}})$—where the MC method is as efficient as the Bayesian approach. When we turn to the variance and the CRs, BMC can again profit from prior assumptions and its screening capability, showing extremely good estimates on only 75 samples (observe that the plots have different scales on the y-axis).

Yaw rate sensor.

yaw rate sensor

YR model: A REM picture of the yaw rate sensor is shown in figure I.1, page 16.

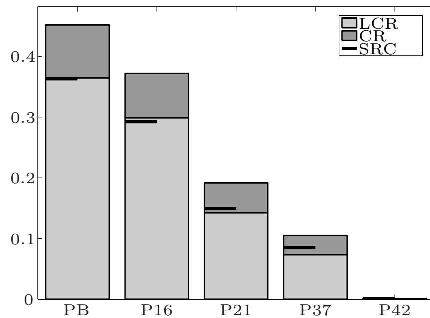


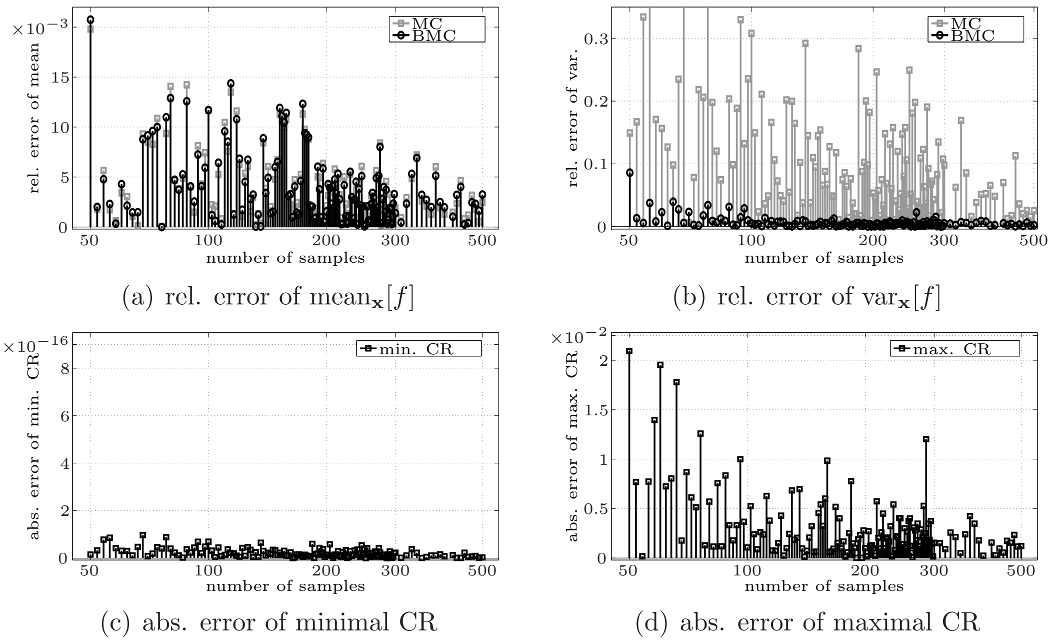
Figure III.10.: Sensitivity analysis for the yaw rate sensor.

The sensor measures the yaw rate which results when the device is moved along a bent curve, and is used in applications like the *Electronic Stability Program*. The model of the yaw rate sensor has 15 fluctuating inputs. We consider the output which corresponds to the responsivity of the device. Figure III.10 shows the results of the sensitivity analysis, which were obtained using BMC on 500 random samples from $p(\mathbf{x})$. The root mean squared error on an independent test set of 4 500 samples was 1.1% relative to the standart deviation. Only 5 parameters contribute more than 1% to the variance of the output. As the SRCs, which are based on a linear fit, give similar results as the LCRs, we can conclude that no strong nonlinear effects can be found along the axes. However, a pronounced difference between LCRs and SRCs indicates that the model is highly non-additive.

Electronic Stability Program

The accuracy of the SA is shown in figure III.9 for varying design sizes. The inputs have been drawn independently from $p(\mathbf{x})$ and the designs were evaluated using MC and BMC.

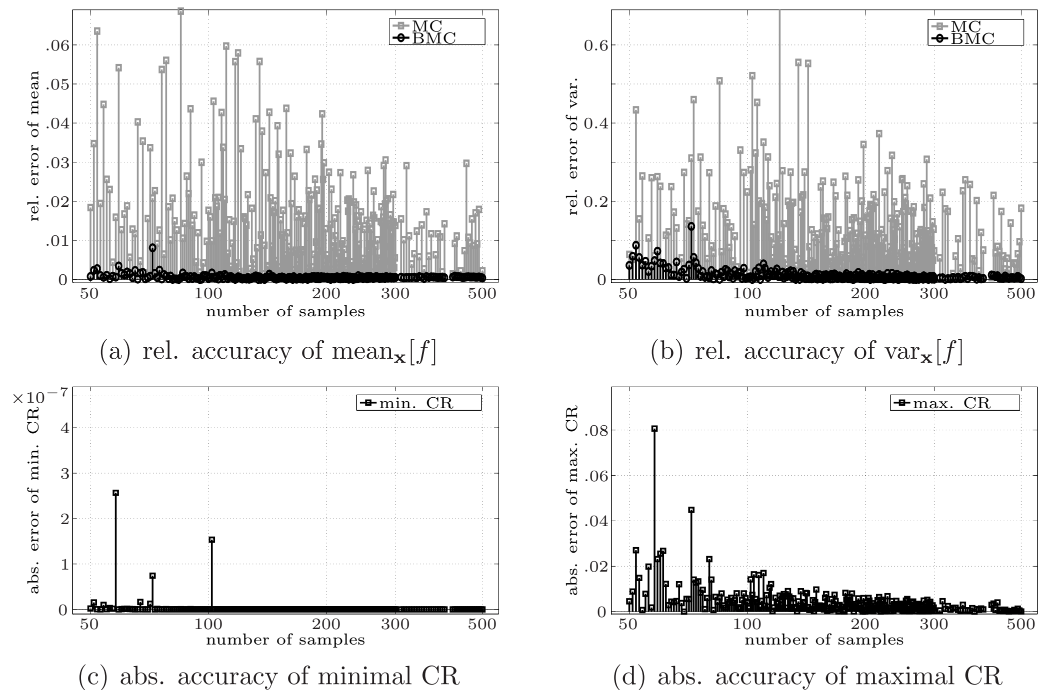
The comparison shows that BMC clearly outperforms MC on both, mean and variance estimate. For the variance the BMC estimate is accurate within 2% on 500 samples, while the deviations of MC are as large as 18%. BMC obtains the same accuracy on the CRs. In particular zero coefficients are found efficiently on few samples due to the screening ability of BMC.



(a) rel. error of $\text{mean}_{\mathbf{x}}[f]$ (b) rel. error of $\text{var}_{\mathbf{x}}[f]$

(c) abs. error of minimal CR (d) abs. error of maximal CR

Figure III.8.: AC model: accelerometer. Convergence for $\text{mean}_{\mathbf{x}}[f]$, $\text{var}_{\mathbf{x}}[f]$ and minimum/maximum CR. The reference is the mean estimate of all approximations on more than 200 samples. The accuracy of the normalized CRs is given on an absolute scale.



(a) rel. accuracy of $\text{mean}_{\mathbf{x}}[f]$ (b) rel. accuracy of $\text{var}_{\mathbf{x}}[f]$

(c) abs. accuracy of minimal CR (d) abs. accuracy of maximal CR

Figure III.9.: YR model: yaw rate sensor. Convergence for $\text{mean}_{\mathbf{x}}[f]$, $\text{var}_{\mathbf{x}}[f]$ and minimum/maximum CR. The reference is the mean estimate of all approximations on more than 200 samples.

5. Discussion

In this chapter we have described a novel approach to design analysis, which has now been implemented for regular use in the designing process of MEMS at Robert Bosch GmbH. We have defined a number of statistically justified sensitivity measures to express the structure of the model: the degree of linearity and additivity, and the impact of fluctuating parameters in mass production. In a real-world case study we have exemplified the interpretation of the results.

The bottleneck of sensitivity analysis is often the time-consuming simulation software. To use simulation runs efficiently—and to re-use them for multiple analyses—we have used Bayesian Monte Carlo. Furthermore, BMC makes it possible to compute the SA analytically on a global response surface, while validating the accuracy through standard methods like cross validation.

Using BMC, we can compute an analytical approximation of the yield, including expressions for its gradient with respect to the parameters of the input distribution. The design can thus be optimized using standard methods like gradient ascent, correctly reflecting the distribution of the fluctuations. This method is conceptually different from previous approaches, which require repeated MC runs or use geometrical design centering approximations.

To evaluate the convergence properties of BMC we have compared it to MC on a variety of problems: Three problems were simulations of MEMS sensors in development at Robert Bosch GmbH, where we have assessed the accuracy over a large range of sample sizes. The models have up to 29 parameters, where the SA showed that between 4 and 7 inputs have significant influence on the output. Due to their screening ability, GPs can efficiently detect inactive parameters and BMC converges significantly faster than MC.

We considered a number of nonlinear benchmark problems from literature with 2 to 20 active parameters. A comparison of MC and BMC showed that on all problems—and especially on simple functions—BMC significantly improves accuracy on a given set of simulation runs. As the number of active parameters grows, the difference between both methods becomes smaller. The situation is similar to classical quadrature, where MC catches up when linear interpolation fails to extract the structure from the data. In this sense, BMC can be seen as an extension of classical quadrature to functions of moderate complexity in high dimensional spaces.

We believe that BMC is particularly well suited for models of devices which are considered for mass production: Typically, the models have a large number of parameters, while the output varies moderately within process tolerances.

The BMC approach meets a number of requirements which suggests its use in industrial engineering. Using the GP emulator, we separate simulation calls from the analysis. Those can therefore be computed as a batch, and the designer can explore the model interactively, avoiding waiting times. The simulation runs can be used for several analyses, as the inputs do not have to reflect the input distribution. Another benefit of BMC is that it can be used in connection with active learning, saving simulation runs by using optimal experimental designs—details can be found in the following chapter IV.

IV. Active Learning for nonparametric regression

In the previous chapter we have described Bayesian Monte Carlo (BMC) for sensitivity analysis. The BMC approach is based on nonparametric Gaussian process (GP) regression, and in comparison to Monte Carlo it can drastically reduce the number of simulation runs required for a given accuracy. Another advantage of the BMC approach is that simulation runs do not have to reflect the distribution of the process fluctuations: This allows for another degree of freedom to increase the efficiency by using an experimental design.

In this chapter we present an active learning scheme that uses available data to run the simulation software at expectedly informative configurations. Our learning scheme is based on the Bayesian expected utility, which measures the expected generalization error on samples from a given input distribution. We show that one can derive the expected utility exactly, and as we avoid expensive numerical approximations, the resulting learning scheme is computationally efficient and easy-to-use. The approach can thus be used by non-experts in combination with the tool which we have developed for the evaluation of computer experiments in sensitivity analysis (see chapter III).

Our experiments show that the proposed scheme can significantly reduce the number of simulation runs required to obtain a desired accuracy. The method is very robust, leading to good designs even in presence of noise and for underlying functions which are hard to learn.

We have described our approach to active learning previously in (Pfingsten, 2006), which is to our knowledge the first work to present the expected loss which corresponds to the generalization error in closed form.

This chapter is grouped into four sections. Section 1 addresses previous work on Bayesian experimental design, and active learning as it is known to the machine learning community. We discuss their relation, asymptotical learning rates, and fundamental difficulties. Section 2 contains the derivation of our active learning scheme from the corresponding Bayesian expected utility. We discuss the objectives which form the basis of A- and D-optimal designs, the effect of the ML-II approximation on optimal designs and nonstationary models. We have tested our approach in a number of experiments using analytical benchmark problems and fully featured sensor models from the development at Robert Bosch GmbH. The results are presented in section 3, and we close with a discussion in section 4.

1. Experimental design and active learning

Before we describe our approach to active learning for Gaussian process regression, we introduce the basic concepts in this section. Bayesian experimental design has a long history, which we outline in section 1.1 in relation to active learning as used by the machine learning community. We show in section 1.2 that there is no fundamental difference between the two approaches, even though the resulting algorithms and their derivations often seem very distinct: both approaches are ultimately approximate solutions of Bayesian decision theory, which correspond to distinct foci. All optimal designs, especially when approximations to the full Bayesian solution are used, need to be used with care. We discuss this fundamental problem in section 1.3. Besides empirical studies there exists much work on asymptotical learning rates of active learning schemes, which we review in section 1.4.

1.1. Historical development

Bayesian experimental design. Experimental design is an instance of decision theory (II 2.2), where the problem is to determine an optimal design matrix $\mathbf{X}$ with respect to some utility function

$$\mathrm{U}(\mathbf{X}|\mathcal{M}, \mathcal{D}_o) \ . \tag{1.1}$$

$\mathcal{D}_o$ denotes prior knowledge and $\mathcal{M}$ the chosen model[1]. The utility function can in principle be based on any objective, possibly specific to a single problem. More general utilities, based on the entropy as a measure for information have been introduced by Lindley (1956, 1968).

While early work focuses on parametric models, experimental design has been proposed for nonparametric Gaussian process regression by Sacks and Ylvisaker (1966, 1968) and O'Hagan (1978).

Comprehensive reviews on Bayesian experimental design are given by Fedorov (1972) and Chaloner and Verdinelli (1995). In this work we focus on the most prominent objectives, which lead to *D-Optimal* and *A-Optimal* designs.

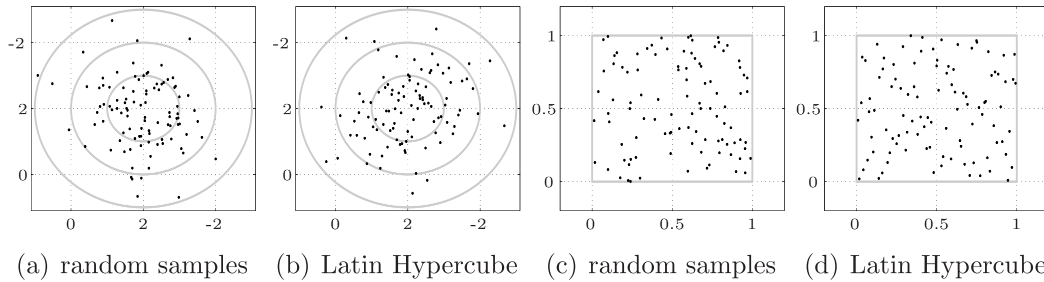
D-Optimal
A-Optimal

Space filling and factorial designs. While Bayesian experimental designs are based on a utility function which encodes the objective of the experiment, a number of heuristics have been proposed to improve independently identically-distributed sampled designs.

space filling designs

For nonparametric regression these are commonly *space filling designs*, which minimize the discrepancy, i.e. avoid uncovered areas in the input region. The most common member of space filling designs is Latin Hypercube, proposed by MacKay et al. (1979). We give a short description in the appendix B. Figure IV.1 illustrates Latin Hypercube and random sampling in two dimensions. Latin Hypercube stratifies samples in one dimensional projections, a refined version for more dimensional projections is given by Ye (1998). Johnson et al.

[1]For simplicity we omit the condition on the model $\mathcal{M}$ in the following. However, the necessary condition on the model is important—we discuss its implications in section 1.3.



(a) random samples (b) Latin Hypercube (c) random samples (d) Latin Hypercube

Figure IV.1.: Random samples and Latin Hypercube designs. The plots show 100 samples for $\mathcal{N}(0,1)^2$ (a, b) and $\mathcal{U}(0,1)$ (c, d). Note that independent samples tend to leave large areas under $p(\mathbf{x})$ uncovered. Latin Hypercube stratifies sampling in one dimensional projections, yet the samples still seem "clustered" in two dimensions and the designs can hardly be distinguished from independent samples.

(1990) define the *MiniMax* and *MaxiMin* criteria, where the latter can be interpreted a limiting case of D-optimal designs for GPs with small length scale parameters.

MiniMax
MaxiMin

For experimental plans with a large number of parameters *factorial designs* are widely used. They are based on linear or quadratic fits and define sparse schemes to determine the model's parameters (Myers and Montgomery, 2002; Taguchi et al., 2004).

factorial designs

Active learning in machine learning Experimental design, which has its origins in Bayesian statistics, has been picked up by the machine learning community under the synonym *active learning* for various applications. MacKay (1992b) reviews statistical approaches with regard to their use for regression with neural networks. Cohn et al. (1996) formulate the statistical approach from the machine learning perspective, using the fundamental decomposition of the generalization error into a variance and a bias contribution. The authors propose to minimize the variance term, which is tightly connected to A-optimal design. In his work from 1997 Cohn derives a new approach which focuses on the bias term, i.e. systematic deviations.

There is a large number of publications on the application of active learning to regression. This includes approaches with neural nets (Plutowski and White, 1993; Cohn, 1994) and Gaussian processes (Seo et al., 2000). Yu et al. (2006) interpret active learning in the setup of transductive learning.

Active learning is by far more popular for classification than for regression, where a whole branch of methods has been developed. Especially in classification the focus has moved away from the Bayesian viewpoint of expected utility to the definition of (often heuristic) sampling schemes. An exception is (Chapelle, 2005), which resembles A-optimality. Bryan et al. (2006) describe active learning to identify thresholds in a continuous output, a problem which is closely related to classification.

In *relevance sampling* experts are only queried for samples which a learning algorithm clearly assigns to a class (Salton and Buckley, 1990). The aim

relevance sampling

is to save human effort on samples which are unlikely to contain any clear structure. *Uncertainty sampling* is designed as the counterpart to relevance sampling, querying the most uncertain inputs (Lewis and Gale, 1994; Roy and McCallum, 2001). A problem common to many machine learning algorithms is that they tend to lack a notion of uncertainty. For support vector machines the uncertainty can be substituted by the distance to the separating hyperplane (Campbell et al., 2000), and Mitra et al. (2004) use the probabilistic interpretation proposed by Platt (1999).

uncertainty sampling

The volume of the *version space* serves as another criterion for the potential information of a new observation. It is used by the popular *Query by Committee Machine* (Seung et al., 1992; Gilad-Bachrach et al., 2006), which basically chooses queries which rule out a maximal number of hypotheses.

version space

Query by Committee Machine

1.2. From experimental design to active learning

Experimental design is used to determine complete optimal designs of N samples before any experiments are performed. A main issue has been to find approximate designs for large N, as the exact problem is NP-hard (Ko et al., 1995).

In the machine learning community the term "active learning" has replaced the statistical expression "experimental design", as the focus has moved from planning a whole batch of experiments to actively planning the experiments one after the other. The advantage of this procedure is that the learning algorithm can be updated after query, hence considering all available information for planning remaining experiments. Although the approaches are quite different in their goal, in the Bayesian setting both are optimally solved by maximizing the expected utility (chapter II (2.2)):

Consider classical experimental design, where the queries $\mathbf{X}$ are planned as a batch, maximizing the expected utility $\mathrm{U}(\mathbf{X}|\mathcal{D}_o)$:

$$\mathcal{D}_o \xrightarrow[\mathbf{x}_1, \ldots \mathbf{x}_N]{\text{design}} \mathcal{D}_N$$

If we assume that the outcomes of all experiments become available at once, the solution is optimal. However, if the results come one-by-one, the remaining experimental schedule should be refined in each step ℓ by considering the measured $y_1, y_2 \ldots y_\ell$ in the prior belief $\mathcal{D}_\ell$ at that time. In the Bayesian formalism it is clear that this information is correctly considered by maximizing $\mathrm{U}(\mathbf{x}_{\ell+1} \ldots \mathbf{x}_N|D_\ell)$, and the active learning scheme becomes:

$$\mathcal{D}_o \xrightarrow[\mathbf{x}_1, \ldots \mathbf{x}_N]{\text{design}} \mathcal{D}_1 \xrightarrow[\mathbf{x}_2, \ldots \mathbf{x}_N]{\text{design}} \mathcal{D}_2 \ \cdots \ \xrightarrow[\mathbf{x}_{N-1}, \ldots \mathbf{x}_N]{\text{design}} \mathcal{D}_N$$

The difference between experimental design and active learning is thus merely whether the queries are processed as a batch or in a sequence. However, most active learning schemes avoid the computational burden of planning all remaining experiments by greedily planning only one step ahead. The *greedy*

active learning scheme is given by

greedy active learning

$$\mathcal{D}_o \xrightarrow[\mathbf{x}_1]{\text{design}} \mathcal{D}_1 \xrightarrow[\mathbf{x}_2]{\text{design}} \mathcal{D}_2 \ \ldots \ \xrightarrow[\mathbf{x}_N]{\text{design}} \mathcal{D}_N$$

where the expected utilities $\mathrm{U}(\mathbf{x}_{\ell+1}|\mathcal{D}_\ell)$ are optimized.

1.3. The fundamental drawback of active learning

The Bayesian framework provides a principled and conclusive formalism for active learning. However, it is hard to conceive the implications of the basic assumptions on the resulting designs. An important aspect is the utility function, which might lead to unexpected designs. We discuss some utilities and the corresponding designs in section 2, and focus here on a very basic problem which already becomes apparent in the very first equation of this section: The expected utility (1.1) is conditioned on $\mathcal{M}$, the underlying generative model of the data.

Conditioning on the model is inevitable in Bayesian analysis, yet implies that one is completely sure that the model is correct. MacKay (1992b) calls this the *Achilles' heel* of active learning, as it is typically impossible to verify the model on the basis of optimally planned experiments.

As the linear model implies very strong assumptions about the structure of the data, optimal designs will typically be extreme, only querying inputs at the limits of the input region. For the linear model this behavior is intuitively clear, as the slopes can best be estimated using far apart points. However, an experimenter will typically place some measurements non-optimally to be able to verify the linearity of the function.

The imperfect belief in the model $\mathcal{M}$ can be encoded by using a composite model $\tilde{\mathcal{M}}$ which accounts for the possibility that the data might be generated by an alternative model $\mathcal{M}_{\text{alt}}$. In the spirit of Jaynes's "two-model model" (Jaynes, 2003, ch. 21) the composite model contains a new hyperparameter $q \in [0, 1]$,

$$p(y|\mathbf{x}, \tilde{\mathcal{M}}) \quad = \quad q\, p(y|\mathbf{x}, \mathcal{M}) \ + \ (1-q)\, p(y|\mathbf{x}, \mathcal{M}_{\text{alt}}) \ , \tag{1.2}$$

where a prior on q encodes the confidence in $\mathcal{M}$[2]. The Achilles' heel is in this view simply the design's plausible sensitivity to overconfident models, where the alternative model is neglected.

A natural extension of the linear model could be one which includes higher order terms. Sacks and Ylvisaker (1966) propose GPs to account for correlated errors in the design, and O'Hagan (1978) proposes localized linear models using GPs. However, even flexible nonparametric models such as GPs impose assumptions such as smoothness on the underlying function, and overconfident designs can hardly be ruled out. An illustrative example is given in section 2.2, where we consider the confident ML-II approximation for GPs.

[2] Note that inference in this class of models can be extremely difficult. A comparison of MCMC and expectation propagation to solve a "two-model model" approach to robust GP regression is given by Kuss, Pfingsten, Csató, and Rasmussen (2005).

1.4. Bounds for learning rates

Experimental design or active learning are applied in particular when samples are rare. However, even though they apply to the opposite limit, asymptotic bounds on learning rates can provide valuable insight into their potential benefit for different applications.

Castro et al. (2006) compare the rates of active and passive learning for regression under noise, showing that on stationary problems active learning does not lead to better asymptotic performance than passive learning: Castro et al. consider Hölder smooth functions, proving that arbitrary active and passive learning schemes lead to the same asymptotic learning rates. Intuitively active learning is particularly useful for setups where the experimenter is only interested in small subsets of the input space, such as the separating hyperplane in classification. Castro et al. show that—despite the discouraging results for stationary problems—active learning can lead to a significant improvement when the function's complexity is concentrated in a low-dimensional subset of the input space. The results correspond to the prevalence of work on active learning for classification in comparison to regression, where the function needs to be explored over the whole input region.

Another interesting view in (Castro et al., 2006) is that experimental design does not belong to the class of learning schemes which have the potential to improve the learning rate in nonstationary problems: Intermediate measurements need to be taken into account to locate regions of high relevance, which the learning scheme is to concentrate on.

Seung et al. (1992) prove an exponential learning rate for classification when no noise is present, using the Query by Committee machine to bisect the version space with each query. Similar results are shown by Baum (1991) for learning neural networks on noiseless samples. Nevertheless, it turns out that the results strongly depend on the premises: Dasgupta (2006) gives examples where active learning does not lead to any improvement even in the noiseless case.

2. Active learning for GP regression

The aim of our approach to active learning is to find a suitable optimal design for GPs in the spirit of Bayesian experimental design, which can be computed efficiently and automatically in an easy-to-use tool for industrial engineering.

We introduce the most common information-based utility functions in section 2.1. We show that the contribution of a query to the expected utilities can be computed in closed form for GPs with squared exponential covariance function, except for the average over hyperparameters. We analyze the ML-II approximation for active learning in 2.2, relating it to the full MCMC solution. As indicated by asymptotic learning rates (section 1.4), active learning is most attractive for nonstationary functions. We briefly discuss nonstationary designs in 2.3. Section 2.4 subsumes the algorithm which we ultimately apply for our experiments, discussing all used approximations briefly.

2.1. Information-based objectives

In the following section we derive and illustrate D-optimal and A-optimal designs for regression, which have long been used in Bayesian experimental design. The utility for D-optimal designs is easier to handle than the A-optimal criterion. However, we show that D-optimal designs may lead to undesirable results and that—for special cases—also the utility for A-optimal designs can be computed efficiently for Gaussian processes.

D-optimal design: maximum information gain. Lindley (1956) introduced the *information gain*[3] as an objective for experimental design for parametric models with parameters $\boldsymbol{\alpha}$. The objective corresponds to the situation where experiments are performed to determine the parameters of the model, e.g. the determination of physical constants. As the prior *entropy*[3] $\mathrm{H}[\boldsymbol{\alpha}]$ can be considered fixed, this is equivalent to minimizing the entropy $\mathrm{H}[\boldsymbol{\alpha}|\mathcal{D}]$ of the parameters' posterior distribution[4]. The name D-optimal stems from the normal linear model, where this objective leads to a minimization of the determinant of the paremeters' posterior covariance matrix—the "D" stands for "D"eterminant.

information gain

entropy

Currin et al. (1991) derive a similar measure for Gaussian processes, considering a finite pool of possible samples $\mathbf{x}$: They minimize the joint posterior entropy of the targets which have not been queried. As for the linear model, this related objective leads to a minimization of the determinant of the posterior's covariance matrix, or equivalently to a maximal $|Q| = |K + \mathrm{diag}[\sigma^2, \ldots, \sigma^2]|$ (Q is defined in (II 3.16)). It turns out that adding a single input to the design matrix $\mathbf{X}$, $|Q|$ is maximized by choosing the input with maximal predictive variance in the pool[5].

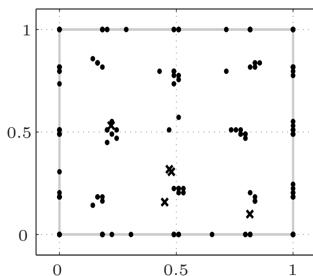


Figure IV.2.: D-optimal design. 5 initial (×) and 195 chosen samples (•).

D-optimal designs are apparently based on a sensible objective function, and it seems intuitively sensible to define a greedy learning scheme which chooses the input with highest predictive variance. However, as argued by MacKay (1992b), D-optimality has undesirable properties in nonparametric regression: The posterior entropy might not be minimized by choosing informative inputs, but rather inputs which lead to simple posterior hypotheses. We show a greedy 2 dimensional D-optimal design in figure IV.2[6] using intermediate length scales

[3] In this chapter we assume that the terms information gain and entropy are known. For an introduction see (Cover and Thomas, 1991). We use the entropy in a different context in ch. V 2.3, page 90, where we introduce its basic properties.

[4] MacKay (1992b) also discusses the Kullback-Leibler divergence between prior and posterior, showing that both corresponding expected utilities are equivalent (II 2.14).

[5] This can easily be shown using a rank-one update of Q^{-1} and the predictive variance in (II 3.17).

[6] We use a GP with squared exponential covariance function (II 3.20) and small noise level ($\sigma = 10^{-2}, v = 1$). The input distribution (gray rectangle) is $\mathcal{U}(0,1)^2$, the pool is a

$w_1, w_2 = 1$. The design tends to choose points at extremes and repeatedly around 9 inner locations, without exploring the region of interest: The designs minimizes the posterior entropy not by choosing informative queries, but instead by choosing them to make a simple hypothesis fit the data.

A-optimal design: minimum generalization error. When the input distribution $p(\mathbf{x})$ is known, the aim of active learning is usually to obtain good generalization to new samples from $p(\mathbf{x})$. This holds in particular for sensitivity analysis, where we are interested in an accurate estimate of averages (III 3.14) over this distribution.

To measure the generalization error of the model we use its predictive variance, averaged over $p(\mathbf{x})$[7]. Integrating out the unseen targets $\mathbf{y}$, we obtain

$$\mathrm{U}_A(\mathbf{X}, \boldsymbol{\theta}|\mathcal{D}_o) = \underbrace{\int \mathrm{d}\mathbf{y}\; p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \mathcal{D}_o)}_{\substack{\text{average over unseen} \\ \text{training targets}}} \underbrace{\int \mathrm{d}\mathbf{x}\; p(\mathbf{x})}_{\substack{\text{average over} \\ \text{region of interest}}} \underbrace{\Big[- \mathrm{var}\left[y|\mathbf{x}, \boldsymbol{\theta}, \mathcal{D}, \mathcal{D}_o\right] \Big]}_{\substack{\text{objective: (negative)} \\ \text{pred. uncertainty}}} . \tag{2.3}$$

For most models the average over $\mathbf{x}$ in (2.3) cannot be calculated in closed form and needs to be approximated numerically. It is therefore common to replace the integral by an MC-like approximation, using a sum over a pool of samples from $p(\mathbf{x})$. When we are given a large pool of data instead of an input distribution, or when we are actually only interested in the given samples, the *pool approach* can be seen as the natural measure rather than an approximation to (2.3). In this case Yu et al. (2006) speak of *transductive experimental design*. Another variant is to estimate the distribution from available samples, e.g. using a Gaussian mixture model.

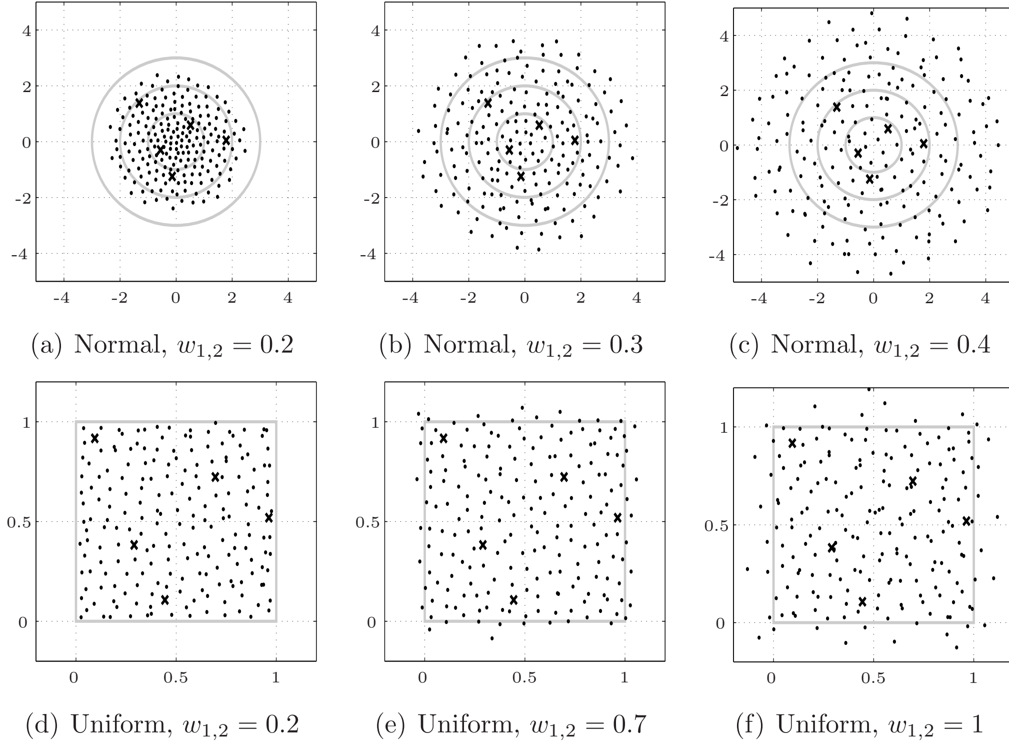
pool approach

The term *A-optimal* stems from the pool approach for the linear model, whose expected utility can be written as the trace of a matrix, which is commonly denoted by “A”.

Exact A-optimal design for GP regression. When dealing with computer experiments with a known input distribution, the pool approach is not justified by itself and it might require a large pool to obtain a good approximation of the integral (2.3). However, for GP regression the utility of an additional training sample can be evaluated in closed form for multivariate normal and uniform input distributions. For other factorizing input distributions the averages can be reduced to one dimensional integrals, and efficient numerical methods for quadrature can thus be applied. The input distribution is usually normal in sensitivity analysis, and for general regression setups it is mostly chosen uniform—hence the exact solution is possible for the most common problem setups. To our knowledge the use of the exact expression for the A-optimal

50×50 grid on $[0, 1]^2$.

[7]MacKay (1992b) discusses several related utility functions to measure the generalization error in a region of interest. An alternative is to use the information gain in the test points, which corresponds to using the logarithm of the predictive variance.



(a) Normal, $w_{1,2} = 0.2$ (b) Normal, $w_{1,2} = 0.3$ (c) Normal, $w_{1,2} = 0.4$

(d) Uniform, $w_{1,2} = 0.2$ (e) Uniform, $w_{1,2} = 0.7$ (f) Uniform, $w_{1,2} = 1$

Figure IV.3.: In this figure we have plotted A-optimal designs of 200 points (•) with 5 initial samples (×). The noise was set to a small level ($\sigma = 10^{-5}$, $w_o = 1$). Panels (a)–(c) show designs for a normal, $\mathcal{N}(0,1)^2$, panels (d–f) for a uniform, $\mathcal{U}(0,1)^2$, input distribution. In contrast to random samples, A-optimal designs tend to spread the samples well apart from each other, where the length scales control the distances between the points.

expected utility function for GPs has first been reported in (Pfingsten, 2006). The derivation is the following:

Assume we add a sample $\tilde{\mathbf{x}}$ to the dataset $\mathcal{D}$. The integral over $\mathbf{y}$ drops out, as the predictive variance is independent of the targets. Using the definitions in (II 3.17), the change in the predictive variance results in[8]

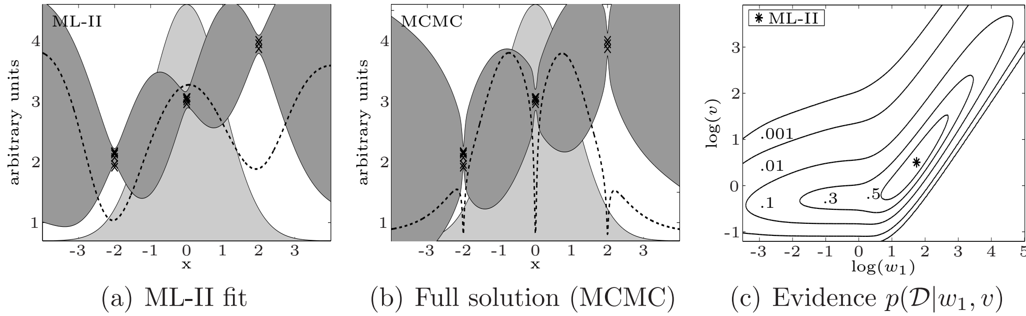
$$\operatorname{var}[y|\mathbf{x}, \mathcal{D}, (\tilde{\mathbf{x}}, \tilde{y})] - \operatorname{var}[y|\mathbf{x}, \mathcal{D}] \quad = \quad -\frac{\left[k(\mathbf{x}, \tilde{\mathbf{x}}) - \mathbf{k}(\mathbf{x})^T Q^{-1} \mathbf{k}(\tilde{\mathbf{x}})\right]^2}{\operatorname{var}[\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}]}\,, \tag{2.4}$$

which, through integrating over $p(\mathbf{x})$, leads to

$$\mathrm{U}_A(\tilde{\mathbf{x}}, \boldsymbol{\theta}|\mathcal{D}) \quad = \quad \text{const} + \int \mathrm{d}\mathbf{x}\, p(\mathbf{x})\, \frac{\left[k(\mathbf{x}, \tilde{\mathbf{x}}) - \mathbf{k}(\mathbf{x})^T Q^{-1} \mathbf{k}(\tilde{\mathbf{x}})\right]^2}{\operatorname{var}[\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}]} \tag{2.5a}$$

$$\quad = \quad \text{const} + \frac{\left[l(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) + \left(Q^{-1}\mathbf{y}\right)^T L \left(Q^{-1}\mathbf{y}\right) - 2\left(Q^{-1}\mathbf{y}\right)^T \mathbf{l}\right]}{\operatorname{var}[\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}]}\,. \tag{2.5b}$$

[8]The predictive variance is given by (II 3.17b). The change for an additional sample can be derived using a rank-one update of Q^{-1}. Note that the utility is valid for both, noisy ($\sigma \neq 0$) and exact ($\sigma = 0$) observations y.



(a) ML-II fit (b) Full solution (MCMC) (c) Evidence $p(\mathcal{D}|w_1, v)$

Figure IV.4.: Active learning, univariate example. Approximate inference using a GP prior on 15 training samples ($\times$): ML-II (a) and an average over the hyperparameters using MCMC (b). The $\pm 2\sigma$ predictive distribution is shaded in dark gray. The dashed line indicates the expected utility (2.5b) of a potential new training sample with the input distribution indicated by the light shade. Note that ML-II and MCMC give qualitatively different results. Panel (c) shows the contours of the posterior for v and w_1 relative to the maximum ($\star$) at the correct noise level: Much posterior mass is located at smaller length scales, and the confident ML-II estimate is inadequate.

We have used the definition[9] $l(\mathbf{x}', \mathbf{x}'') = \int d\mathbf{x}\ p(\mathbf{x})\ k(\mathbf{x}, \mathbf{x}')k(\mathbf{x}, \mathbf{x}'')$ which we have already introduced in (III 3.16). The explicit solutions for the squared exponential kernel function (II 3.20) and Gaussian or uniform input distribution are given in the appendix B, where we also treat the case of arbitrary factorizing input distributions.

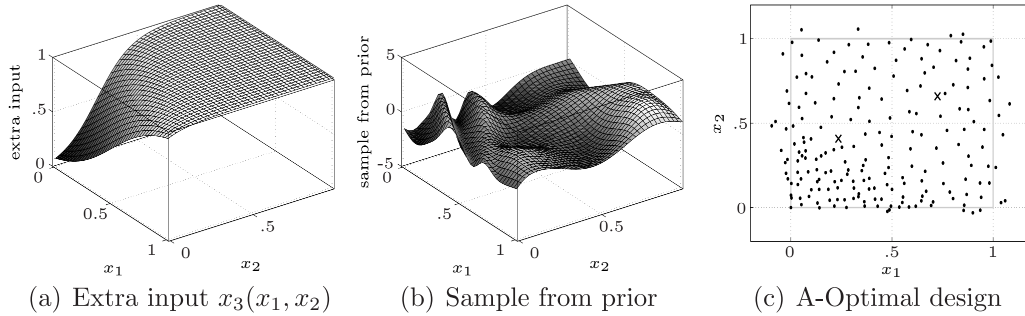
2.2. The ML-II approximation

In the expected utility $\mathrm{U}(\tilde{\mathbf{x}}, \boldsymbol{\theta}|\mathcal{D})$ (2.5b) the hyperparameters $\boldsymbol{\theta}$ are assumed to be known. In the designs which we have shown so far we have kept them constant for illustration, however, in our experiments they are updated after each query to adjust to the functions' properties.

As required in Bayesian decision theory, unknown parameters need to be integrated out using the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$. However, an analytical solution is infeasible (chapter II 1). Good numerical estimates, e.g. using MCMC, are computationally expensive and require an experienced user. A simpler alternative is the ML-II estimate, where the posterior is approximated by a delta distribution around its mode $\boldsymbol{\theta}^*$ (2.12), and the expected utility simply becomes $\mathrm{U}(\tilde{\mathbf{x}}|\mathcal{D}, \boldsymbol{\theta}^*)$.

Problems using ML-II in extreme situations. The ML-II approximation is widely used for inference. However, there are critical situations in which there is a qualitative difference between ML-II and MCMC estimates for A-optimal scores: In figure IV.4 we show such a case. In this univariate example

[9] As for the covariance function k we use:
$\mathbf{l}(\tilde{\mathbf{x}}) \in \mathbb{R}^N$, $L \in \mathbb{R}^{N\times N}$ with $[\mathbf{l}(\tilde{\mathbf{x}})]_\ell = l(\mathbf{x}_\ell, \tilde{\mathbf{x}})$ and $L_{i\ell} = l(\mathbf{x}_i, \mathbf{x}_\ell)$.



(a) Extra input $x_3(x_1, x_2)$ (b) Sample from prior (c) A-Optimal design

Figure IV.5.: Optimal design for a nonstationary GP model. Augmenting the inputs by a function of x_1 and x_2 as an extra dimension (panel a), leads to a new, nonstationary kernel function. A sample from the corresponding GP prior is shown in (b), and the A-optimal design is given by panel (c).

we have used 15 training samples which we have placed around only three locations. Gray shades indicate the Gaussian input distribution $p(\mathbf{x})$, and the predictive distribution. Note that ML-II (a) returns optimistic predictions with significantly smaller predictive variance than MCMC (b) and maximal expected utility (dashed line) far from the origin and at the maximum of $p(\mathbf{x})$ where a number of samples is already given. This seems unreasonable as no samples have been queried between ± 2 and 0. In contrast, the MCMC solution (b) shows exactly the characteristic which we would have expected intuitively: The utility is very small where targets have been observed, and maximal around ± 1. Panel (c) displays why the ML-II solution is inadequate in this example. As the training samples are placed only around three locations, the length scales' posterior distribution has its mass spread from $w_1 = 0.1$ to 20, and MCMC samples with shorter w_1 contribute strongly to the variance in between samples. These are neglected by ML-II, which fixes w_1 to 6.

2.3. Nonstationary GP priors

As we have outlined in chapter II 3.2, a common assumption when specifying a GP prior is stationarity, i.e. that the covariance between function values only depends on the distances $(\mathbf{x} - \mathbf{x}')$, not on their location. It is far more difficult to specify a GP prior allowing the function to have different properties in different parts of the input space.

Assuming stationarity implies that we exclude the case where the function varies fast in one region of the input space, while being very smooth elsewhere. Unfortunately, just this is the most interesting case for active learning: Using few samples where the function is smooth, the learning scheme could place samples more densely in the "interesting" region.

Gramacy et al. (2004) propose to use nonstationary Gaussian process trees to locate and exploit such regions for design. Pfingsten et al. (2006b) approach nonstationarity via a latent input dimension.

Figure IV.5 illustrates an A-optimal design for a nonstationary GP with one given extra input: Panel (a) shows the extra input as a function of the first two,

Algorithm 2 Greedy A-optimal active learning

Require: N_o initial samples $\mathcal{D}_{N_o}$. Input distribution $p(\mathbf{x})$.
1: **for** $\ell = N_o + 1$ to N **do**
2: find the ML-II estimate $\boldsymbol{\theta}^*$ using $\mathcal{D}_{\ell-1}$.
3: find $\mathbf{x}_\ell \leftarrow \text{argmax}_{\tilde{\mathbf{x}}} \text{U}_A(\tilde{\mathbf{x}}|\mathcal{D}_{\ell-1}, \boldsymbol{\theta}^*)$.
4: query target y_ℓ to obtain new dataset $\mathcal{D}_\ell \leftarrow \mathcal{D}_{\ell-1} \cup \{(\mathbf{x}_\ell, y_\ell)\}$.
5: **end for**

$x_3(x_1, x_2)$, which introduces extra variability where x_3 is not constant. We have used a squared exponential kernel (II 3.20) with $v = 1$, $w_1 = w_2 = 0.2$ and $w_3 = 0.1$ for the extra input. Panel (b) shows a sample from the prior, which exhibits fast variations in the leftmost corner. Accordingly, the corresponding designin panel (c) turns up to place samples more densely in this region than elsewhere. Note that active learning, in contrast to experimental design, would be able to detect the nonstationarity, enabling the learning scheme to concentrate on interesting regions.

2.4. The greedy A-optimal scheme for active learning

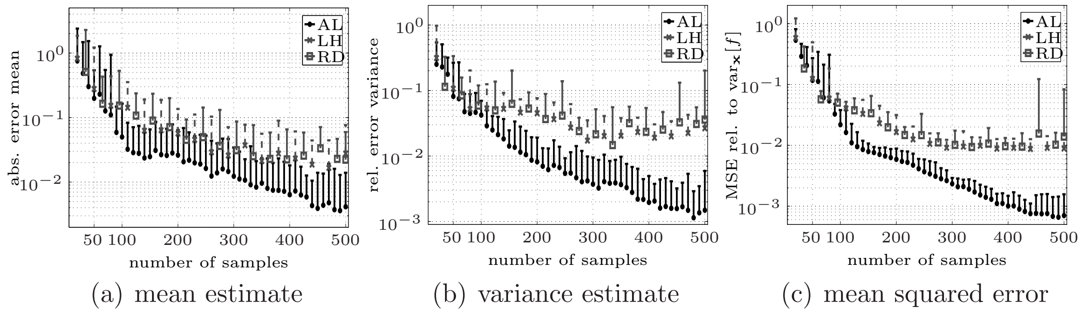
In the preceding sections we have outlined possible approaches and approximations to Bayesian active learning. In the following we outline the implementation which we have used for our experiments.

In section 1.2 we have described the conceptual difference between Bayesian experimental design and active learning as commonly used in machine learning. For our implementation we have chosen the greedy active learning scheme, which updates the model after each query, yet plans only one step ahead. From the shown low dimensional examples we conjecture that the greedy approximation is usually appropriate, while updating the GPs unknown hyperparameters has a significant influence on the design.

For sensitivity analysis and standard regression setups, where the underlying function can be queried in form of a simulation software, the A-optimal design (2.1) is the most natural. We use the exact averages over $p(\mathbf{x})$ (2.5b), which we optimize by choosing the maximum value out of a pool of 10 000 samples from $p(\mathbf{x})$ and 10 000 samples from a distribution with twice the variance of $p(\mathbf{x})$. The brute-force maximization can be replaced by a gradient-based scheme, however, we expect a large number of local minima where the optimization could get caught.

The example in 2.2 shows that the ML-II approximation might lead to suboptimal designs. Nevertheless, a full MCMC solution is not feasible as it can hardly be adapted for non-expert use, and thus we use the ML-II approximation. The hope is that ML-II leads to good results in practice, which was in fact verified in an extensive empirical study (see section 3).

Nonstationary GPs (2.3) are difficult to handle numerically, and it is not clear whether such priors are useful for high dimensional models. We believe that this class of models is very interesting for future research, however, for our application of active learning we choose the simpler stationary model with



(a) mean estimate (b) variance estimate (c) mean squared error

Figure IV.6.: Friedman's function. Learning rates of passive learning—using independent samples from $p(\mathbf{x})$ (RD) and Latin Hypercube (LH) designs—compared to greedy active learning (AL). We consider the mean (a) and variance estimate (b) for SA, and the MSE on samples from $p(\mathbf{x})$ (c).

squared exponential kernel (II 3.20). Nonstationarity is introduced to sensitivity analysis via the nonconstant weighting given by $p(\mathbf{x})$. Our approach is subsumed by algorithm 2.

3. Evaluation and use in practice

Our active learning scheme is rigorously derived from the Bayesian expected utility, which addresses the generalization error. However, for the reasons we have outlined above, it is not clear whether the scheme can actually save queries in practice. We have benchmarked the algorithm in comparison to passive learning on a number of problems of differing complexity, which we believe to cover problems that are important in practical application:

We consider the benchmark problems for sensitivity analysis, which have been introduced in chapter III. Here we assess the performance of GP regression and Bayesian Monte Carlo, directly comparing the generalization error of the GP emulator using the mean squared error (MSE) on samples from the input distribution, and the accuracy of the estimates for the first two moments of the uncertainty distribution. The examples are divided into analytical benchmark problems (section 3.1) and simulation models from development at Robert Bosch GmbH (section 3.2).

MSE

3.1. Artificial benchmark functions

Friedman's function. The results for the example on the basis of Friedman's function[10] are shown in figure IV.6. We have initialized our greedy active learning scheme (AL) with $N_o = 20$ random samples from $p(\mathbf{x})$, and subsequently added 480 optimal queries. The plots compare the error to those obtained using Latin-Hypercube designs (LH) and independent samples from $p(\mathbf{x})$ (RD) for varying design sizes. Panels (a–c) show the accuracy of the

[10] For the definition see chapter III 4.1, p. 54.

	relative error in %		ratio
	passive learning: LH	active learning: AL	means
$\mathrm{mean}_{\mathbf{x}}[f]$	0.34 [0.07, 0.65]	0.52 [0.08, 0.94]	**1.55**
$\mathrm{var}_{\mathbf{x}}[f]$	2.66 [0.79, 4.83]	0.81 [0.07, 1.70]	**0.30**
MSE	1.05 [0.71, 1.39]	0.30 [0.26, 0.37]	**0.28**

Table IV.1.: Convergence of active and passive learning for Friedman's function with noise. The figures represent the performance on 10 independent designs with 500 samples (mean [best, worst]). The last column contains the ratio of mean AL- and LH-error.

BMC mean and variance estimate, and the mean squared error of the GP emulator on 10 000 independent test points from $p(\mathbf{x})$. Each errorbar shows the mean and maximal error out of 20 independent runs. We have omitted the minimal values to increase the legibility of the plots.

AL significantly outperforms both passive sampling schemes on all quantities of interest on designs of more than approximately 100 points. At 500 samples the AL mean estimate is around 5 times more accurate compared to passive learning, for the variance estimate and the MSE optimal designs are better by a factor 10. A surprising fact is that for all measures there is hardly any difference between independent samples and Latin Hypercube designs. Note also that the fluctuations in the AL designs are much smaller than for random samples, as the sampling scheme is to large extend deterministic[11].

The MSE in panel (c) appears to be dominated by three phases during active learning: Up to 100 samples the function's structure is only roughly captured by the GP, so that the active planning can hardly profit from prior measurements. The gap between optimal and random designs increases very rapidly from 100 to 450 samples. Our intuition is that the structure is well approximated in this phase, so that the planning procedure results to be very effective. From 450 samples on the improvement slows down. We believe that the saturation at an accuracy of 0.1% relative to the correct variance $\mathrm{var}_{\mathbf{x}}[f]$ might be due to a slight mismatch between prior and data.

Noisy observations In our experiments we aim mainly at assessing the performance of active learning for noiseless observations from computer experiments. Therefore we have not added noise to the artificial examples. To evaluate whether our learning scheme also improves the learning rate in the presence of noise, we consider here the Friedman function adding normal noise with unit variance, as proposed in the original problem by Friedman (1991). The results are given in table IV.1.

On a separate test set of 10 000 samples (without noise) from $p(\mathbf{x})$, active learning reduces the MSE by a factor of 3.5 and the relative accuracy of the variance estimate by a factor of 3.3. Surprisingly, the mean estimate is by a factor of 1.5 worse than for passive learning.

[11]Due to the logarithmic scale the AL errorbars appear stretched.

Model	D	N	**passive learning: LH**	**active learning: AL**	**ratio**
			mean squared error relative to $\mathrm{var}_{\mathbf{x}}[f]$		means
1	2	100	$5.6\ 10^{-8}$ $[1.3\ 10^{-8},\ 2.2\ 10^{-7}]$	$1.1\ 10^{-8}$ $[4.7\ 10^{-9},\ 1.8\ 10^{-8}]$	**0.20**
2(a)	6	600	$1.2\ 10^{-4}$ $[8.3\ 10^{-5},\ 2.2\ 10^{-4}]$	$9.4\ 10^{-5}$ $[6.2\ 10^{-5},\ 1.4\ 10^{-4}]$	**0.76**
2(b)	20	500	$3.5\ 10^{-2}$ $[3.1\ 10^{-2},\ 4.5\ 10^{-2}]$	$3.5\ 10^{-2}$ $[2.6\ 10^{-2},\ 5.2\ 10^{-2}]$	**1.0**
3(a)	2	30	$6.2\ 10^{-6}$ $[3.6\ 10^{-7},\ 5.0\ 10^{-5}]$	$2.2\ 10^{-7}$ $[1.3\ 10^{-7},\ 4.8\ 10^{-7}]$	**0.04**
3(b)	2	300	$7.0\ 10^{-6}$ $[4.3\ 10^{-7},\ 3.2\ 10^{-5}]$	$3.0\ 10^{-8}$ $[2.4\ 10^{-7},\ 3.8\ 10^{-7}]$	**0.04**
4	8	500	$1.3\ 10^{-2}$ $[1.1\ 10^{-2},\ 1.9\ 10^{-2}]$	$9.9\ 10^{-3}$ $[8.9\ 10^{-3},\ 1.1\ 10^{-2}]$	**0.79**
5	3	500	$4.0\ 10^{-4}$ $[2.0\ 10^{-4},\ 6.6\ 10^{-4}]$	$1.4\ 10^{-5}$ $[1.3\ 10^{-5},\ 1.7\ 10^{-5}]$	**0.04**

Table IV.2.: Convergence of active and passive learning on SA benchmark problems (appendix B). The figures represent the performance on 20 independent designs (mean [best, worst]). The last column contains the ratio of mean AL- and LH-error.

Our explanation for the differing results on the mean estimate is the following: active learning cannot improve the uncertainty in the offset due to noisy observations. According to the standard bound given by the central limit theorem, the relative standard deviation of the mean estimate with 500 observations is 0.39%, which is in the range of accuracy of both learning schemes. We conjecture that the error in the mean estimate is dominated by the uncertain offset due to the noise, which means that the bad performance of AL is not significant. In contrast, the maximum MSE for active learning is smaller than the minimum MSE for random sampling.

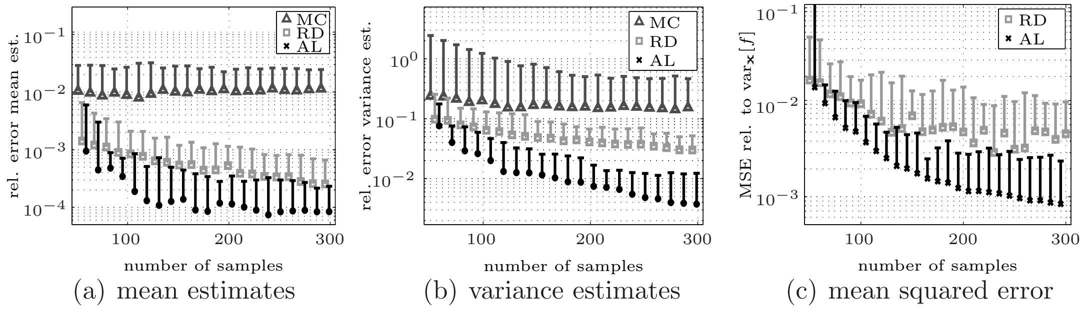
Many heuristic approaches to active learning tend to fail in the presence of noise, and there has been an extensive discussion on how to solve the problem (Balcan et al., 2006). We believe that our learning scheme does not show this behavior, as it is rigorously derived from the Bayesian expected loss. As the used GP models are probabilistic, they provide a notion of uncertainty and separate noise-variance naturally from predictive uncertainty.

SA benchmark functions. A large variety of benchmark functions for sensitivity analysis has been proposed by a number of authors. We have used a set of these problems to evaluate the BMC scheme, as defined in the appendix B. The problems are all nonlinear, yet of very different complexity with two to twenty active dimensions.

Table IV.2 shows the results of passive and active learning, comparing the MSE on a fixed number of samples[12]. Except for model 2(b) and 4 we have used the same number of samples as in table III.1, p. 58, where we compared MC and BMC. Models 2(b) and 4 turned out to be extremely difficult, requiring 2 000 and 3 000 samples, respectively. Therefore we have used only 500 samples to limit the computational expense.

The ratio of the MSE for active and passive learning varies from 0.04 for problems 3(ab) and 5 to 1 for problem 2(b). In particular on the low dimensional examples active learning leads to a great improvement, while we obtained only a slight or no improvement for the hardest problems.

[12]We have used $N_o = 5$ (model 1, 3(a,b)) and 20 (model 2(a,b), 4, 5) initial random samples.



(a) mean estimates (b) variance estimates (c) mean squared error

Figure IV.7.: Learning rates for the PS model: The estimates for the output's mean and variance using the MC method, passive and active Bayesian quadrature are plotted in panels (a) and (b). The mean squared error for active and passive learning is shown in panel (c) relative to $\text{var}_\mathbf{x}[f]$. The error bars indicate the median and maximum value out of 35 runs.

This connection is related to our results for the comparison of MC and BMC, where we found that BMC leads to more improvement where the function was easier to capture by the GP prior: As we have seen in the Friedman example, the active learning scheme can only start being effective when parts of the function are captured well by the GP. We believe, that the function in example 2(b) is too difficult to make AL work on only 500 samples. Note, however, that even here AL performs as well as passive learning and does not waste simuilation runs.

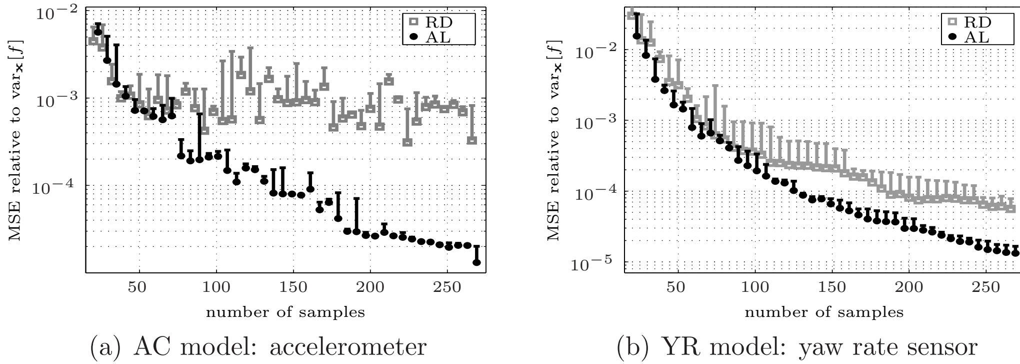
3.2. Examples from development

The following examples are fully features models of MEMS sensors, which we have introduced in chapter III, section 4.2.

PS model. Figure IV.7 shows the empirical comparison of MC, active and passive BMC for the PS model (chapter III 4.2 p. 49). The active learning scheme has been initialized with $N_o = 20$ random samples. We observe that AL outperforms random sampling on each quantity for more than 75 samples. The accuracy at 300 samples is improved by roughly a factor of five.

AC and YR model. The AC model has been introduced in chapter III 4.2 p. 57, the YR model can be found on p. 59. In both cases we have initialized the active learning scheme with $N_o = 20$ random samples. For the previous examples we have seen that the accuracy of the moment estimates for BMC tightly corresponds to the MSE. Hence, to keep the presentation brief, we only show the MSE of active and passive learning against the number of training samples (figure IV.8).

We observe that active learning significantly increases the accuracy starting at around 100 samples in both problems. For designs with 270 samples we gain a factor of 15 (AC model) and 5 (YR sensor).



(a) AC model: accelerometer

(b) YR model: yaw rate sensor

Figure IV.8.: Learning curves for the model of the AC model (a) and the YR model (b). The markers indicate the median of 6 (a) and 3 (b) runs, the error bars cover the interval up to the maximal value. We compare the performance on designs randomly sampled from $p(\mathbf{x})$ (RD) and A-optimal designs (AL), where the mean squared error is measured relative to $\mathrm{var}_{\mathbf{x}}[f]$.

4. Discussion

The asymptotical learning rates for stationary regression models are discouraging, as they state that active learning schemes cannot lead to faster convergence than passive learning. However, these results do not cover the case where few samples are available, which is in practice of main interest for active learning. Accordingly, our approach to active learning was motivated by the need of an efficient way to compute sensitivity measures for computationally expensive computer models of MEMS.

Our approach is rigorously derived from a Bayesian expected utility, and—in contrast to many previous works—we have avoided heuristics and numerical approximation where possible: To our knowledge our approach is the first to avoid the pool-approximation for A-optimal designs. Therefore, the evaluation of our utility function is computationally as cheap as a simple prediction for GP regression at that input, and the method is easily implemented.

As indicated by asymptotical learning rates for active learning and its sensitivity to misspecifications of the model, it is not clear a-priori whether a learning scheme improves the learning rate in practice. On the example of the ML-II approximation we have shown that in particular approximating the expected loss might have undesirable effects. To test the fitness of our active learning scheme we have therefore performed experiments on various benchmark problems, and case studies on fully featured models from the development of MEMS sensors.

Our experiments show that—for a fixed number of samples—active learning can greatly reduce the generalization error of the GP model in a region of interest, and thus increase the accuracy of a SA while keeping simulation time constant.

The improvement is particularly large for smooth functions, including the high dimensional sensor models where only a restricted number of parameters have significant impact. As active learning can only become efficient when

enough structure of the approximated function is captured, the scheme becomes less attractive for very difficult problems. However, our experiments have shown that active learning never performs significantly worse than passive learning—we loose the effort spent in computing the optimal design, but queries are not wasted at uninformative positions.

Our approach to active learning is very efficient and robust—due to its principled derivation. It can thus be integrated into our engineering tool for analyzing computer experiments, where the simulation code is automatically called at optimal settings until a required accuracy is obtained.

V. Feature selection for troubleshooting

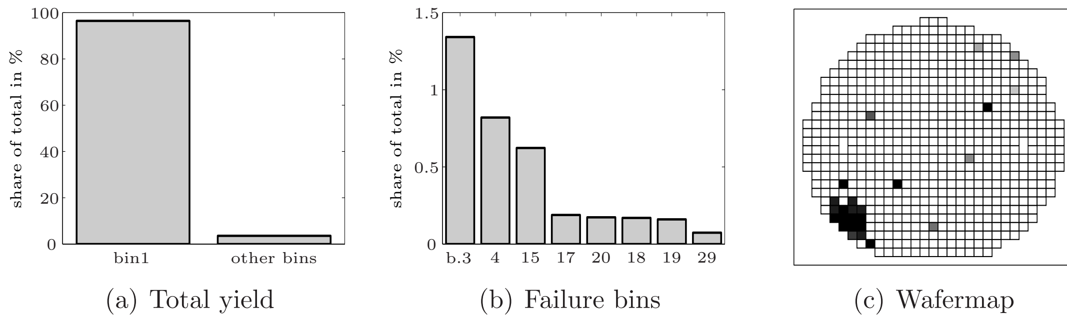
In chapter I we have discussed semiconductor manufacturing, including the available data and typical problems one faces in mass production. Though all processes are tightly controlled in mass production by keeping a large number of in-line measurements in a small tolerance window, the faultless functioning of all devices cannot be guaranteed. Each device is therefore controlled in rigorous quality checks at the end of the production line to verify that the specifications are met. A small number of devices is always found to fail the quality tests due to particle contamination and random fluctuations—and in rare cases a systematic error leads to an increased failure rate. While it is the aim of robust designs (see chapter III) to minimize random failures, here we present a scheme to detect and localize systematic errors due to malfunctioning processes.

Our approach for troubleshooting is a novel application of *feature selection*, which is also described in (Pfingsten et al., 2005). In contrast to previous methods, the approach does not model the structure of the fabrication, and is thus not restricted to a specific process.

In order to detect systematic errors we make use of data from finishing quality checks. These consist of tens of numerical measurements, which we use to separate failures into classes that are likely to be related to different root causes. The classification is typically done manually when certain error patterns are repeatedly found by the operator, and for semiconductor manufacturing it is possible to tailor an automatic error detection. We outline several mechanisms in section 1.1. In section 1.2 we describe how to combine the lot-history and data from other sources to obtain a list of features which might explain the errors.

When dealing with a moderate number of possible root causes and a large number of samples, troubleshooting is an easy task. The crux of the troubleshooting problem is that we usually have less than a hundred detected errors while there are thousands of potential causes. Also, potential causes need to be treated jointly as we expect that an unfortunate concurrence of several incidences causes the problem. The problem which we end up with in troubleshooting has been studied extensively in machine learning as feature selection, where powerful algorithms have been developed. We discuss the generic problem of feature selection in section 2, introducing the basic concepts (2.1 and 2.2). Our implementation is described in section 2.3.

Our approach has been cast into a tool for the Bosch semiconductor foundry, where it has proven valuable in several incidences. In section 3 we present a case study on four real-world problems from the foundry, which has served



(a) Total yield (b) Failure bins (c) Wafermap

Figure V.1.: The results of on-wafer tests can be used in different levels of detail. The total yield (a) compares only the total number of good and bad dies. The failures can be classified into several bins (b), which may contain information about the root cause. The position of failures on the wafer (c) indicates systematic errors through deviations from a uniform distribution.

us to test the algorithm before clearing it for common usage. The results are discussed in section 4.

1. Data preprocessing

A crucial step in machine learning is the preprocessing, as this is where great parts of the expert knowledge enter the analysis. In the presented approach to troubleshooting we have used known algorithms, and the main contribution of this work was to identify the analogy of troubleshooting and previously studied problems.

In the following section 1.1 we describe the detection of systematic errors in semiconductor manufacturing. In our feature selection approach these errors are sought to be explained by a small number of features, given by the way of a lot through the manufacturing line. In section 1.2 we describe how to merge all potential causes in a standardized format which suits troubleshooting.

1.1. Detection of systematic errors

The outset of troubleshooting is the detection of systematic deviations which are caused by an abnormal behavior of the manufacturing line. Depending on the product which is under consideration, these deviations may be noticed by a great variety of attributes and are usually a matter of manual classification.

For semiconductor manufacturing we have the possibility to define some characteristics which can be detected automatically. See chapter I for an overview on a typical semiconductor foundry and the available data. During production wafers are the basic workpieces and pass through the manufacturing line in groups (lots), which represent a batch of thousands of single devices. Each device is tested individually in electrical on-wafer quality checks. In figure V.1[1] we illustrate the levels of detail, in which the results of the on-wafer

[1]The plots show artificial data.

tests may be analyzed: The yield (panel a) is the rate of compliance with the specifications and can be defined for single wafers as well as for the lot as a whole. It is widely used, in particular to analyze the cost-effectiveness of the foundry, however, it is often not specific enough for troubleshooting as it averages over all possible causes for failure. A minor refinement to the overall yield is to brake down the failure rate into error classes according to the specification which is violated (panel b). Using the position of failed dies on the wafer we can go a step further and explore their spatial distribution. Panel (c), for example, shows a suspicious spot with concentrated failures at the bottom left of the wafer. By considering such patterns we can make sure not to miss on rare systematic failures in a background of random errors. Patterns on wafermaps as indicators for systematic losses have been studied in several previous publications:

Duvivier (1999) describes a scheme to identify regions where dies tend to fail more often than on average, and Fountain et al. (2002) and Riordan et al. (2005) use these dependencies to predict failures before on-wafer tests are performed[2]. The detection of patterns in a database of wafermaps is an unsupervised learning problem, where no pre-classified training set is available—the main goal is to discover unknown patterns, not to assign wafers to known templates. Nicolao et al. (2003) compare several methods for unsupervised classification using a set of artificial examples. Defect maps, which capture the quality of the wafer's surface at several production stages, can complement maps of on-wafer tests.

In our case study we have not made use of automatically extracted patterns, as critical regions on the wafers had already been identified by the operators. Using these templates, we could identify affected wafers comparing the failure rates on critical regions to the overall yield. As the detection of patterns on wafermaps has been extensively studied in the above works, we concentrate here on how to use the data to locate root causes.

As we have mentioned above, flaws in the production line can lead to systematic failures with very different attributes. Our troubleshooting approach is independent of the errors' specific attributes and how they are detected, we only distinguish "regular" and "conspicuous" lots. If, for example, the occurrence of systematic patterns serves us as an attribute, we define a suspicious lot as one where patterned wafers are found. In order to use the yield we need to define a threshold below which we consider a lot to be conspicuous. Let us denote the number of lots in the dataset by N. The results of the error detection can be collected in a binary vector

$$\mathbf{y} = (y_1, y_2 \dots y_N)^T \in \{0,1\}^N \,, \tag{1.1}$$

where an entry $y_\ell = 0$ stands for lot number ℓ being unaffected, and $y_\ell = 1$ for a lot being affected by the error.

[2]The cut down of on-wafer tests can be valuable as testing contributes with a non-negligible fraction to the total costs of a device.

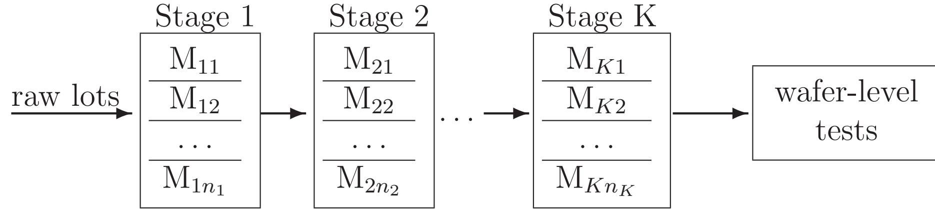


Figure V.2.: Scheme of a serial-group manufacturing line. The raw lots undergo manipulations in K production stages, and in each stage ℓ we have several machines $\mathrm{M}_{\ell 1} \dots \mathrm{M}_{\ell n_\ell}$ to choose from. Most lots take different paths through the machinery, as the allocation is designed to maximize the plant utilization. After processing, the functionality of each unit is tested on the wafer.

1.2. Features and root causes

When doing troubleshooting we need information about the production of each lot—in addition to the classification $\mathbf{y}$ into regular and irregular given through error detection. The information may consist of in-line measurements or other, more general details. Here we can think of the information that a certain supplier has delivered a chemical substance, or that the lot has been processed during a night shift. Suppose we have defined a list of D features out of which we believe to be able to read a root cause—no matter whether we believe that a single feature or a combination of them has caused the observed errors. Using a binary vector

feature

$$\mathbf{x}_\ell = (x_{\ell 1}, x_{\ell 2} \dots x_{\ell D})^T \in \{0,1\}^D \tag{1.2}$$

we can describe for each lot ℓ which features j have been observed in its fabrication ($x_{\ell j} = 1$) and which have not ($x_{\ell j} = 0$). We will call the $\mathbf{x}_\ell$ *input vectors*, collecting them in a matrix $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_N)^T \in \{0,1\}^{N \times D}$.

input vector

Using a binary coding to record whether some feature has been active during production, we can combine arbitrary information about a lot. The binary vector can be expanded by in-line measurements, as our feature selection scheme is also valid for numerical data.

In our case study we use the lot-history to define plausible root causes. Recall the scheme of a serial-group manufacturing line introduced (figure V.2): The wafers are processed in K stages, and in each stage j we randomly choose one out of n_j equivalent machines. If we believe that the use of a machine in a specific stage causes the error, we obtain

$$D = \sum_{j=1}^{K} n_j \tag{1.3}$$

different features, each of which standing for a combination of a stage and a machine. In realistic problems we have to deal with approximately 5 machines in each of 400 stages and obtain $D = 2\,000$ features. Machines can be used in several stages in the foundry, and an error might be caused when a machine

is used anywhere in the processing independently of the stage. To account for this possibility we append features which we set to one if the according machine is used anywhere in the process.

Our strategy for troubleshooting is the following: We use an algorithm to extract combinations of features which contain information about the target $\mathbf{y}$. The interpretation is simple. We identify the root cause with the set of features which is suited best to predict whether an error occurs.

target

2. Feature selection

In the previous section we have described how to combine error detection with other data from production, to obtain a target vector and feature vectors which enlist its possible root causes. Section 2.1 introduces the generic concept of feature selection, where the objective is to identify small subsets of features which are informative with respect to the target. In the troubleshooting task this resembles the search for the root cause of the problem. In 2.2 we review previous approaches to feature selection to motivate our approach to troubleshooting, which we present in 2.3.

2.1. Objective in feature selection

In chapter II we have introduced methods to learn mappings $f(\mathbf{x})$ from inputs $\mathbf{x} \in \mathbb{R}^D$ to targets y. It is often the case in real-world problems that the mapping only depends on a subset of the given input dimensions (features). In the following we call a feature *irrelevant* if f does not depend on it, and accordingly call features which enter f *relevant*. To indicate the relevancy of features we define a vector $\boldsymbol{\sigma} \in \{0,1\}^D$ with an entry $\sigma_\ell = 1$ when the feature x_ℓ is relevant and $\sigma_\ell = 0$ when x_ℓ is irrelevant. We denote the number of relevant features, i.e. nonzero entries in σ by $D_{\boldsymbol{\sigma}}$.

irrelevant

relevant

To obtain good generalization performance an induction algorithm needs to be able to ignore irrelevant inputs. For the nearest neighbor classifier, for example, Langley and Iba (1993) study the importance of feature selection and show that the predictive performance rapidly decreases with the number of irrelevant features. The automatic relevance determination (ARD), which we have introduced in chapter II, automatically detects the importance of features by learning the appropriate length scales. The basic idea in ARD—and other methods which concentrate on informative features—is to encode our prior preference of simple mappings that depend on few features.

In the troubleshooting problem we are faced with an unfavorable proportion of thousands of features and only a tens of training instances. Any model has to restrict the number of active features, i.e. $D_{\boldsymbol{\sigma}} \ll D$, to be useful for these applications, and feature selection has therefore received a lot of interest in the machine learning community. An introduction is given by Guyon and Elisseeff (2003)[3], Blum and Langley (1997) review earlier works and discuss the basic

[3]The introduction is part of a special issue of the Journal of Machine Learning Research

approaches to the problem.

Assume we have made up our mind about the model $\mathcal{M}_{\boldsymbol{\sigma}}$ which we want to use to describe the relationship between the active features $\boldsymbol{\sigma}$ and the target. The choice of the subset $\boldsymbol{\sigma}$ is a model selection problem of the kind we have discussed in chapter II 2.1, and in the Bayesian framework we use the posterior probabilities $p(\mathcal{M}_{\boldsymbol{\sigma}}|\mathcal{D})$ as a basis for our decision. To encode our belief in models which depend on few features, we might use a prior of the type $p(\boldsymbol{\sigma}) = p(D_{\boldsymbol{\sigma}})$ which favors small $D_{\boldsymbol{\sigma}}$.

Non-probabilistic models usually optimize some loss function, where we can identify an empirical risk term which corresponds to the $(-\log)$ likelihood, and a regularization term which corresponds to the $(-\log)$ prior (Schölkopf and Smola, 2002, chapter 3 and 4). Just as we may introduce a prior which favors small sets of active features, we can adjust the regularization term such that it penalizes large $D_{\boldsymbol{\sigma}}$. A well known method, the *Lasso*, has been proposed by Tibshirani (1996). The Lasso is a linear model, $f(\mathbf{x}) = \sum_\ell w_\ell x_\ell$, where the mean-squared loss is minimized under the constraint $\sum_\ell |w_\ell| \leq \lambda$. As the extra parameter λ approaches zero, more and weights equal exactly zero, effectively making the corresponding features inactive. Weston et al. (2003a) propose a similar approach for linear and kernel methods, directly penalizing the number of active features $D_{\boldsymbol{\sigma}}$.

Any model selection is an optimization task, be the objective function the posterior probability as in Bayesian approaches, some other loss function, or the cross validation error. Let us name the objective function for feature selection, the score,

$$\mathcal{S}(\mathcal{M}_{\boldsymbol{\sigma}}, \mathcal{D}) \;, \tag{2.4a}$$

where $\mathcal{D}$ is the available training data. We will call its maximizer,

$$\boldsymbol{\sigma}^* = \underset{\boldsymbol{\sigma}}{\operatorname{argmax}}\; \mathcal{S}(\mathcal{M}_{\boldsymbol{\sigma}}, \mathcal{D}) \quad \text{with } D^* \text{ active features} \;, \tag{2.4b}$$

the optimal or most informative subset of features.

2.2. Wrappers, filters, and embedded methods

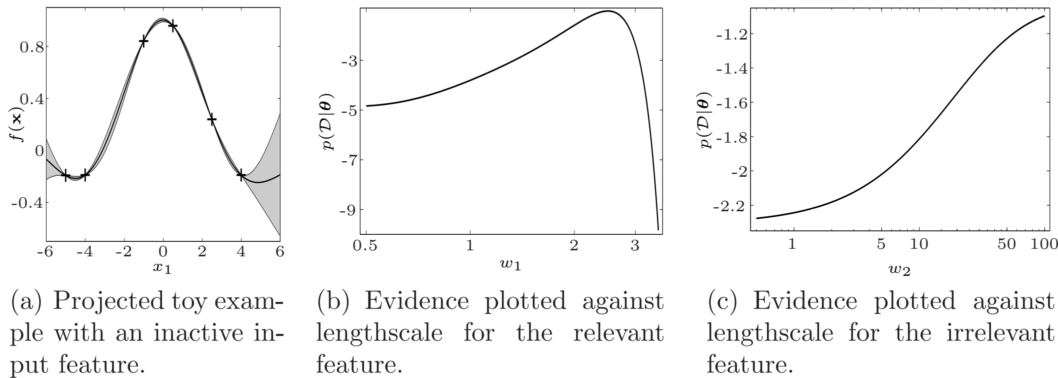
If we compare only small subsets with $D_{\boldsymbol{\sigma}}$ being one or two, we might be able to evaluate $\mathcal{S}(\boldsymbol{\sigma})$ (2.4) for all possible subsets, but as the number of subsets of length smaller or equal $D_{\max}$ is

$$\operatorname{card}\{\boldsymbol{\sigma}|D_{\boldsymbol{\sigma}} \leq D_{\max}\} = \sum_{d=1}^{D_{\max}} \binom{D}{d} \;, \tag{2.5}$$

an exhaustive search is impossible for large D and $D_{\max}$[4]. The approaches to the optimization problem are manifold, all are necessarily suboptimal and are

(Kaelbling, 2003) on the subject.

[4]Take, for example, a problem with a moderate number of $D = 100$ features and $D_{\max} = 5$ relevant features. In order to do an exhaustive search we need 919 days of computation if the evaluation takes one second per score.



(a) Projected toy example with an inactive input feature.

(b) Evidence plotted against lengthscale for the relevant feature.

(c) Evidence plotted against lengthscale for the irrelevant feature.

Figure V.3.: The ARD mechanism on a toy example with two input features. Panel (a) shows the projection of the 2-dimensional training data (+) with the GP fit (mean (—) and predictive uncertainty (2σ, shaded)). Panels (b) and (c) show the evidence for varying length scales. While w_1 is optimal around 2.5, w_2 is driven to values $w_2 \gg \max_{i\ell} \|x_{i2} - x_{\ell 2}\|$, thus eliminating x_2 from the covariance function.

designed to restrict the search to few feature sets which seem promising. In the following we outline different approaches. According to Blum and Langley (1997) we have grouped the methods into three classes, *embedded methods*, *wrappers* and *filter methods*.

Embedded methods. Our first example for feature selection has been ARD, as introduced in chapter II 3.2. ARD belongs to a family of methods which do not directly consider the combinatorial problem (2.4), instead tuning a smoothed problem with continuous "importance" parameters.

embedded methods

In figure V.3[5] we illustrate ARD on a toy example with two input dimensions. The output f is a function of the first parameter only, making the second feature irrelevant. Panel (a) shows the fit to the data, which is obtained by maximizing the evidence with respect to the hyperparameters $\boldsymbol{\theta}$. From a projection to the first length scale parameter w_1 (b), we observe that the evidence is maximal for $w_1 \approx 2.5$. The other length scale w_2 (panel c) maximizes the evidence at $w_2 \gg \max_{i\ell} \|x_{i2} - x_{\ell 2}\|$, thus eliminating the second feature.

Smooth approximations to the feature selection problem have also been considered for the support vector machine (SVM), e.g. by Weston et al. (2003a) and Bradley et al. (1998). Smoothed problems can be solved more efficiently than combinatorial tasks as they ease a search for local extrema. We can, however, not expect to be dealing with a convex problem where the maximum is unique, and searching for a global maximum might therefore still be impossible for realistic problems.

[5] The 2nd feature, not shown in the plots, was randomly drawn from $U(0,1)$ and does not enter the function $f(\mathbf{x})$. We have used the ARD squared exponential covariance function (3.20), and the ML-II approach, where the hyperparameters $\boldsymbol{\theta}^*$ are tuned to maximize the evidence $p(\mathcal{D}|\boldsymbol{\theta})$. Panel (b) and (c) show the evidence for varying length scales, while the other parameters are set to $\boldsymbol{\theta}^*$.

Smoothed approximations are examples for *embedded methods*, described in detail by Lal et al. (2006). The characteristic of embedded methods is that they do not search blindly for optimal sets of features using the model as a black box. Instead, they are tailored to the induction algorithms and use internal quantities of the model in addition to outcome of $\mathcal{S}(\boldsymbol{\sigma})$. To come back to the ARD example, the optimization is made efficient by the use of the evidence's gradient with respect to the hyperparameters $\boldsymbol{\theta}$.

The mentioned Lasso is another embedded method, just as the *optimal brain damage* for neural nets by Cun et al. (1990) and the related *recursive feature elimination* for SVMs (Guyon et al., 2002). Both latter methods start with the full set of features and eliminate them according to their impact on the objective function. Instead of computing the change of $\mathcal{S}(\boldsymbol{\sigma})$ by re-evaluating the predictor, the methods estimate the change using Taylor approximations.

Recursive feature elimination ranks the features one-by-one and can therefore not propose to remove (or include) related groups of features. The situation is similar in the Bayesian setting when Gibbs sampling is used to propose new $\boldsymbol{\sigma}$s:
Gibbs sampling compares the conditional probabilities $p(\sigma_\ell = 0|\boldsymbol{\sigma}^{\setminus\ell}, \mathcal{M}, \mathcal{D})$ and $p(\sigma_\ell = 1|\boldsymbol{\sigma}^{\setminus\ell}, \mathcal{M}, \mathcal{D})$[6], i.e. we only look at single features while leaving the others fixed. Nott and Green (2004) resolve this constraint by applying the Swendsen-Wang algorithm to the problem, which had originally been developed to solve the Ising model. The algorithm proposes changes not one-by-one, but finds connections between the features and accordingly proposes the removal and inclusion of whole groups of features.

wrappers

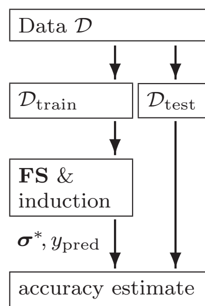


Figure V.4.: Accuracy estimate for feature selection.

The wrapper approach. Wrappers treat the induction algorithm as a black box and rate sets of feature only on the basis of the predictive performance of the constrained model $\mathcal{M}_{\boldsymbol{\sigma}}$. The idea to use the leave-one-out or cross validation error has been reviewed by John et al. (1994); Kohavi and John (1998), who also address the problem of overfitting. As they put it, we are likely to find a subset of features which is consistent with the holdout set by pure chance when the number of features is large. Therefore it is fundamental to test the selected features on another holdout set which has not been used for feature selection. We have sketched the validation scheme in figure V.4.

While embedded methods tackle the optimization problem by using model specific measures for the impact of features, wrappers necessarily need to do without such guidance, computing the change in $\mathcal{S}$ by retraining the induction algorithm for each possible change. A common way to do a non-exhaustive search over the set of active sets $\boldsymbol{\sigma}$ is the so-called *plus-l-take-away-r* method as proposed by Stearns (1976), which considers nested subsets by adding l and omitting r features in several steps. An

[6]For details see (MacKay, 2003, chapter 29).

example is the *oblivion* by Langley and Sage (1994), which starts with a full set of features and leaves them out one-by-one ($l = 0, r = 1$). Almuallim and Dietterich's *focus* (1991) performs a forward search to build a set of features ($l = 1, r = 0$). The floating search of Pudil et al. (1994) adds single features and then omits features until $\mathcal{S}$ decreases ($l = 1, r$ flexible).

Greedy search schemes such as the (l, r) strategies are necessary to restrict the search to a reasonable number of subsets $\boldsymbol{\sigma}$, yet it is clear that they might not lead to optimal solutions. Consider the backward search where we start with the full set of features. Once the induction algorithm is able to filter some relation to the target, leaving out unnecessary features will generally lead to a good solution. However, when the number of features is large compared to the number of training instances, the induction algorithm might completely fail to do predictions on the full set and cannot measure the relevance of features.

The XOR problem (John et al., 1994; Guyon and Elisseeff, 2003) shows that a forward selection can fail when features are only jointly informative: Consider a binary function f with two relevant features,

$$f(1,1,\dots) = 0, \quad f(0,0,\dots) = 0, \quad f(0,1,\dots) = 1, \quad f(1,1,\dots) = 1\,. \tag{2.6}$$

The first two inputs become completely meaningless on their own if their outcomes 0 and 1 are equally probable, yet together they completely define f. Therefore, a forward selection which adds single informative features will not find evidence for the relevance of either feature.

Filters. Kohavi and John (1998) recommend using wrapper methods, as they directly optimize the predictive performance, which is usually what one is interested it. Also, the schemes can be wrapped around any induction algorithm used as a black box. However, in comparison to embedded methods, wrappers are computationally disadvantageous. A problem which can be even more severe is that wrappers tend to overfit the data when the test set is small:

filter methods

The tendency to overfit is analyzed in detail by Ng (1998), who considers two feature selection schemes. *Standard-wrap* is a wrapper which does an exhaustive search to solve (2.4), using the predictive error on a test set as score $\mathcal{S}(\boldsymbol{\sigma})$. Let the size of the holdout set be a fraction γ of N training samples. *Ordered-wrap* is a filter mechanism which ranks all features without using the test set. It constructs nested subsets $\boldsymbol{\sigma}^0, \boldsymbol{\sigma}^1 \dots \boldsymbol{\sigma}^D$, where $\boldsymbol{\sigma}^\ell$ contains all ℓ top-ranked features. Out of these D subsets ordered-wrap chooses the maximizer of the score $\mathcal{S}$. Ng shows that the generalization bound for standard-wrap depends on D via the term $\sqrt{D/(\gamma N)}$. Therefore, when we have four times as many features, we need twice as many training instances to expect the same performance on new data. Ordered-wrap can, in contrast, deal with exponentially many features, as the number of features enters the bound via $\sqrt{\log(D)/(\gamma N)}$: We need twice as many training instances only as the number of features is squared.

What distinguishes ordered-wrap from standard-wrap is that it searches only over D nested sets which are defined independently of the score. In accordance to Blum and Langley (1997) we use the term *filter method* for all related

schemes which include candidates into $\boldsymbol{\sigma}$ in the order of some ranking which depends on the training set only. The robustness of filter schemes makes them the method of choice when $D \gg N$, i.e. when we have many more features than samples. This can be the case for example in object recognition (Vidal-Naquet and Ullman, 2003)[7], in bioinformatics (Weston et al., 2003b)[8], and in text categorization (Yang and Pedersen, 1997)[9]—and the troubleshooting problem which we address here.

2.3. The troubleshooting approach

Ranking scores. In the preceding section we have argued that filter methods are preferable when the number of features is much larger than the number of samples. This situation is given in the troubleshooting setup, and therefore we have chosen a filter scheme in our analysis. A convenient side effect is that filter methods are computationally much cheaper than wrappers or embedded methods, as only a small number of nested subsets is considered.

According to the definition of Ng (1998), any ranking criterion is admissible as long as it only depends on the training set, and the inclusion of the ℓth feature may thus depend on the previously selected subset $\boldsymbol{\sigma}^{\ell-1}$. However, most ranking criteria are univariate, i.e. they only compare single features x_ℓ to the target y using a ranking score $R_i = R(x_i, y)$. Accordingly, the subset $\boldsymbol{\sigma}^\ell$ is chosen to contain the first ℓ features with best scores R_i. The number of univariate scores which can be found in literature is large, examples are the *correlation coefficients*, which measure the quality of a linear fit to the corresponding parameter, and the related *Fisher score* (Guyon and Elisseeff (2003); Bishop (1995)).

correlation coefficients
Fisher score

Mutual information. We have chosen the *mutual information* (MI) as a measure which is based on information theoretic principles. It is in general hard to calculate[10], but as we are dealing with binary features and targets it can be estimated from the available data with little computational effort. The mutual information is based on the concept of *entropy*, long known in statistical physics. It was introduced by Clausius as early as 1855 for thermodynamics, and reinterpreted as the degree of disorder by Boltzmann in 1877 (for details see e.g. (Feynman et al., 1963, ch. 44.6)). Shannon (1948) derived the entropy as a measure for the information content of a random process, using only few

mutual information
entropy

[7] Vidal-Naquet and Ullman search for image fragments which are representative for their class. They are dealing with 59 200 of features and use a training set of 934 images.

[8] Weston et al. predict the chemical activity of a drug based on 139 351 binary features which describe the geometry of the molecule. They have access to 1 909 samples, 42 of which being active.

[9] Yang and Pedersen present a comparative study of several filters and consider two datasets with 16 039/72 076 features and training sets of 9 610/1 990 instances. Text categorization is somewhat special, in that documents are assigned to a large number of different categories (92/12).

[10] When the features are continuous one needs to use nonparametric density estimation. See Bonnlander and Weigend (1994).

axiomatic properties. In the following paragraphs we give a short overview, restricting the presentation to the special case which is important for our feature selection setup. A comprehensive introduction to information theory can be found in (MacKay, 2003) or (Cover and Thomas, 1991).

ILLUSTRATION. Consider a Bernoulli $\mathcal{B}(q)$ distributed random variable X —i.e. $p(X=1)=q$ and $p(X=0)=1-q$. The entropy H is defined as

$$H(X) = -\sum_{x\in\{0,1\}} p(X=x)\log_2 p(X=x) \quad (2.7a)$$

$$= -\left[q\log_2 q + (1-q)\log_2(1-q)\right] . \quad (2.7b)$$

Note that we have chosen the logarithm to the base two to measure the entropy in *bits*, while in physics it is usually measured in terms of heat capacity, i.e. units of Joule per Kelvin by using the natural logarithm and the Boltzmann constant.

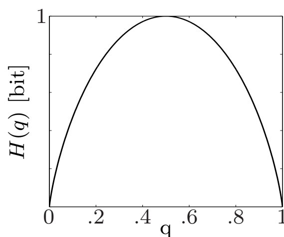


Figure V.5.: The entropy of a Bernoulli $\mathcal{B}(q)$ distribution.

See figure V.5 for a plot of $H(q)$ against q. For $q=0$ or 1 the outcome of X is precisely known and the outcome of a trial does not provide us with any new information. In the other extreme case $q=\frac{1}{2}$ we know nothing about the results and we need exactly one bit to store the outcome of each trial. Shannon's source code theorem (Shannon, 1948, theorem 4) guarantees that we can extend these results to all values of q: Sequences of N randomly chosen symbols can be stored using H bits per symbol in the limit of large N.

After defining the entropy as a measure for the information content of a random variable, we can define the mutual information between two random variables X and Y:

$$\mathrm{MI}(X,Y) = H(X) - H(X|Y) . \quad (2.8a)$$

The conditional entropy $H(X|Y)$ is the average entropy of the conditional distribution $p(X|Y=y)$ over Y:

$$H(X|Y) = -\sum_{y\in\{0,1\}} p(Y=y)\left[\sum_{x\in\{0,1\}} p(X=x|Y=y)\log_2 p(X=x|Y=y)\right] . \quad (2.8b)$$

Hence, the MI measures how much information the outcome of Y contains about X on average. Note that as the MI is symmetric, i.e. we obtain $\mathrm{MI}(X,Y)=\mathrm{MI}(Y,X)$.

In the feature selection setup we are given a target vector $\mathbf{y}$ and feature vectors $(x_{1\ell}, x_{2\ell}\ldots x_{N\ell})^T$, which contain the binary class label and feature ℓ for each sample. If we think of the entries as samples from a joint distribution $p(X^\ell, Y)$, we can approximate the ranking scores $\mathrm{MI}(X^\ell, Y)$ in (2.8) by

replacing the probability distributions by their empirical estimates. The computations are, in the end, done by counting the occurrence of the respective events. See appendix C.

Conditional mutual information. Univariate measures rank the features one-by-one, and therefore they cannot capture whether those are redundant or whether they are more informative when considered jointly (as in the XOR-problem (2.6)). However, simple rankings are usually good enough to identify candidates for a subset to be used in a more powerful classifier[11].

For our troubleshooting problem it is more important to understand the structure of the data than to correctly predict the targets, and a more detailed ranking can help to capture the underlying processes. The natural bivariate extension of the MI is the *conditional* mutual information (CMI), which measures how much information observations of a random variable X^i supply about a target variable Y when a third variable X^ℓ has already been observed. The conditional mutual information is defined as

conditional mutual information

$$\mathrm{CMI}(X^i, Y|X^\ell) = \mathrm{H}(X^i|X^\ell) - \mathrm{H}(X^i|YX^\ell) \;, \tag{2.9}$$

where the difference to (2.8) is simply that we condition all entropies on X^ℓ. It has been used in previous work by Vidal-Naquet and Ullman (2003) and Fleuret (2004) to discard redundant features in the selected subsets. We extend this approach for a detailed analysis of the data generating process.

The interpretation of the CMI is the following: When two features X^i and X^ℓ convey different information to Y, the CMI reduces to the $\mathrm{MI}(X^i, Y)$, while it is zero when they are perfectly correlated. In the XOR-problem (2.6), which is the other extreme case, X^i and X^ℓ might be completely uninformative (i.e. $\mathrm{MI}(X^{i,\ell}, Y) = 0$) when regarded independently—yet together they contain all information about Y and $\mathrm{CMI}(X^i, Y|X^\ell) = H(Y)$.

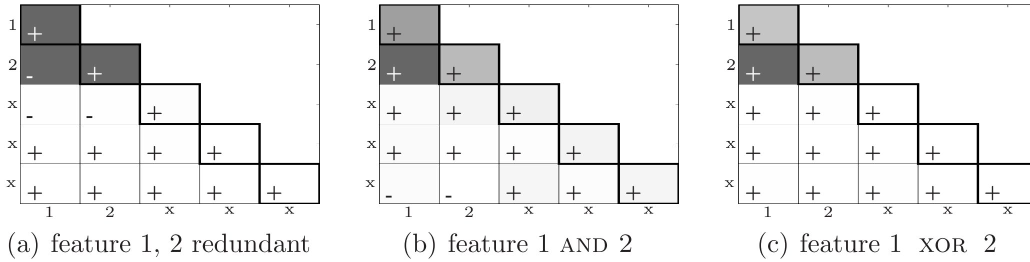
Consider the simple examples[12] in figure V.6, where we show the MI of each feature X^i and the target Y, comparing it to the $\mathrm{CMI}(X^i, Y|X^\ell)$ by plotting the differences

$$\Delta\mathrm{MI}(X^i, Y|X^\ell) = \mathrm{CMI}(X^i, Y|X^\ell) - \mathrm{MI}(X^i, Y) \;. \tag{2.10}$$

It can easily be seen, that ΔMI is symmetric in X^i and X^ℓ. Panel (a) shows a typical case where two features (1 and 2) are strongly correlated to the target, yet perfectly correlated to one another. We obtain large MIs and zero $\mathrm{CMI}(X^1, Y|X^2)$. The difference $\Delta\mathrm{MI}(X^1, Y|X^2)$ is therefore the negative MI. The cases (b) and (c) are contrary: as the target is given as $X^1 \wedge X^2$ and X^1 XOR X^2 respectively, the CMI is larger than the MI. The difference is

[11] Guyon and Elisseeff (2003) argue that redundant features can lead to a better predictive performance, as they help to suppress noise.

[12] We have randomly created the data for this example using 5 binary features and $N = 1000$ samples, choosing $p(x_\ell = 1) = 0.2\ \forall \ell = 1 \ldots 5$. We have set the targets to zero and changed them to one with $p(y = 1|\lambda(x_1, x_2) = 1) = 0.5$. Noise was added by setting $p(\text{swap target}) = 0.05$. The placeholder λ stands for the operators $\lambda(x_1, x_2) = x_1$ (a), $\lambda(x_1, x_2) = x_1 \wedge x_2$ (b) or $\lambda(x_1, x_2) = x_1$ XOR x_2 (c).



(a) feature 1, 2 redundant (b) feature 1 AND 2 (c) feature 1 XOR 2

Figure V.6.: Artificial toy example[12]. The plots show the conditional mutual information (2.9) for all combinations of features in the form of a (symmetric) matrix. The diagonals (ℓ, ℓ) show $\mathrm{MI}(X^\ell, Y)$, the off-diagonal elements (ℓ, i) give the differences $\Delta\mathrm{MI}(X^\ell, Y|X^i)$. Large absolute values are indicated by dark gray shades, the signs are explicitly given. In (a) features 1 and 2 are perfectly correlated and each defines Y. In the example (b) Y is given by $X^1 \wedge X^2$, and in (c) by X^1 XOR X^2.

positive as each feature gives us more information about the target if the other is known. Thus, from matrices as those shown in figure V.6, we can read the logical dependency between features, and analyze which features contribute to the target. Redundancies and joint contributions such as in the XOR or AND examples can be perceived at one glance.

The support vector machine. In the preceding section we have introduced the ranking criteria MI and CMI in detail, which are crucial to understand our approach to the troubleshooting problem. In the following we introduce the support vector machine (SVM), which we have used to validate the predictive power of the selected subsets of features. SVM are covered for example in the textbooks by Schölkopf and Smola (2002) and Cristianini and Shawe-Taylor (2000), for a tutorial see (Burges, 1998). Our implementation uses the LIBSVM-package by Chang and Lin (2001).

SVM

The *weighted soft-margin SVM* (referred to as C-SVM), which we have chosen for our problem, solves the following constrained optimization problem[13]:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \left[\tfrac{1}{2}\|\mathbf{w}\|^2 + C \left(C_1 \sum_{i;y_i=1} \xi_i \; + C_0 \sum_{i;y_i=0} \xi_i \right) \right] \tag{2.11a}$$

$$\text{s.t.} \quad (\mathbf{w}\boldsymbol{\phi}(\mathbf{x}_\ell) + b) \begin{cases} \leq \xi_\ell - 1 & \text{if} \quad y_\ell = 0 \\ \geq 1 - \xi_\ell & \text{if} \quad y_\ell = 1 \end{cases} \tag{2.11b}$$

$$\xi_\ell \geq 0 \tag{2.11c}$$

The objective function in (2.11a) consists of two terms. The norm $\|\mathbf{w}\|^2$ penalizes the complexity of the solution, while the second term penalizes wrong predictions. Note that the constants C_1 and C_0 weight the misclassification dependent of the classes, and can be adjusted to reflect the importance we ascribe to a correct classification of good or bad lots.

[13] We have used the same notation as Hsu et al. (2001). See equation (1) therein.

We are dealing with an unbalanced distribution of the classes $y = 0$ (good) and $y = 1$ (conspicuous), yet we are particularly interested in a correct prediction of the conspicuous lots. Therefore we set the weights such that the loss for misclassification is equal for both class:

$$C_1 = \frac{\#[\mathbf{y} = 0]}{\#[\mathbf{y}]} \qquad \text{and} \qquad C_0 = \frac{\#[\mathbf{y} = 1]}{\#[\mathbf{y}]} \,. \tag{2.12}$$

score

To be able to rate the predictive performance of the C-SVM we need a score function which measures how well the relation of inputs and targets has been understood. In agreement with the above loss function we do not rate classifications in both classes equally, as we believe that errors are only rarely found with absent root cause, while flawed machines will still produce good parts. Thus we use the balanced score function introduced by Weston et al. (2003b),

$$\mathcal{S}(\mathbf{y}_o, \mathbf{y}_*) = \frac{1}{2}\left[\frac{\#\{\mathbf{y}_o = \mathbf{y}_* = 1\}}{\#\{\mathbf{y}_o = 1\}} + \frac{\#\{\mathbf{y}_o = \mathbf{y}_* = 0\}}{\#\{\mathbf{y}_o = 0\}}\right] , \tag{2.13}$$

which is one when all lots are classified correctly, assigning an equal contribution to each class. We denote test samples by $\mathbf{y}_o$ and the corresponding predictions by $\mathbf{y}_*$.

The C-SVM (2.11) has a free parameter C, and the Gaussian kernel function $k(\mathbf{x}, \mathbf{x}') = \exp\{-\gamma|\mathbf{x} - \mathbf{x}'|^2\}$, which we have chosen for our implementation, requires another parameter γ to be set. As the SVM is not a Bayesian method, we cannot do a formal model selection and need to resort to another criterion. We have chosen a 10 fold cross validation (CV) to optimize C and γ, and 25 folds to rank subsets of features.

3. Case study

Above we have introduced the filter approach for feature selection, which we have chosen for our troubleshooting scheme. The mutual information (2.8), the conditional mutual information (2.9) and the predictive performance of the C-SVM classifier $\mathcal{S}$ (2.8) play a crucial role in the analysis. In this section we present the results of a case study, which has been performed before transferring our troubleshooting scheme to the Bosch semiconductor foundry, where it is now regularly used. The purpose of the presentation on the following pages is, on the one hand, to demonstrate the good performance on examples from mass production, and, on the other hand, to serve as an instruction for the practioner on how to interpret its output.

3.1. Datasets

The datasets which are analyzed in this case study have been extracted from the database in the foundry. The data stem from the mass production of different products, representing recent and historic problems. We have collected

Name	number of stages	number of machines	number of features	number of lots tot.	number of samples class 1
PC1	403	357	1896	112	41
PC2	355	331	1758	98	28
YC1	157	142	779	870	11
YC2	339	391	2104	261	37

Table V.1.: Benchmark datasets from the Bosch semiconductor foundry.

the details of the data in table V.1. For two of the datasets, PC1 and PC2[14], we can expect to obtain clear results as they come with a relatively large number of samples in class one—41 flawed lots have been detected in PC1, 28 in PC2. Also, the lots have been classified according to the occurrence of patterns, which, as we have argued in section 1.1, are known to be excellent indicators for systematic errors. In the datasets YC1 and YC2[14] we have defined the lots to be conspicuous when the total number of defects exceeded a threshold. In contrast to the patterns, this criterion is not very sharp as it mixed several root causes and random failures. In YC1 we are given as many as 870 lots in total, but with only 11 observed errors the root cause will be hard to locate. YC2 relies again on more (37) samples.

3.2. Interpretation of the results

PC2. For the case PC2 we extracted data from a time interval, in which characteristic patterns were observed on some wafermaps. The root cause had not yet been understood. The results of our analysis, displayed in figure V.7, lead to the identification of the flaw, and the maintenance team could be pointed to the responsible machine.

Using the mutual information we have ranked all 1758 features from the lot history. See plot V.7(a). Most features are ranked below a noise level of roughly 0.05 bit, and one feature is found to be highly informative. Its ranking score is around $\frac{1}{2}$ bit. Two other features apparently carry some information about the target, obtaining about half of the first feature's score. The ΔMI matrix in plot V.7(b) displays the results of the ranking in more detail. Note that we plot all values normalized with respect to the maximal element in the matrix. The features have been sorted according to the MI (shown on the diagonal), and their interdependencies ΔMI are plotted below the diagonal. Note that we have discarded all but the first 10 features for a clearer illustration. The absolute values of the ΔMI are shown in gray shades, and their signs are given within the matrix elements. The top-ranked feature, f.288, corresponds to the first column of the matrix.

We find that feature f.289 is uninformative once we know f.288: As we condition on f.288 it does not provide us with much new information and

[14]Our abbreviations stand for (P)attern based (C)lassification and (Y)ield based (C)lassification.

$\Delta\text{MI}(X^{289}, Y|X^{288}) \approx -\text{MI}(X^{289}, Y)$). The same holds for f.28a. Note, however, that both features apparently complement each other, and f.28a is more informative as we condition on f.289 or vice versa. The structure our ranking method has uncovered here reflects the production plan for the considered device. The machines behind all three features belong to a single production stage, where the one corresponding to f.288 was related to the error. Feature f.289 and f.28a carry some information about the occurrence of the error, as the lots pass only one of the machine in the stage. As we are given both features, however, we know exactly whether it has passed machine 288 and is thus likely to be affected by the error.

Plot V.7(c) shows the predictive performance of the SVM classifier with error bars obtained through cross validation. Using only the most informative feature f.288 we obtain an extremely good performance of $\mathcal{S} \approx 80\%$, which substantiates its strong ranking of $\frac{1}{2}$ bit. As we add more features the score does not increase, confirming the dependencies predicted in the ΔMI matrix. As we add the $9^{\text{th}}$and 10^{th}feature the predictive performance decreases: the information content on the active set can certainly not decrease as we add more features, but as we pointed out in section 2, the classifier might degrade as we add many irrelevant features.

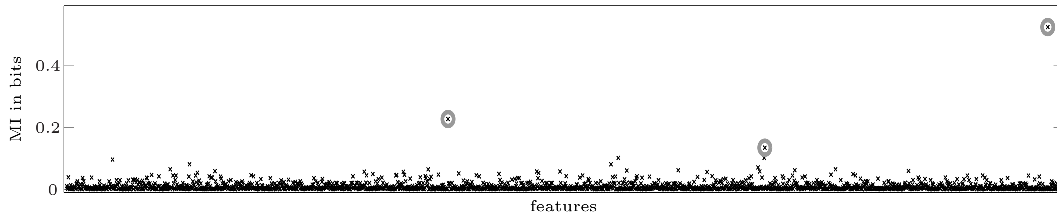
PC1. Dataset PC1 is again an example for a pattern based error detection, where we are given a good number of examples for flawed lots. The data represents a historic problem in the plant, which had already been solved manually—thus requiring a lot of manpower. Using our method we could correctly reproduce the known results, this time, however, in an automated manner. The output of our method is shown in figure V.8.

A look at plot V.8(a) shows that three features are ranked higher than the great mass, obtaining relatively high scores between 0.13 and 0.25 bit. In V.8(b) the ranking score is broken down to dependencies between the 10 top ranked features. The top features f.675 and f.404 are apparently highly correlated, the third feature f.3cf, however, brings new information which is not related to the first two. The same holds for the fourth feature f.626.

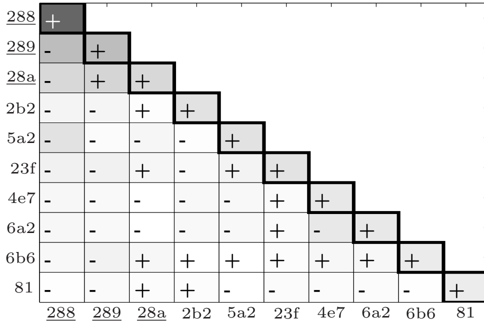
The classification in V.8(c) supports our analysis for the top three features. We obtain a score of 70% for the first or first two features, yet adding f.3cf increases the performance by 11%. Feature 626, in contrast, has not been validated to be informative and we can conjecture that a combination of either f.675 and f.3cf or f.404 and f.3cf was responsible for the error.

YC1. Dataset YC1 is another historic case, where an unusually high defect rate was observed from time to time. It consists of a large number of sample lots, the number of affected batches, however, is very small (11 out of 870). Still, our method successfully uncovered the root cause. Find the results in figure V.10.

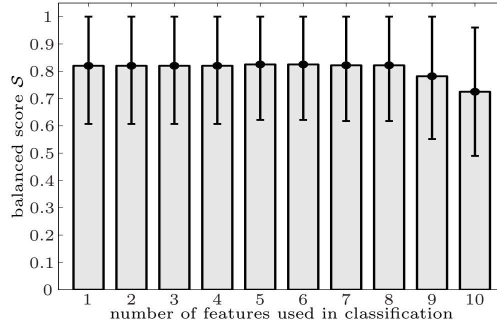
One of the 779 features (f.2a5) in YC1 sticks out of the noise with a ranking score of 0.025 bit (plot V.10(a)). The score is, compared to the other examples, extremely low, but as the entropy of the target is itself very low



(a) MI ranking score for all features.

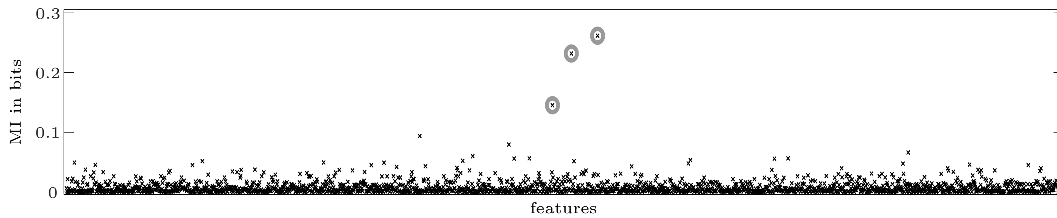


(b) ΔMI matrix with MI on the diagonal.

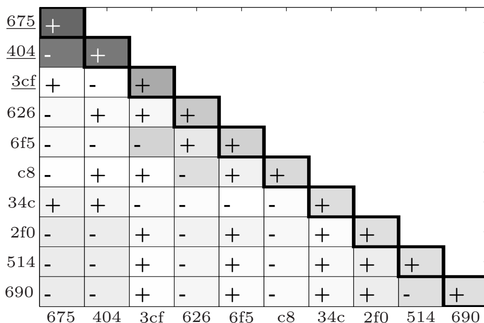


(c) Predictive performance of the SVM.

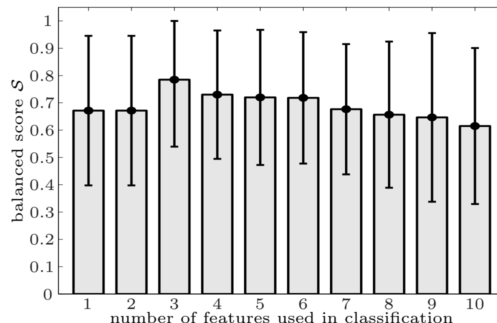
Figure V.7.: Dataset PC2: We find in plot (a) that f.288 obtains a much higher ranking score than all other features. In plot (b) we show the 10 top-ranked features and use the ΔMI to analyze their dependencies. Note that the following f.289 and f.28a completely depend on f.288, indicated by large negative $\Delta\text{MI}(\cdot, Y|X^{288})$s. The good ranking for f.288 ($\frac{1}{2}$ bit) is reflected by a large prediction score (c) of over 80%.



(a) MI ranking score for all features.



(b) ΔMI matrix with MIs on the diagonal.



(c) Predictive performance of the SVM.

Figure V.8.: Dataset PC1: The MI ranking coefficients for all features are shown in (a), where three features stand out. The ΔMI, shown in (b), shows that f.675 and f.404 are highly dependent, while f.3cf gives additional information. This is substantiated in the classification (c), where the score is better when f.3cf is added. The high rankings ($\sim$ 0.2 bit each) correspond a predictive performance of ($\sim$ 80%).

(H(Y) = 0.098 bit), this corresponds to an information content of 25%.

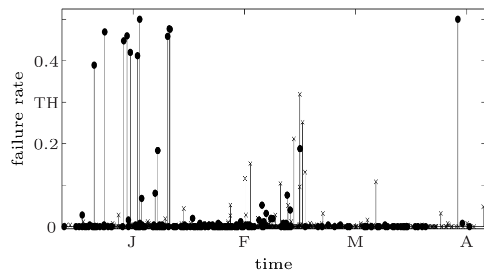


Figure V.9.: Failure rates for all lots in YC1 against the production date[15]. Lots which have passed f.2e5 have been marked by •s. TH marks the threshold.

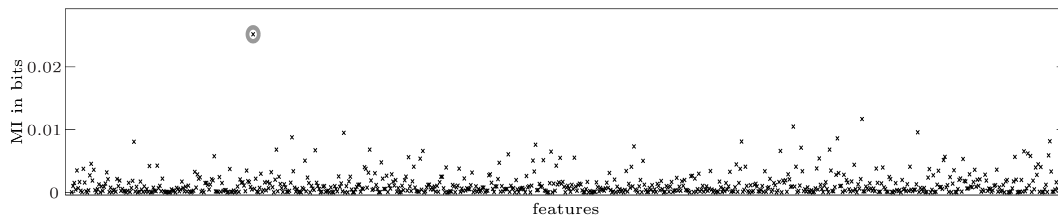
As only one feature appears to be informative, the ΔMI matrix shown in plot V.10(b) cannot provide us with much new insight. The resulting classifier is, with a score of slightly less than 60%, not much better than a random guess (plot V.10(c)). A closer look at the data, however, shows that the predictions can in fact not be much better. All suspicious lots have actually passed the machine referred to by f.2a5, but it still produced faults in less than 5% of the cases. We see in figure V.9[15], that the machine produced an increased failure rate only in a restricted time window—with the classifier's predictions naturally being wrong in the meanwhile.

YC2. The results for the dataset YC2, shown in figure V.11, is based on a problem from the foundry where we look for the root cause of an increased failure rate. As it is mostly the case for error detection based on failure rates, it is hard to separate different causes and an appropriate threshold is not easily identified. Our troubleshooting method does not point to a root cause here, but note that we are not lead onto a wrong trace—the analysis shows that no conclusions can be drawn from this dataset.

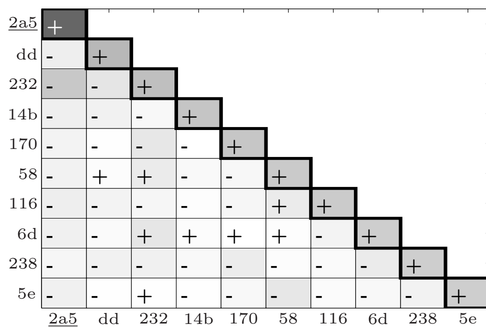
Though the ranking scores, shown in V.11(a), are relatively high, no feature is ranked significantly above the noise level. Accordingly, we cannot find any interesting dependencies in the ΔMI matrix (plot V.11(b)). The predictive performance, given in V.11(c), shows scores around 60%, thus only slightly above random. As we are given a large fraction of affected lots this indicates that there is really no connection between the lot-history and the target present in the data.

[15] For reasons of nondisclosure we have normalized failure rates and dates.

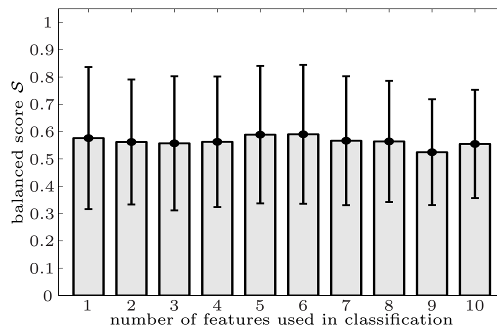
replacemen



(a) MI ranking score for all features.

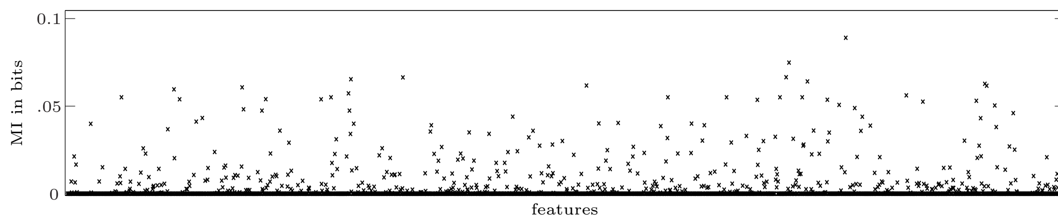


(b) ΔMI matrix with MI on the diagonal.

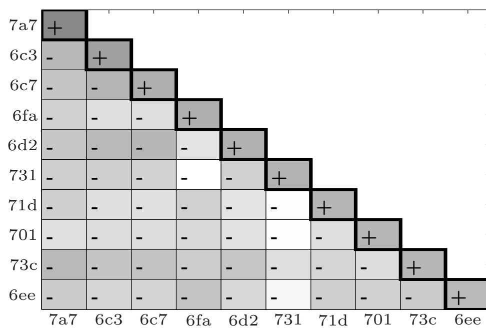


(c) Predictive performance of the SVM.

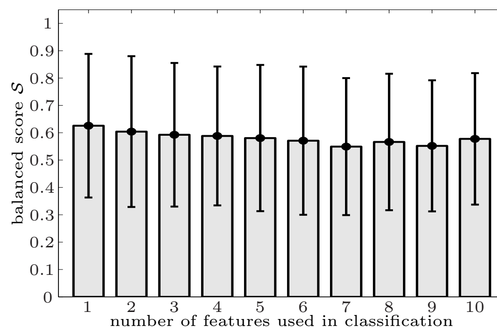
Figure V.10.: Dataset YC1: Panel (a) shows that the ranking for feature 2a5 is well above the noise level. However, the low score of 0.025 bits indicates that the relation to the target is weak. With roughly 60% the predictive performance (c) is only slightly above random. Panel (b) shows that f.dd gives some additional information about the target, while f.232 is completely dependent on f.2a5.



(a) MI ranking score for all features.



(b) ΔMI matrix with MI on the diagonal.



(c) Predictive performance of the SVM.

Figure V.11.: Dataset YC2: The ranking (a) does not uncover any prominent feature, and the ΔMI matrix in (b) shows that the top-ranked features are strongly correlated. The predictive performance of the SVM (c) is accordingly weak.

4. Discussion

In this section we have discussed a novel mechanism for troubleshooting in complex manufacturing lines. The work was initiated to improve quality assurance and process control in a semiconductor foundry, where it is impossible to control all device specifications continuously. However, as we do not rely on process specific models our approach holds for any serial-group manufacturing line. The method is based on the connection of two different types of data which are available in the plant's database: the results of final measurements in quality control, and the lot-history which tracks the batches' passage through the manufacturing line.

The idea of our approach is to combine both types of information to create an automatic tool, which enables the process control team to rapidly identify root causes for systematic defects. An error which can be attributed to few machines is usually easily fixed, while it is nearly impossible to inspect all machines in the cleanroom. When a great number of affected batches is available in the database the troubleshooting problem is easily solved, and it is the limited amount of data which makes the task a formidable problem: as the aim is to eliminate an error as fast as possible, typically we only have tens of positive examples, while there are thousands of possible root causes.

We have applied feature selection methods, which have recently received a lot of attention in the machine learning community, to solve this task. The filter approach, which we have chosen here, is particularly efficient for problems like the troubleshooting task where the number of samples is much smaller than the number of features. The ranking scores—the entropy-based mutual information and conditional mutual information—which we have used, are theoretically justified and enable us to analyze the data's structure in detail. To rule out overfitting we apply an extra validation step, where we determine the predictive performance of an SVM.

Before transferring our method as a tool to the foundry, we conducted a series of experiments to demonstrate its capability. The benchmark test consisted of four datasets from mass production, two of them representing historic cases and two being based on recent problems. We were able to confirm both historic problems, and to solve one of the recent problems. No conclusion could be drawn from the fourth dataset, but as we can avoid overfitting, our method does not point to wrong traces and states that no conclusion can be drawn.

The aim of the project was to construct a tool for the production facility, and thus it had to be designed to be used by non-specialists. Our implementation does not require the user to set any parameters, and the results can be read conveniently from few plots. The tool is now in regular use, and was confirmed to lead to a facilitation of the regular work and an accelerated error correction.

Conclusions

This thesis has examined mass production and industrial engineering as a challenging new field for the application of machine learning. Identifying applications for machine learning included collaboration with departments over the whole range from research and development to production and quality assurance at Robert Bosch GmbH—making this project markedly versatile. Particularly bringing together expertise from the applicational side at Bosch and the methodological side at the Max Planck Institute for Biological Cybernetics in Tübingen has been essential for this project to lead to useful and actually *used* innovation.

The conventional approach to data analysis in production and engineering is to build a model based on the underlying physical reality of the examined system. Therefore, the approach is of limited practicability for complex systems, and new concepts need to embrace methods which remain feasible when it is no longer practicable to describe devices and production facilities by physical models. Machine learning has been developed as an alternative to this deductive approach: using flexible statistical models of the interrelations' structure, the connection between the inputs and the answer of the system is inferred empirically from observations. The aim of this thesis was to assess where machine learning is a valuable complement of conventional data analysis in product design and mass production.

The survey over the state-of-the-art in data analysis in mass production showed that machine learning is now only used in few and very specific applications like automated optical tests. However, since—especially in semiconductor manufacturing—many data are already automatically stored in centralized data bases, the prerequisite for implementing machine learning solutions is usually given. These approaches can, on the one hand, automate tasks which are today left to a manual handling, and thus increase cost efficiency. On the other hand, some tasks require the joint analysis of thousands of variables, and only become feasible using machine learning methods, rendering possible completely new analyses.

Two identified problems have been addressed explicitly in this thesis, where the common thread is the yield as a central benchmark figure for the cost-effectiveness of a semiconductor foundry:

Starting at the development stage, product designs have to be made robust against **process tolerances** to reduce the number of random failures. This thesis introduces statistically justified sensitivity measures to facilitate the analysis and interpretation of high dimensional computer simulations. The machine learning approach to sensitivity analysis has lead to a novel optimiza-

tion scheme, which permits efficient robust optimization on computationally demanding simulations.
Malfunctioning processes may lead to **systematic failures**. Modern manufacturing lines are often extremely complex, thus frustrating a manual localization of root causes. A novel approach to troubleshooting is described, which combines error detection in quality control with other available data in the shop floor. The approach allows to locate the root cause in a large number of potential causes, using only few observations and thus cutting down on systematic losses by accelerating the elimination of the error.

Design alanysis. Numerical simulations are now extensively used in industrial engineering to validate the fitness of a design for mass production. Using such simulations, the system's response can be computed for various parameter settings, however, they can usually not facilitate an intuitive understanding of the system. To help the designer grasp the basic relations encoded in the model, this thesis proposes a procedure to analyze the response of the system to process fluctuations. Using statistically justified measures, the interpretation is eased by quantifying the importance of single parameters, the degree of nonlinearity and the strength of the entanglement between pairs of parameters.

In recent years such sensitivity analysis has received a lot of attention, however, to our knowledge this work has been the first to adapt it for industrial mass production—and actually making it available to practioners by providing a software package.

For computationally expensive simulations, the bottleneck of sensitivity analysis is the number of required simulation runs. Using a Bayesian approach based on nonparametric Gaussian process regression, the presented sensitivity analysis makes very efficient use of the available results, thus reducing the expense of the design analysis. Furthermore, the Gaussian process might be used as an efficient emulator of the computer model to let the designer interactively explore the model.

An extensive empirical study, using analytical benchmark problems and a number of simulators of micro electro-mechanical devices (MEMS) in development at Bosch, showed that the proposed analysis significantly outperforms conventional approaches: The Bayesian approach using Gaussian processes outperformed the Monte Carlo method, which can be considered state-of-the-art, on all eleven considered problems. On the MEMS models, which can be considered typical for industrial engineering tasks, the computational load due to simulation runs could be reduced roughly by an order of magnitude.

Design optimization. Process tolerances, which may have a significant effect especially in semiconductor manufacturing, should be accounted for in any design optimization. Available optimization schemes fail to account for the distribution of the fluctuations, or are infeasible due to a large number of required simulation runs.

In this thesis a novel approach to robust optimization has been developed, which combines a Gaussian process emulator with the distribution of the pro-

cess fluctuations in closed form. Thus, the optimization can be done efficiently using a gradient-based optimization scheme. To our knowledge, the presented scheme is the first computationally feasible approach to robust optimization, which quantitatively accounts for process tolerances.

Active learning. Active learning has been studied extensively in the last years and has been considered in this thesis to reduce the number of computer experiments, necessary to build a Gaussian process emulator. Many works in active learning are based on heuristic considerations. In contrast, probabilistic models such as Gaussian processes can be used to directly construct experimental designs, maximizing the utility of simulation runs. However, the derivation of a suitable utility function is intricate, and most previous works rely on expensive numerical approximations. This thesis is the first to derive the loss which corresponds to the generalization error—on uniform (regression setup) and Gaussian (sensitivity analysis) input distributions—in closed form.

In an extensive empirical study it could be shown that the proposed active learning scheme significantly improves the learning rate, including the case of noisy observations. Since it does not rely on heuristic considerations, it does not lead to uninformative designs, even in cases where not enough data is available to capture the underlying function. For seven analytical benchmark functions the proposed scheme increases the accuracy by a factor between one and 25, depending on the complexity on the mapping. On the MEMS models the required number of simulation runs to obtain a given accuracy could be reduced by a factor between two and five.

Troubleshooting. The presented troubleshooting approach combines error detection in quality checks with data which is collected during production. It has been verified in regular use to save time in production by replacing manual analysis, and to reduce costs by accelerating repairs. Using feature selection, the troubleshooting scheme combines data from various sources according to a standardized preprocessing, and is therefore applicable to most modern fabrications.

The empirical verification of the approach comprised four examples from the data base in Bosch's semiconductor foundry, which were used to clear the software tool for regular use. Carefully avoiding overfitting, the approach is not prone to false alarms, and could be shown to filter the root cause from 779 features using as few as 11 observations.

Outlook. By virtue of flexible statistical modeling, machine learning has the advantage that approaches and models are not necessarily specific to a certain physical relationship, but may instead be valid for a whole class of problems. Machine learning concepts can therefore, once established in engineering and production, accelerate the solution of new problems, and make completely new approaches possible where it is impractical to use physical models—giving machine learning the potential for great impact and value in this field. It is

to hope, that data analysis in industrial production becomes better known in the machine learning community as a challenging new field for research.

Appendix

A. Mean square differentiability

The properties of random processes are intimately related to the properties of the corresponding covariance function. Especially the smoothness of a random process is directly given by the smoothness of the covariance. A detailed discussion is given by (Stein, 1999), who focuses on the power spectrum of stationary and isotropic processes.

In this section we reproduce the basic results regarding the connection between the continuity and differentiability of a random process and the covariance function. The results have been taken from (Abrahamsen, 1997).

For random processes continuity and differentiability can be defined in several ways. One possibility is to define:

Definition 1 (Mean square (MS) continuity and differentiability)
Consider a random process X in $B \subset \mathbb{R}^D$.

- *X is* MS continuous*, if*

$$\mathrm{E}\big[|X(\mathbf{x}_n) - X(\mathbf{x})|^2\big] \to 0 \qquad as \quad n \to \infty$$

  $\forall \mathbf{x} \in B$ *and for all sequences* $\{\mathbf{x}_n\} \subset B$ *with* $\|\mathbf{x}_n - \mathbf{x}\| \to 0$ *for* $n \to \infty$*.*

- *X is* MS differentiable*, if* $\forall \ell = 1 \ldots D$

$$\mathrm{E}\left[\left|\tfrac{\partial}{\partial t_\ell} X(\mathbf{x}_n) - \tfrac{\partial}{\partial t_\ell} X(\mathbf{x})\right|^2\right] \to 0 \qquad as \quad n \to \infty$$

  $\forall \mathbf{x} \in B$ *and for all sequences* $\{\mathbf{x}_n\} \subset B$ *with* $\|\mathbf{x}_n - \mathbf{x}\| \to 0$ *for* $n \to \infty$*.*

The following theorem shows how the mean square properties relate to the covariance function:

Theorem 2 (Random process and covariance function)
Consider a random process X in $B \subset \mathbb{R}^D$ with covariance function $k(\mathbf{x}, \bar{\mathbf{x}})$. For simplicity assume the mean function $\mu(\mathbf{x})$ is zero.

- *X is MS continuous at $\tilde{\mathbf{x}}$*
  if and only if $k(\mathbf{x}, \bar{\mathbf{x}})$ is continuous at $\mathbf{x} = \bar{\mathbf{x}} = \tilde{\mathbf{x}}$.

- *If the derivative*

$$\frac{\partial^{2|\boldsymbol{\kappa}|}\, k(\mathbf{x}, \bar{\mathbf{x}})}{\partial x_1^{\kappa_1} \cdots \partial x_D^{\kappa_D}\, \partial \bar{x}_1^{\kappa_1} \cdots \partial \bar{x}_D^{\kappa_D}}$$

exists and is finite at $\mathbf{x} = \bar{\mathbf{x}} = \tilde{\mathbf{x}}$*, then* X *is* $|\boldsymbol{\kappa}|$ *times MS differentiable in* $\tilde{\mathbf{x}}$*, and the above expression is the covariance of*

$$\frac{\partial^{|\boldsymbol{\kappa}|} X(\tilde{\mathbf{x}})}{\partial x_1^{\kappa_1} \cdots \partial x_D^{\kappa_D}} \quad .$$

B. Bayesian Monte Carlo

Latin Hypercube designs

Latin Hypercube

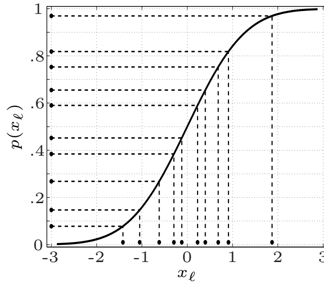


Figure V.12.: Latin Hypercube design for $\mathcal{N}(x|0,1)$.

The *Latin Hypercube* design is a popular and simple scheme for factorizing input distributions $p(\mathbf{x}) = \prod_\ell p_\ell(x_\ell)$: Figure V.12 depicts the sampling scheme for the normal distribution. To obtain N samples, for each dimension ℓ the interval $[0,1]$ is divided into N equal bins (grid of the y-axis) and a sample is drawn uniformly from each of them (dots to the left). Using the inverse CDF (solid line) of the parameter's distribution $p(x_\ell)$, corresponding samples x_ℓ are computed (dots on the x-axis). The samples for the single parameters are combined in random order to create a design which is stratified in one dimensional projections. The method has been proposed by MacKay et al. (1979), and extended to stratified designs in more dimensions by Tang (1993) and Ye (1998).

Moments of the uncertainty distribution

In the following section we specify the estimates for the mean and variance of the uncertainty distribution $p_{\mathbf{x}}(f)$ when a Gaussian process prior is used (see chapter III, section 3.2).

The estimate for the mean is simply the average over the predictive mean, as the expectations $\mathrm{E}_{f|\mathcal{D}}$ and $\mathrm{E}_{\mathbf{x}}$ can be swapped:

$$\begin{aligned} \mathrm{E}_{f|\mathcal{D}}\Big[\mathrm{E}_{\mathbf{x}}[f]\Big] &= \mathrm{E}_{\mathbf{x}}\Big[m(\mathbf{x})\Big] = \int \mathrm{d}\mathbf{x}\; p(\mathbf{x})\; m(\mathbf{x}) \\ &= \int \mathrm{d}\mathbf{x}\; p(\mathbf{x})\; \mathbf{k}(\mathbf{x})^T Q^{-1}\mathbf{y} = \mathbf{z}^T Q^{-1}\mathbf{y}\;, \end{aligned} \tag{B.14a}$$

where we have used the definition of the mean $m(\mathbf{x})$ from (II 3.17b) and the abbreviation $\mathbf{z}$ from (III 3.16). The estimate of the variance relies as well on the predictive uncertainty, and we need to decompose it into three terms (see also Oakley and O'Hagan (2004)):

$$\begin{aligned} \mathrm{E}_{f|\mathcal{D}}\Big[\mathrm{var}_{\mathbf{x}}[f]\Big] &= \mathrm{var}_{\mathbf{x}}\Big[\mathrm{E}_{f|\mathcal{D}}[f]\Big] + \\ &\quad + \mathrm{E}_{\mathbf{x}}\Big[\mathrm{var}_{f|\mathcal{D}}[f]\Big] - \mathrm{var}_{f|\mathcal{D}}\Big[\mathrm{E}_{\mathbf{x}}[f]\Big]\;. \end{aligned} \tag{B.14b}$$

If the GP model perfectly fits the function f with zero predictive variance, the sum reduces to the variance over the predictive mean:

$$\begin{aligned}\mathrm{var}_{\mathbf{x}}\Big[\mathrm{E}_{f|\mathcal{D}}[f]\Big] &= \mathrm{var}_{\mathbf{x}}\Big[m(\mathbf{x})\Big] \\ &= \int \mathrm{d}\mathbf{x}\, p(\mathbf{x}) \,\left(\mathbf{k}(\mathbf{x})Q^{-1}\mathbf{y}\right)^2 - \mathrm{E}_{\mathbf{x}}[m(\mathbf{x})]^2 \\ &= \mathrm{trace}\left[(Q^{-1}\mathbf{y})(Q^{-1}\mathbf{y})^T L\right] - \mathrm{E}_{\mathbf{x}}[m(\mathbf{x})]^2\ ,\end{aligned} \tag{B.14c}$$

where L is defined by (III 3.16). Due to finite predictive uncertainty we obtain the following two contributions,

$$\begin{aligned}\mathrm{E}_{\mathbf{x}}\Big[\mathrm{var}_{f|\mathcal{D}}[f]\Big] &= \mathrm{E}_{\mathbf{x}}\Big[\sigma^2(\mathbf{x})\Big] \\ = \int \mathrm{d}\mathbf{x}\, p(\mathbf{x})\, \Big[k(\mathbf{x},\mathbf{x}) - \mathbf{k}(\mathbf{x})^T Q^{-1}\mathbf{k}(\mathbf{x})\Big] &= k_o \ - \ \mathrm{trace}[Q^{-1}L]]\end{aligned} \tag{B.14d}$$

and

$$\begin{aligned}\mathrm{var}_{f|\mathcal{D}}\Big[\mathrm{E}_{\mathbf{x}}[f]\Big] &= \mathrm{E}_{f|\mathcal{D}}\Big[\big(\mathrm{E}_{\mathbf{x}}[f] - \mathrm{E}_{f|\mathcal{D}}\left[\mathrm{E}_{\mathbf{x}}[f]\right]\big)^2\Big] \\ &= \int \mathrm{d}\mathbf{x}\, p(\mathbf{x}) \int \mathrm{d}\mathbf{x}'\, p(\mathbf{x}')\, \mathrm{E}_{f|\mathcal{D}}\Big[\,\big(f(\mathbf{x}) - \mathrm{E}_{f|\mathcal{D}}\left[f(\mathbf{x})\right]\big) \\ &\qquad\qquad \times\ \big(f(\mathbf{x}') - \mathrm{E}_{f|\mathcal{D}}[f(\mathbf{x}')]\big)\,\Big] \\ &= \int \mathrm{d}\mathbf{x}\, p(\mathbf{x}) \int \mathrm{d}\mathbf{x}'\, p(\mathbf{x}')\, \mathrm{cov}_{f|\mathcal{D}}\left[f(\mathbf{x}), f(\mathbf{x}')\right] \\ &= \int \mathrm{d}\mathbf{x}\, p(\mathbf{x}) \int \mathrm{d}\mathbf{x}'\, p(\mathbf{x}')\ \left[k(\mathbf{x},\mathbf{x}') - \mathbf{k}(\mathbf{x})^T Q^{-1}\mathbf{k}(\mathbf{x}')\right] \\ &= k_c\ -\ \mathbf{z}^T Q^{-1}\mathbf{z}\ ,\end{aligned} \tag{B.14e}$$

where k_o and k_c are defined in (III 3.16). Note that the last term is identical to the predictive uncertainty for the mean-estimate.

Exact average for specific input distributions

When the common squared exponential covariance function (II 3.20) is used, the integrals (III 3.16) for the moments of the uncertainty distribution (chapter III, section 3.2) can be solved in closed form for several input distributions $p(\mathbf{x})$. Among those are the uniform and the Gaussian input distribution. When the input distribution factorizes, the averages can be reduced to one dimensional integrals which can in any case readily be solved numerically.

Gaussian input distribution. For the Gaussian input distribution (III 3.17), $p(\mathbf{x}) = \mathcal{N}(\hat{\mathbf{x}}, B)$, we obtain

$$\begin{aligned}
k_o &= \int \mathrm{d}\mathbf{x}\, p(\mathbf{x})\; k(\mathbf{x},\mathbf{x}) = v^2 && \text{(B.15a)}\\
k_c &= \int \mathrm{d}\mathbf{x}\, p(\mathbf{x}) \int \mathrm{d}\mathbf{x}'\, p(\mathbf{x}')\; k(\mathbf{x},\mathbf{x}') && \text{(B.15b)}\\
&= w_o^2\, |2A^{-1}B + I|^{-\frac{1}{2}} \\
z_\ell &= \int \mathrm{d}\mathbf{x}\, p(\mathbf{x})\; k(\mathbf{x},\mathbf{x}_\ell) && \text{(B.15c)}\\
&= w_o^2 (2\pi)^{\frac{d}{2}} |A|^{\frac{1}{2}} \int \mathrm{d}\mathbf{x}\, \mathcal{N}(\mathbf{x}|\bar{\mathbf{x}}, B) \mathcal{N}(\mathbf{x}|\mathbf{x}_\ell, A)\\
&= w_o^2 |A^{-1}B + I|^{-\frac{1}{2}}\\
&\quad \times e^{-\frac{1}{2}\left[(\mathbf{x}_\ell - \bar{\mathbf{x}})^T (A+B)^{-1} (\mathbf{x}_\ell - \bar{\mathbf{x}})\right]}\\
\ell(\mathbf{x},\mathbf{x}') &= \int \mathrm{d}\tilde{\mathbf{x}}\, p(\tilde{\mathbf{x}})\; k(\tilde{\mathbf{x}},\mathbf{x}) k(\tilde{\mathbf{x}},\mathbf{x}') && \text{(B.15d)}\\
&= w_o^4 |2A^{-1}B + I|^{-\frac{1}{2}}\\
&\quad \times e^{-\frac{1}{2}\left[(\mathbf{x}-\mathbf{x}')^T \frac{1}{2} A^{-1} (\mathbf{x}-\mathbf{x}')\right]}\; e^{-\frac{1}{2}\left[(\bar{\mathbf{x}}-\hat{\mathbf{x}})^T \left[\frac{1}{2}A + B\right]^{-1} (\bar{\mathbf{x}}-\hat{\mathbf{x}})\right]}\\
\text{with} \quad \bar{\mathbf{x}} &= \tfrac{1}{2}\left(\mathbf{x} + \mathbf{x}'\right),
\end{aligned}$$

where we have defined $A = \mathrm{diag}\{w_1^2 \dots w_D^2\}$ as in (II 3.18).

Factorizing input distributions. In the special—yet important—case of factorizing input distributions

$$p(\mathbf{x}) = \prod_d p_d(x_d) \qquad \text{(B.16)}$$

and factorizing covariance functions like the squared exponential kernel (II 3.20)

$$k_{\mathrm{SE}}(d_{\mathrm{ARD}}) = v^2 \exp\left[-\tfrac{1}{2} d_{\mathrm{ARD}}^2\right] = v^2 \prod_{d=1}^{D} \exp\left[-\tfrac{(x_d - x_d')^2}{2w_\ell^2}\right], \qquad \text{(B.17)}$$

the multidimensional averages can be reduced to products of one dimensional integrals, which are in general much easier to solve:

$$\begin{aligned}
k_c &= \int \mathrm{d}\mathbf{x}\, p(\mathbf{x}) \int \mathrm{d}\mathbf{x}'\, p(\mathbf{x}')\; k_{\mathrm{SE}}(\mathbf{x},\mathbf{x}') && \text{(B.18a)}\\
&= v^2 \prod_{d=1}^{D} \int \mathrm{d}x_d\, p(x_d) \int \mathrm{d}x_d'\, p(x_d')\; \exp\left[-\tfrac{(x_d - x_d')^2}{2w_d^2}\right]\\
k_o &= \int \mathrm{d}\mathbf{x}\, p(\mathbf{x})\; k_{\mathrm{SE}}(\mathbf{x},\mathbf{x}) = v^2 && \text{(B.18b)}\\
z_\ell &= \int \mathrm{d}\mathbf{x}\, p(\mathbf{x})\; k_{\mathrm{SE}}(\mathbf{x},\mathbf{x}_\ell) && \text{(B.18c)}\\
&= v^2 \prod_{d=1}^{D} \int \mathrm{d}x_d\, p(x_d) \exp\left[-\tfrac{(x_d - x_{\ell d})^2}{2w_d^2}\right]
\end{aligned}$$

$$\begin{aligned}\ell(\mathbf{x},\mathbf{x}') &= \int \mathrm{d}\tilde{\mathbf{x}}\; p(\tilde{\mathbf{x}})\; k_{\mathrm{SE}}(\tilde{\mathbf{x}},\mathbf{x}) k_{\mathrm{SE}}(\tilde{\mathbf{x}},\mathbf{x}') \\ &= v^4 \prod_{d=1}^{D} \int \mathrm{d}\tilde{x}_d\; p(\tilde{x}_d) \exp\left[-\tfrac{(\tilde{x}_d - x_d)^2}{2w_d^2}\right] \exp\left[-\tfrac{(\tilde{x}_d - x'_d)^2}{2w_d^2}\right] .\end{aligned} \tag{B.18d}$$

If no analytical solution can be found, we can apply e.g. the efficient one dimensional Gauss-Hermite quadrature, as the covariance function factorizes into one dimensional Gaussian distributions.

Uniform input distribution. For uniform input distributions $p_d(x_d) = \mathcal{U}(a_d, b_d)$ the one dimensional integrals can be solved in closed form, since all reduce to integrating a normal distribution between the limits a_d and b_d.

Benchmark problems for sensitivity analysis

In chapter III, section 4 we analyze the convergence of Bayesian MC on a number of benchmark problems for sensitivity analysis, which are collected in (Saltelli et al., 2000a, Chap. 2). All are nonlinear, and range from 2 to 20 input dimensions.

Example 1 (Saltelli et al., 2000a).

The problem has $D = 2$ dimensions, where $p(\mathbf{x}) = \mathcal{U}(0,1)^2$:

$$f(\mathbf{x}) = x_1 + x_2^4$$

Mean and variance can be calculated exactly: $\mathrm{mean}_{\mathbf{x}}[f] = 0.7$, $\mathrm{var}_{\mathbf{x}}[f] = 139/900$.

Example 2 (Sobol and Levitan, 1999).

This problem consists of two versions with uniform input distribution, $p(\mathbf{x}) = \mathcal{U}(0,1)^D$:

$$f(\mathbf{x}) = \exp\left[\sum_\ell b_\ell x_\ell\right] - \prod_\ell \frac{\exp b_\ell - 1}{b_\ell}.$$

a) $D = 6, \quad \mathbf{b} = (1.5,\ 0.9, \ldots, 0.9)$.
One obtains $\mathrm{mean}_{\mathbf{x}}[f] = 0$ and $\mathrm{var}_{\mathbf{x}}[f] = 427.2751$.

b) $D = 20$, $b_\ell = 0.6$ for $\ell = 1 \ldots 10$, $b_\ell = 0.4$ for $\ell = 11 \ldots 20$.
Here $\mathrm{mean}_{\mathbf{x}}[f] = 0$ and $\mathrm{var}_{\mathbf{x}}[f] = 18\,022$.

Example 3 (Gardner et al., 1981).

Problem 3 is two dimensional, $D = 2$:

$$f(\mathbf{x}) = x_2^4 / x_1^2 .$$

Two subproblems are given through

a) $p(\mathbf{x}) = \mathcal{U}(0.9, 1.1)^2$ with $\text{mean}_{\mathbf{x}}[f] = \frac{51001}{49500}$ and $\text{var}_{\mathbf{x}}[f] = \frac{115340737213}{1637379562500}$

b) $p(\mathbf{x}) = \mathcal{U}(0.5, 1.5)^2$ with $\text{mean}_{\mathbf{x}}[f] = \frac{121}{60}$ and $\text{var}_{\mathbf{x}}[f] = \frac{503101}{72900}$.

Example 4 (Saltelli and Sobol, 1995).

Benchmark problem 4,

$$f(\mathbf{x}) = \prod_{\ell} \frac{|4x_\ell - 2| + a_\ell}{1 + a_\ell},$$

with $\mathbf{a} = (0, 1, 4.5, 9, 99, 99, 99, 99)$ has $D = 8$ dimensions, which are uniformly distributed $p(\mathbf{x}) = \mathcal{U}(0, 1)^8$.
$\text{mean}_{\mathbf{x}}[f] = 1$ and $\text{var}_{\mathbf{x}}[f] = \prod_\ell \frac{-a_\ell^3 + (2+a_\ell)^3}{6(1+a_\ell)^2} \approx 0.465424432$.

Example 5 (Ishigami and Homma, 1990).

Here we have $D = 3$ dimensions with $p(\mathbf{x}) = \mathcal{U}(-\pi, \pi)^3$ and

$$f(\mathbf{x}) = \sin(x_1) + 7\sin^2(x_2) + \tfrac{1}{10} x_3^4 \sin(x_1).$$

For mean and variance one obtains $\text{mean}_{\mathbf{x}} = \frac{7}{2}$ and $\text{var}_{\mathbf{x}}[f] = \frac{\pi^4}{50} + \frac{\pi^8}{1\,800} + \frac{1}{2} + \frac{49}{8}$.

C. Estimates of entropy and (conditional) mutual information

In the feature selection setup we are given a target vector $\mathbf{y}$ and feature vectors $(x_{1\ell}, x_{2\ell} \dots x_{N\ell})^T$, which contain the binary class label and feature ℓ for each sample. If we think of the entries as samples from a joint distribution, we can approximate the MI in (2.8) by replacing the probability distributions by their empirical counterparts:

$$p(Y=1) \approx \tfrac{1}{N} \sum_i y_i = 1 - p(Y=0) \qquad \text{(C.19a)}$$

$$p(X_\ell=1) \approx \tfrac{1}{N} \sum_i x_{i\ell} = 1 - p(X_\ell=0) \qquad \text{(C.19b)}$$

$$p(X_\ell=1, Y=1) \approx \tfrac{1}{N} \sum_i x_{i\ell}\, y_i \qquad \dots \text{ and so forth.} \qquad \text{(C.19c)}$$

The discrete nature of features and class labels makes the estimates extremely simple. If a feature X_ℓ was continuous, one would need to estimate the density $p(x_\ell)$ using a discretization or a more sophisticated method such as kernel density estimation (Hastie et al., 2001, chap.6).

Bibliography

P. Abrahamsen. A review of Gaussian random fields and correlation functions. Technical Report 917, Norwegian Computing Center/Applied Research and Development, 1997.

J. M. Agosta and T. Gardos. Bayes network "smart" diagnostics. *Intel Technology Journal*, 8(4):361–372, 2004.

H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proc. of the Ninth National Conference on Artificial Intelligence (AAAI)*, volume 2, pages 547–552. AAAI Press, 1991.

K. J. Antreich, H. E. Graeb, and C. U. Wieser. Circuit analysis and optimization driven by worst-case distances. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 13(1):57–71, 1994.

M.-F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *ICML*, 2006.

J. W. Bandler and S. H. Chen. Circuit optimization: the state of the art. *IEEE Trans. on Microwave Theory and Techniques*, 36(2):424–443, 1988.

E. B. Baum. Neural net algorithms that learn in polynomial time from examples and queries. *IEEE Trans. on Neural Networks*, 2(1):5–19, 1991.

M. Bensch, M. Schröder, M. Bogdan, and W. Rosenstiel. Feature selection for high-dimensional industrial data. In *ESANN*, 2005.

J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer NY, 1985.

F. Bergeret and C. Le Gall. Yield improvement using statistical analysis of process dates. *IEEE Trans. on Semiconductor Manufacturing*, 16(3):535–542, 2003.

M. C. Bernardo, R. Buck, L. Liu, W. A. Nazaret J. Sacks, and W. J. Welch. Integrated circuit design optimization using a sequential strategy. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 11 (3):361–372, 1992.

C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

B. V. Bonnlander and A. S. Weigend. Selecting input variables using mutual information and nonparametric density estimation. In *Proc. of the Int. Symp. on Artificial Neural Networks*, pages 42–50, 1994.

P. S. Bradley, O. L. Mangasarian, and W. N. Street. Feature selection via mathematical programming. *INFORMS Journal on Computing*, 10(2):209–217, 1998.

D. Braha and A. Shmilovici. Data mining for improving a cleaning process in the semiconductor industry. *IEEE Trans. on Semiconductor Manufacturing*, 15(1):91–101, 2002.

R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli. A survey of optimization techniques for integrated circuit design. *Proc. of the IEEE*, 69 (10):1334–1392, 1981.

B. Bryan, J. Schneider, R. C. Nichol, C. J. Miller, C. R. Genovese, and L. Wasserman. Active learning for identifying function threshold boundaries. In *NIPS 18*, 2006.

W. L. Buntine. Graphical models for discovering knowledge. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 59–82. MIT Press, 1996.

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *ICML*, pages 111–118, 2000.

R. Castro, R. Willett, and R. Nowak. Faster rates in regression via active learning. In *NIPS 18*, 2006.

K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3):273–304, 1995.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. URL `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

O. Chapelle. Active learning for parzen window classifier. In *AI Stats*, pages 49–56, 2005.

W. Cheetham, A. Varma, and K. Goebel. Case-based reasoning at general electric. In *FLAIRS Conference*, pages 93–97, 2001.

R. B. Chinnam. Support vector machines for recognizing shifts in correlated and other manufacturing processes. *Int. J. of Production Research*, 40(17): 4449–4466, 2002.

J. Classen, J. Franz, O. A. Prütz, and A. Kretschmann. Design methodology for micromechanical sensors in automotive applications. In *Eurosensors*, 2004.

D. A. Cohn. Neural network exploration using optimal experiment design. In *NIPS 6*, 1994.

D. A. Cohn. Minimizing statistical bias with queries. In *NIPS 9*, 1997.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

S. Conti, C. W. Anderson, M. C. Kennedy, and A. O'Hagan. A Bayesian analysis of complex dynamic computer models. In *Proc. of the 4th International Conference on Sensitivity Analysis of Model Output*, 2004.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley, 1991.

R. T. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 14(1):1–13, 1946.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

Y. Le Cun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *NIPS 2*, 1990.

C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *J. of the American Statistical Assiciation*, 86(416): 953–963, 1991.

S. Dasgupta. Coarse sample complexity bounds for active learning. In *NIPS 18*, 2006.

W. Demtröder. *Experimentalphysik 1. Mechanik und Wärme*. Springer Berlin, 1994.

S. W. Director, P. Feldmann, and K. Krishna. Statistical integrated circuit design. *IEEE Journal of solid-state circuits*, 28(3):193–202, 1993.

F. Duvivier. Automatic detection of spatial signature on wafermaps in a high volume production. In *International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 61–66, 1999.

V. V. Fedorov. *Theory of optimal experiments*. Academic Press, 1972.

J. S. Fenner, M. K. Jeong, and J.-C. Lu. Optimal automatic control of multistage production processes. *IEEE Trans. on Semiconductor Manufacturing*, 18(1):94–103, 2005.

R. P. Feynman, R. B. Leighton, and M. Sands. *The Feynman Lectures on Physics*. Addison-Wesley, MA, 1963.

S. E. Fienberg. When did Bayesian inference become Bayesian? *Bayesian Analysis*, 1(1):1–40, 2006.

F. Fleuret. Fast binary feature selection with conditional mutual information. *JMLR*, 5:1531–1555, 2004.

T. Fountain, T. Dietterich, and B. Sudyka. Data mining for manufacturing control: An application in optimizing ic test. In *Exploring Artificial Intelligence in the New Millenium*, pages 381–400. Morgan-Kaufmann, 2002.

J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–67, 1991.

R. H. Gardner, R. V. O'Neill, J. B. Mankin, and J. H. Carney. A comparison of sensitivity analysis and error analysis based on a stream ecosystem model. *Ecological Modelling*, 12(3):173–190, 1981.

M. N. Gibbs. *Bayesian Gaussian Processes for Regression and Classification*. PhD thesis, University of Cambridge, 1997.

R. Gilad-Bachrach, A. Navot, and N. Tishby. Query by committee made real. In *NIPS 18*, 2006.

K. Glien, J. Graf, H. Höfer, M. Ebert, and J. Bagdahn. Strength and reliability properties of glass frit bonded micro packages. In *Conference on Design, Test, Integration and Packaging of MEMS/MOEMS*, 2004.

P. W. Goldberg, C. K. I. Williams, and C. M. Bishop. Regression with input-dependent noise: A Gaussian process treatment. In *NIPS 10*, 1998.

R. B. Gramacy, H. K. H. Lee, and W. MacReady. Parameter space exploration with Gaussian process trees. In *ICML*, pages 353–360, 2004.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR*, 3:1157–1182, 2003.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3): 389–422, 2002.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer NY, 2001.

R. G. Haylock and A. O'Hagan. On inference for outputs of computationally expensive algorithms with uncertainty on the inputs. In *Bayesian Statistics 5*, pages 629–637, 1996.

C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification, 2001. URL `www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

R. L. Iman and S. C. Hora. A robust measure of uncertainty importance for use in fault tree system analysis. *Risk Analysis*, 10(3):401–406, 1990.

T. Ishigami and T. Homma. An importance quantification technique in uncertainty analysis for computer models. In *Proc. of the First International Symposium on Uncertainty Modeling and Analysis*, pages 398–403, 1990.

E. T. Jaynes. Information Theory and Statistical Mechanics. *The Phys. Rev.*, 106(4):620–630, 1957a.

E. T. Jaynes. Information Theory and Statistical Mechanics II. *The Phys. Rev.*, 108(2):171–190, 1957b.

E. T. Jaynes. Bayesian methods: General background. In *Proc. of Maximum Entropy and Bayesian Methods in Applied Statistics*. Cambridge University Press, 1985.

E. T. Jaynes. *Probability Theory*. Cambridge University Press, 2003.

G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *ICML*, pages 121–129, 1994.

M. E. Johnson, D. Ylvisaker, and L. Moore. Minimax and maximin distance designs. *J. of Statistical Planning and Inference*, 26:131–148, 1990.

M. W. Johnson, Q. P. Herr, and J. W. Spargo. Monte-carlo yield analysis. *IEEE Trans. on applied superconductivity*, 9(2):3322–3325, 1999.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In *Learning in Graphical Models*. MIT Press, 1999.

L. P. Kaelbling, editor. *JMLR, special issue on Variable and Feature Selection*, volume 3, 2003.

R. E. Kass and A. E. Raftery. Bayes factors. *J. Am. Stat. Ass.*, 90(430): 773–795, 1995.

R. E. Kass, B. P. Carlin, A. Gelman, and R. M. Neal. Markov chain Monte Carlo in practice: A roundtable discussion. *The American Statistician*, 52 (2):93–100, 1998.

M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *J. of the Royal Statistical Society Series B*, 63(3):425–464, 2001.

M. C. Kennedy, C. W. Anderson, S. Conti, and A. O'Hagan. Case studies in Gaussian process modeling of computer codes. In *Proc. of the 4th Int. Conf. on Sensitivity Analysis of Model Output*, 2004.

C.-W. Ko, J. Lee, and M. Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691, 1995.

R. Kohavi and G. H. John. The wrapper approach. In H. Liu and H. Motoda, editors, *Feature Selection for Knowledge Discovery and Data Mining*, pages 33–50. Kluwer Academic Publishers, 1998.

J. Kohlmorgen and S. Lemm. An on-line method for segmentation and identification of non-stationary time series. In *Neural Networks for Signal Processing XI*, pages 113–122, 2001.

M. Kuss, T. Pfingsten, L. Csató, and C. E. Rasmussen. Approximate inference for robust Gaussian process regression. Technical Report 136, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, 2005.

T. N. Lal, O. Chapelle, J. Weston, and A. Elisseeff. Embeded methods. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature extraction, foundations and Applications*, chapter Embeded methods. Springer NY, 2006.

P. Langley and W. Iba. Average-case analysis of a nearest neighbor algorithm. In *Proc. of the 13th Int. Joint Conference on Artificial Intelligence*, 1993.

P. Langley and S. Sage. Oblivious decision trees and abstract cases. In *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*. AAAI Press, 1994.

S. Leang, S.-Y. Ma, J. Thomson, B. J. Bombay, and C. J. Spanos. A control system for photolithographic sequences. *IEEE Trans. on Semiconductor Manufacturing*, 9(2):191–207, 1996.

D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proc. of the 17th annual int. ACM SIGIR conf. on research and development in information retrieval*, pages 3–12. Springer NY, 1994.

X. Li, J. Le, L. Pileggi, and A. Strojwas. Projection-based performance modeling for inter/intra-die variations. In *Proc. of IEEE/ACM Int. Conference on Computer Aided Design*, pages 721–727, 2005.

D. V. Lindley. On the measure of information provided by an experiment. *Ann. math. Statist.*, 27(4):986–1005, 1956.

D. V. Lindley. The choice of variables in multiple regression. *J. of the Royal Statistical Society B*, 30(1):31–66, 1968.

K. K. Low and S. W. Director. A new methodology for the design centering of IC fabrication processes. *IEEE Trans. on Computer-aided design*, 10(7): 895–903, 1991.

D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992a.

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

D. J. C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):589–603, 1992b.

D. J. C. MacKay. Comparison of approximate methods for handling hyperparameters. *Neural Computation*, 11(5):1035–1068, 1999.

M. D. MacKay, R. J. Beckmann, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

M. Manago and E. Auriol. Using data mining to improve feedback from experience for equipment in the manufacturing & transport industries. In *IEE Colloquium on Knowledge Discovery and Data Mining (Digest No. 1996/198)*, 1996.

G. Matheron. Principles of geostatistics. *Economic Geology*, 58(8):1246–1266, 1963.

P. Mitra, C. A. Murthy, and S. K. Pal. A probabilistic active support vector learning algorithm. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(3):413–418, 2004.

G. E. Moore. Cramming more components onto integrated circuits. *Electronics, April 19*, 86(1):114–117, 1965.

M. D. Morris. Input screening: Finding the important inputs on a budget. In *Proc. of the 4th Int. Conference on Sensitivity Analysis of Model Output*, 2004.

M. D. Morris, T. J. Mitchell, and D. Ylvisaker. Bayesian design and analysis of computer experiments: Use of derivatives in surface prediction. *Technometrics*, 35(3):243–255, 1993.

R. H. Myers and D. C. Montgomery. *Response Surface Methodology*. Wiley Series in Probability and Statistics, 2002.

R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG–TR–93–1, University of Toronto, 1993.

R. M. Neal. *Bayesian Learning for Neural Networks*. Springer NY, 1996.

R. M. Neal. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report CRG–TR–97–2, University of Toronto, 1997.

A. Y. Ng. On feature selection: Learning with exponentially many irrelevant features as training examples. In *ICML*, 1998.

G. De Nicolao, E. Pasquinetti, G. Miraglia, and F. Piccinini. Unsupervised spatial pattern classification of electrical failures in semiconductor manufacturing. In *Proc. of the Artificial Neural Networks in Pattern Recognition, Florence*, 2003.

H. Niederreiter. *Random number generation and quasi-Monte Carlo methods.* SIAM, 1992.

D. J. Nott and P. J. Green. Bayesian variable selection and the Swendsen-Wang algorithm. *J. of Computational and Graphical Statistics*, 13(1):1–17, 2004.

J. E. Oakley and A. O'Hagan. Bayesian inference for the uncertainty distribution of computer model outputs. *Biometrika*, 89:769–784, 2002.

J. E. Oakley and A. O'Hagan. Probabilistic sensitivity analysis of complex models: a Bayesian approach. *J. of the Royal Statistical Society B*, 66(3): 751–769, 2004.

A. O'Hagan. Curve fitting and optimal design for prediction. *J. R. Stat. Soc. B*, 40(1):1–42, 1978.

A. O'Hagan. Monte Carlo is fundamentally unsound. *The Statistician*, 36 (2/3):247–249, 1987.

A. O'Hagan. Bayes-Hermite quadrature. *J. of Statistical Planning and Inference*, 29(3):245–260, 1991.

A. O'Hagan. Bayesian analysis of computer code outputs: A tutorial. Technical Report No. 543/04, Department of Probability and Statistics, University of Sheffield, 2004.

A. O'Hagan, M. C. Kennedy, and J. E. Oakley. Uncertainty analysis and other inference tools for complex computer codes. In *Bayesian Statistics 6*, pages 503–524, 1998.

C. J. Paciorek and M. J. Schervish. Nonstationary covariance functions for Gaussian process regression. In *NIPS 16*, 2004.

T. Pfingsten. Bayesian active learning for sensitivity analysis. In *ECML*, 2006.

T. Pfingsten and K. Glien. Statistical analysis of slow crack growth experiments. *J. of the European Ceramic Society*, 26(15):3061–3065, 2006.

T. Pfingsten and C. E. Rasmussen. Fully Bayesian model selection for Gaussian process regression. (to appear), 2006.

T. Pfingsten, D. J. L. Herrmann, T. Schnitzler, A. Feustel, and B. Schölkopf. Feature selection for trouble shooting in complex assembly lines. *IEEE Trans. on Automation Science and Engineering (in press)*, 2005.

T. Pfingsten, D. J. L. Herrmann, and C. E. Rasmussen. Model-based design analysis and yield optimization. *IEEE Trans. on Semiconductor Manufacturing*, 19(4):475–486, 2006a.

T. Pfingsten, M. Kuss, and C. E. Rasmussen. Nonstationary Gaussian process regression using a latent extension of the input space. URL `http://www.kyb.mpg.de/~tpfingst`. 2006b.

J. C. Platt. Probabilities for support vector machines. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large-Margin Classifiers*, pages 61–74. MIT Press, 1999.

M. Plutowski and H. White. Selecting concise training sets from clean data. *IEEE Trans. on Neural Networks*, 4(2):1993, 1993.

W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 1st edition, 1986.

P. Pudil, J. Novovičová, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994.

S. Rao, A. J. Strojwas, J. P. Lehoczky, and M. J. Schervish. Monitoring multistage integrated circuit fabrication processes. *IEEE Trans. on Semiconductor Manufacturing*, 9(4):495–505, 1996.

C. E. Rasmussen. Gaussian processes to speed up hybrid Monte Carlo for expensive Bayesian integrals. In *Bayesian Statistics 7*, pages 651–659, 2003.

C. E. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. In *NIPS 15*, 2003.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

J. E. Rayas-Sánchez. EM-based optimization of microwave circuits using artificial neural networks: The state-of-the-art. *IEEE Trans. on Microwave Theory and Techniques*, 52(1):420–435, 2004.

W. C. Riordan, R. Miller, and E. R. St. Pierre. Reliability improvement and burn in optimization through the use of die level predictive modeling. In *Proc. of Int. Reliability Physics Symposium*, pages 435–445, 2005.

N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *ICML*, 2001.

J. Sacks and D. Ylvisaker. Design for regression problems with correlated errors. *The Annals of Mathematical Statistics*, 37(1):66–89, 1966.

J. Sacks and D. Ylvisaker. Design for regression problems with correlated errors: Many parameters. *The Annals of Mathematical Statistics*, 39(1): 49–69, 1968.

J. Sacks and D. Ylvisaker. Linear estimation for approximately linear models. *The Annals of Statistics*, 6(5):1122–1137, 1978.

J. Sacks, S. B. Schiller, and W. J. Welch. Design for computer experiments. *Technometrics*, 31(1):41–47, 1989a.

J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989b.

A. Saltelli. Making best use of model evaluations to compute sensitivity indices. *Computer Physics Communications*, 145(2):280–297, 2002.

A. Saltelli and I. M. Sobol. About the use of rank transformation in sensitivity analysis of model output. *Reliability Engineering*, 50(3):225–239, 1995.

A. Saltelli, K. Chan, and E. M. Scott. *Sensitivity Analysis*. Wiley, 2000a.

A. Saltelli, S. Tarantola, and F. Campolongo. Sensitivity analysis as an ingredient of modeling. *Statistical Science*, 15(4):377–395, 2000b.

G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *J. of the am. society for information science*, 41(4):288–297, 1990.

A. Mello Schmidt and A. O'Hagan. Bayesian inference for nonstationary spatial covariance structure via spatial deformations. *J. of the Royal Statistical Society B*, 65:745–758, 2003.

B. Schölkopf and J. Smola. *Learning with Kernels*. MIT Press, 2002.

S. Seo, M. Wallat, T. Graepel, and K. Obermayer. Gaussian process regression: Active data selection and test point rejection. In *IEEE-INNS-ENNS Int. Joint Conference on Neural Networks*, pages 241–246, 2000.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proc. of the 5th anual workshop on Computational learning theory*, pages 287–294. ACM Press, 1992.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, 1948.

I. M. Sobol. Sensitivity estimates for nonlinear mathematical models. *Mathematical Modeling and Computational Experiment*, 1(4):407–411, 1993.

I. M. Sobol and Y. L. Levitan. On the use of variance reducing multipliers in Monte Carlo computations of a global sensitivity index. *Computer Physics Communications*, 117(1):52–61, 1999.

S. D. Stearns. On selecting features for pattern classifiers. In *Third Int. Conf. on Pattern Recognition*, 1976.

M. L. Stein. *Interpolation of Spatial Data. Some Theory for Krigging*. Springer NY, 1999.

G. Taguchi, R. Jugulum, and S. Taguchi. *Computer-Based Robust Engineering*. ASQ Quality Press, 2004.

B. Tang. Orthogonal array-based Latin Hypercube samples. *J. of the American Statistical Associtation*, 88(424):1392–1397, 1993.

G. C. Temes and D. A. Calahan. Computer-aided network optimization: the state-of-the-art. *Proc. of the IEEE*, 55(11):1832–1863, 1967.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. of the Royal Statistical Society B*, 58(1):267–288, 1996.

K. W. Tobin, T. P. Karnowski, and F. Lakhani. Integrated applications of inspection data in the semiconductor manufacturing environment. In *Proc. of SPIE*, volume 4275, pages 31–40, 2001.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer New York, 1995.

M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *Proc. of the Ninth IEEE Int. Conf. on Computer Vision*, page 281, 2003.

W. J. Welch, R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell, and M. D. Morris. Screening, prediction, and computer experiments. *Technometrics*, 34(1):15–25, 1992.

J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping. Use of the zero-norm with linear models and kernel methods. *JMLR*, 3(7-8):1439–1461, 2003a.

J. Weston, F. Pérez-Cruz, O. Bousquet, O. Chapelle, A. Elisseeff, and B. Schölkopf. Feature selection and transduction for prediction of molecular bioactivity for drug design. *Bioinformatics*, 19(6):764–771, 2003b.

S. A. Vander Wiel. Monitoring processes that wander using integrated moving average models. *Technometrics*, 38(2):139–151, 1996.

J. E. Wieringa. *Statistical process control for serially correlated data*. PhD thesis, Rijksuniversiteit Groningen, 1999.

Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, pages 412–420, 1997.

K. Q. Ye. Orthogonal column Latin Hypercubes and their application in computer experiments. *J. of the American Statistical Association in Computer Experiments*, 93(444):1430–1439, 1998.

K. Yu, J. Bi, and V. Tresp. Active learning via transductive experimental design. In *ICML*, 2006.

A. H. Zaabab, Q. J. Zhang, and M. Nakhla. A neural network modeling approach to circuit optimization and statistical design. *IEEE Trans. on microwave theory and techniques*, 43(6):1349–1358, 1995.

Index

A-Optimal, 64
accelerometer, 57
ARD, 34
automatic relevance determination, 34

Bayes' factor, 29
Bayes' theorem, 27
Bayesian expected utility, 31
Bayesian experimental design, 24

case-based reasoning, 21
conditional mutual information, 92
correlation coefficients, 90
correlation ratios, 47
cross correlation ratios, 48

D-Optimal, 64
defect maps, 20
discrepancy, 50

Electronic Stability Program, 59
embedded methods, 87
entropy, 69, 90
error bins, 19
evidence, 27

factorial designs, 65
feasibility region, 42
feature, 84
feature selection, 21
filter methods, 89
final tests, 17
Fisher score, 90
foundry, 15

Gaussian process, 23, 25
geometrical design centering, 44
GP, 25
greedy active learning, 67

hyperparameters, 29

in-line measurements, 17
indicator function, 41
information gain, 69
ink, 19
input distribution, 40
input vector, 84
integrated circuit, 15
irrelevant, 85
isotropic, 33

Latin Hypercube, 50, 106
length scale, 34
likelihood function, 26
local correlation ratios, 47
local/global sensitivity measures, 44
lot, 17
lot-history, 18

marginal likelihood, 27
Markov Chain Monte Carlo, 28
Matérn kernels, 36
MaxiMin, 65
Maximum likelihood of type-II, 29
MCMC, 28
mean square differentiable, 35
MEMS, 15
MiniMax, 65
ML-II, 29
Monte Carlo, 27, 43, 50
MSE, 75
mutual information, 90

nominal parameters, 41
nonparametric models, 31

Occam's razor, 30
optical inspection, 20

parametric yield, 42

pool approach, 70
posterior, 27
posterior odds, 29
pressure sensor, 48
prior, 26
probabilistic models, 24
process modeling, 20
process tolerances, 41

Quasi-Monte Carlo, 50
Query by Committee Machine, 66

rational quadratic kernel, 36
relevance sampling, 65
relevant, 85
response surface, 44

score, 94
screening, 45
semiconductor manufacturing, 15
sensitivity analysis, 23, 44
Sheward chart, 20
smoothness, 35
space filling designs, 64
SPC, 20
squared exponential kernel, 35
standardized regression coefficients, 46
stationary, 33
SVM, 93

target, 85
troubleshooting, 21

uncertainty analysis, 23, 41
uncertainty distribution, 41
uncertainty sampling, 66
utility function, 31, 53

variability, 35
version space, 66

wafer, 16
wafer-level tests, 18
wafermaps, 19
wrappers, 88

yaw rate sensor, 16, 59
yield, 21