STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG

MARKUS KAUPP

Ein Verfahren zur automatischen Erzeugung intelligenter Prozessüberwachungssysteme

Universität Stuttgart

Fraunhofer IPA

STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG BAND 36

Herausgeber:
Univ.-Prof. Dr.-Ing. Thomas Bauernhansl
Univ.-Prof. Dr.-Ing. Dr. h.c. mult. Alexander Verl
Univ.-Prof. a. D. Dr.-Ing. Prof. E.h. Dr.-Ing. E.h. Dr. h.c. mult. Engelbert Westkämper

Markus Kaupp

Ein Verfahren zur automatischen Erzeugung intelligenter Prozessüberwachungssysteme

FRAUNHOFER VERLAG

Kontaktadresse:
Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon 0711 970-00, Telefax 0711 970-1399
info@ipa.fraunhofer.de, www.ipa.fraunhofer.de

STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG

Herausgeber:
Univ.-Prof. Dr.-Ing. Thomas Bauernhansl
Univ.-Prof. Dr.-Ing. Dr. h.c. mult. Alexander Verl
Univ.-Prof. a. D. Dr.-Ing. Prof. E.h. Dr.-Ing. E.h. Dr. h.c. mult. Engelbert Westkämper

Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart
Institut für Industrielle Fertigung und Fabrikbetrieb (IFF) der Universität Stuttgart
Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen (ISW)
der Universität Stuttgart

Titelbild: © Patrizia Tilly - Fotolia.com

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.
ISSN: 2195-2892
ISBN (Print): 978-3-8396-0780-0

D 93

Zugl.: Stuttgart, Univ., Diss., 2014

Druck: Mediendienstleistungen des Fraunhofer-Informationszentrum Raum und Bau IRB, Stuttgart
Für den Druck des Buches wurde chlor- und säurefreies Papier verwendet.

Fraunhofer-Informationszentrum Raum und Bau IRB
Postfach 80 04 69, 70504 Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon 0711 970-2500
Telefax 0711 970-2508
E-Mail verlag@fraunhofer.de
URL http://verlag.fraunhofer.de

GELEITWORT DER HERAUSGEBER

Produktionswissenschaftliche Forschungsfragen entstehen in der Regel im Anwendungszusammenhang, die Produktionsforschung ist also weitgehend erfahrungsbasiert. Der wissenschaftliche Anspruch der „Stuttgarter Beiträge zur Produktionsforschung" liegt unter anderem darin, Dissertation für Dissertation ein übergreifendes ganzheitliches Theoriegebäude der Produktion zu erstellen.

Die Herausgeber dieser Dissertations-Reihe leiten gemeinsam das Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA und jeweils ein Institut der Fakultät für Konstruktions-, Produktions- und Fahrzeugtechnik an der Universität Stuttgart.

Die von ihnen betreuten Dissertationen sind der marktorientierten Nachhaltigkeit verpflichtet, ihr Ansatz ist systemisch und interdisziplinär. Die Autoren bearbeiten anspruchsvolle Forschungsfragen im Spannungsfeld zwischen theoretischen Grundlagen und industrieller Anwendung.

Die „Stuttgarter Beiträge zur Produktionsforschung" ersetzt die Reihen „IPA-IAO Forschung und Praxis" (Hrsg. H.J. Warnecke / H.-J. Bullinger / E. Westkämper / D. Spath) bzw. ISW Forschung und Praxis (Hrsg. G. Stute / G. Pritschow / A. Verl). In den vergangenen Jahrzehnten sind darin über 800 Dissertationen erschienen.

Der Strukturwandel in den Industrien unseres Landes muss auch in der Forschung in einen globalen Zusammenhang gestellt werden. Der reine Fokus auf Erkenntnisgewinn ist zu eindimensional. Die „Stuttgarter Beiträge zur Produktionsforschung" zielen also darauf ab, mittelfristig Lösungen für den Markt anzubieten. Daher konzentrieren sich die Stuttgarter produktionstechnischen Institute auf das Thema ganzheitliche Produktion in den Kernindustrien Deutschlands. Die leitende Forschungsfrage der Arbeiten ist: Wie können wir nachhaltig mit einem hohen Wertschöpfungsanteil in Deutschland für einen globalen Markt produzieren?

Wir wünschen den Autoren, dass ihre „Stuttgarter Beiträge zur Produktionsforschung" in der breiten Fachwelt als substanziell wahrgenommen werden und so die Produktionsforschung weltweit voranbringen.

Alexander Verl Thomas Bauernhansl Engelbert Westkämper

Ein Verfahren zur automatischen Erzeugung intelligenter Prozessüberwachungssysteme

Von der Fakultät Konstruktions-, Produktions- und Fahrzeugtechnik
der Universität Stuttgart
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Abhandlung

Vorgelegt von
Markus Kaupp
aus Tübingen

Hauptberichter:	Prof. Dr.-Ing. Dr. h. c. mult. Alexander Verl
Mitberichter:	Prof. Dr.-Ing. Jörg Krüger
Tag der mündlichen Prüfung:	11.07.2014

Institut für Steuerungstechnik der Werkzeugmaschinen und
Fertigungseinrichtungen der Universität Stuttgart
2014

Vorwort des Autors

Die vorliegende Arbeit entstand während meiner Tätigkeit am Fraunhofer-Institut für Produktionstechnik und Automatisierung (IPA) in Stuttgart. Ich danke Herrn Professor Alexander Verl für die Unterstützung meiner Arbeit und für die Übernahme des Hauptberichts. Herrn Professor Jörg Krüger danke ich für die Durchsicht der Arbeit und die Übernahme des Mitberichts.

Ein großer Dank geht auch an meine ehemaligen Fraunhofer-Kollegen aus der Abteilung Bild- und Signalverarbeitung unter der Leitung von Markus Hüttel. Speziell danke ich hier meinen langjährigen Weggefährten Hartmut Eigenbrod, Marius Pflüger und Joachim Neher.

Ganz besonders danke ich meinen Eltern für die Entfaltungsmöglichkeiten, die sie mir gaben, und für die Freiheiten, die sie mir ließen. Ich danke auch meiner Frau. Sie war und ist mir eine wichtige Hilfe und eine große Stütze.

Stuttgart, im August 2014 Markus Kaupp

Kurzinhalt

Eine Voraussetzung für die Automatisierung von Produktionsprozessen ist die Existenz zuverlässiger Prozessüberwachungssysteme. Solche Systeme ermöglichen es, ungünstige Prozesszustände schnell zu erkennen. Prozessüberwachungssysteme erfassen Sensordaten aus dem zu überwachenden Prozess. Aus den Sensordaten wird – entweder nach starren Regeln oder mittels künstlicher Intelligenz – der aktuelle Prozesszustand abgeleitet. Die intelligenten Systeme gelten dabei als die leistungsfähigere Variante. Bisher ist die Erstellung intelligenter Prozessüberwachungssysteme sehr zeitaufwändig und erfordert ein hohes Maß an Expertenwissen. Dies ist ein Hemmnis für den flächendeckenden Einsatz solcher Systeme. In dieser Arbeit wird ein Verfahren für die automatische Erzeugung intelligenter Prozessüberwachungssysteme für beliebige zyklische Fertigungsprozesse vorgestellt. Für die Umsetzung wurde ein generisches Prozessüberwachungssystem implementiert. Dieses bietet die Infrastruktur für die Datenerfassung und die benötigten Datenflüsse. Das System enthält zunächst keine Logik für die Verarbeitung und Bewertung der erfassten Daten. Diese Logik wird von außen in Form eines Analysemodells vorgegeben. Solch ein Analysemodell ist eine Verarbeitungskette, die aus aufeinander abgestimmten Verfahren für die Signalverarbeitung,

die Kenngrößenbildung, die Kenngrößenselektion und die Klassifikation besteht. Durch Setzen eines geeigneten Analysemodells lässt sich das generische Prozessüberwachungssystem an jeden Fertigungsprozess anpassen. Mit diesem Konzept ist das Erzeugen eines Prozessüberwachungssystems für einen Fertigungsprozess ein Optimierungsproblem: Man sucht dasjenige Analysemodell, das das generische Prozessüberwachungssystem am besten an den Fertigungsprozess anpasst. Für die Lösung dieses Optimierungsproblems wurde ein Optimierungsverfahren mit dem Namen Artificial-Bee-Colony-Optimierung gewählt. Im Rahmen der hier beschriebenen Arbeit wurde dieses Optimierungsverfahren entscheidend erweitert, sodass es auf die gegebene Problemstellung angewandt werden konnte.

Short Summary

Manufacturing sites in developed countries can only exist in the long run if their production processes are automated. Prerequisite for this automation is the existence of reliable monitoring systems to detect and fix unfavorable process states. Monitoring systems capture sensor data from the observed process. The current process state is deduced from this data, either based on fixed rules (thresholds, envelopes ...) or by the means of artificial intelligence. Intelligent process monitoring systems are considered to be the more powerful type. Currently the creation of intelligent process monitoring systems for a given manufacturing method is time-consuming and requires a high degree of expert knowledge. This is a major barrier for a comprehensive application of such systems. This thesis presents a method for the automatic creation of intelligent process monitoring systems for arbitrary cyclic production processes. These production processes can be described by a set of training data. Each data record in this training data set contains both: data that has been measured in a single cycle of the production process, and the desired prediction result for this production cycle. Based on these training data, the new method generates a process monitoring system that is able to assign predictions even to such data records that have not been part of the training

data set. For the realization of the proposed system, a generic process monitoring system was designed and implemented. This system provides the infrastructure for data acquisition and the data streams required for the generated monitoring systems. In the first stage, the generic system does not hold any program logic for processing and evaluating the acquired data. This logic is provided by an external analysis model. Such a model is a processing chain integrating methods for signal preprocessing, feature generation, feature selection and classification. By setting the analysis model, the generic process monitoring system can be adapted to any manufacturing process. With the concept described above, the creation of a process monitoring system for a manufacturing process can be reduced to an optimization problem. The goal is to find the analysis model that adapts best the generic monitoring system to the given manufacturing process. To solve this optimization problem, a heuristic optimization algorithm named Artificial Bee Colony Optimization is applied. For the method proposed in this thesis, the original Artificial Bee Colony Optimization was adapted to handle non-real valued problem spaces.

Inhaltsverzeichnis

Abkürzungen und Formelzeichen

Abkürzungen

AR	Auto Regressive
ARMA	Auto Regressive Moving Average
C-SVM	C-Support Vector Machine
FFT	schnelle Fourier-Transformation
GRNFN	General Regression Neuro-Fuzzy Network
LVQ	Learning Vector Quantization
NEPRES	Neuronales Prozessregelungssystem
NEPRES II	Verbundforschungsprojekt Qualitätsorientierte Prozessüberwachung und -regelung zyklischer Produktionsprozesse
ν-SVM	ν-Support Vector Machine
QP-UVS	Verbundforschungsprojekt Qualitätsprognose mittels Prozessinformation beim Ultraschall- und Vibrationsschweißen
QSLR	Verbundforschungsprojekt Qualitätssicherung bei Low-Runner-Prozessen

SC-QUPUS	Verbundforschungsprojekt Signalbasiertes Clustering zur qualitätsorientierten Prozessüberwachung beim Ultraschallschweißen thermoplastischer Kunststoffprodukte

Arabische Formelzeichen

b	Offset-Wert
c_i	Ausgabeklasse
$\mathcal{D}$	Menge von Lerndatensätzen
E	Erwartungswert einer Zufallsvariablen
$\mathcal{H}$	Hilbertraum
K	Kernelfunktion
$\mathcal{K}$	Menge von Kenngrößen
k_v	Anzahl interner Validierungsdurchläufe
L	Schadensfunktion für die Bewertung von Prognosen
m	Anzahl der Lerndatensätze
$\mathcal{M}$	Menge aller Modellbeschreibungen
N	Normalverteilung
n_max	maximale Anzahl fruchtloser Optimierungsiterationen
P	Wahrscheinlichkeit
$\mathcal{P}(X)$	Potenzmenge der Menge X
q	Grad eines Polynoms
$R[f]$	Erwartetes Risiko eines Lernmodells
T	unbalancierte Trefferquote
T_B	balancierte Trefferquote
$\bar{T}$	durchschnittliche Trefferquote
$\boldsymbol{w}$	Normalenvektor

$\boldsymbol{w'}$	Urbild eines Normalenvektors
$\boldsymbol{x}$	Eingabevektor
$\mathcal{X}$	Menge von Eingangsvariablen einer Prognoseaufgabe
y	Klassifikationsergebnis
$\mathcal{Y}$	Wertebereich der Ausgabewerte einer Prognoseaufgabe
z	Anzahl der Klassen in der Ergebnismenge

Griechische Formelzeichen

$\boldsymbol{\alpha}$	Vektor von Parametern
γ	Regularisierungsparameter
γ_t	Faktor für die Gewichtung der balancierten und unbalancierten Trefferquoten
$\boldsymbol{\theta}$	Vektor von Hyperparametern
μ	Mittelwert einer Normalverteilung
$\boldsymbol{\xi}$	Vektor von Schlupfvariablen
σ	Standardabweichung einer Normalverteilung
τ	Entscheidungsfunktion einer Support Vector Machine
ϕ	Abbildung vom Eingaberaum in einen Hilbertraum

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Problemstellung

Fertigungsbetriebe können in Hochlohnländern nur dann bestehen, wenn sie sich durch Flexibilität, Termintreue und Qualität vom globalen Wettbewerb abheben. Daher müssen Fertigungsstätten an solchen Standorten in der Lage sein, qualitativ hochwertige Teile kostengünstig, schnell und zuverlässig zu fertigen [Wiendahl u. a. 2004, Westkämper u. a. 2009]. Eine weitgehend automatisierte Fertigung gilt als Schlüssel für das Erreichen dieser Ziele. Eine wesentliche Voraussetzung für die Automatisierung von Fertigungsprozessen ist die Existenz zuverlässiger Prozessüberwachungssysteme. Solche Systeme ermöglichen es, im Falle ungünstiger Prozesszustände schnell in den Prozess eingreifen zu können [Liang u. a. 2004, Kovač u. a. 2011].

Die moderne Prozessüberwachung basiert auf der Erfassung und Ermittlung von Prozessvariablen. Daher wurde in den vergangenen Jahrzehnten eine Vielzahl

von Sensoren entwickelt, die speziell auf die Datenerfassung im Fertigungsumfeld zugeschnitten sind. Solche Sensoren können direkt in den Fertigungsprozess integriert werden. Zudem wurden zahlreiche Verfahren für die Verarbeitung und Analyse der gewonnenen Signale erarbeitet, die eine zyklussynchrone Überwachung von Produktionsprozessen erlauben [Liang u. a. 2004, König 2007]. In einigen Bereichen der Kunststoffverarbeitung ist eine laufende und dokumentierte Prozessüberwachung bereits seit langem Stand der Technik und wird oft auch vom Kunden als Qualitätsdokumentation gefordert [Wurm 2006].

Die in der Praxis eingesetzten Prozessüberwachungssysteme schließen meist von Anomalien in einzelnen Signalverläufen direkt auf Prozessstörungen oder Qualitätsabweichungen. Die Überwachung erfolgt dann, indem überprüft wird, ob alle Signale innerhalb zuvor festgelegter Grenzwerte oder Hüllkurven verlaufen. Die meisten Fertigungsprozesse sind aber aus systemtheoretischer Sicht hochgradig komplex. Die Grenzwert- und die Hüllkurvenüberwachung können folglich nur einen kleinen Teil der im Prozess auftretenden Phänomene ausreichend gut erkennen. Daher lag in den vergangenen Jahrzehnten ein starker Fokus auf der Entwicklung intelligenter Prozessüberwachungssysteme. Die intelligenten Prozessüberwachungssysteme gelten als die leistungsfähigsten Systeme zur sensordatenbasierten Prozessüberwachung. Diese Systeme beschreiben die Fertigungsprozesse mit Modellen aus dem Bereich des überwachten maschinellen Lernens. Dadurch sind sie in der Lage, auch die komplexen Zusammenhänge stark wechselwirkungsbehafteter Fertigungsprozesse abzubilden [Liang u. a. 2004, Abellan-Nebot u. a. 2010, Neher 2012].

Allerdings ist die Entwicklung eines intelligenten Überwachungssystems für einen Fertigungsprozess eine schwierige und langwierige Aufgabe. Nach dem aktuellen Stand der Forschung benötigt man für solch eine Anpassung einerseits detaillierte Kenntnisse über den jeweiligen Fertigungsprozess und andererseits Ex-

pertenwissen auf dem Gebiet der Signalanalyse. Auch der zeitliche Aufwand ist sehr hoch. Nicht selten nimmt die Entwicklung solch eines Systems mehrere Jahre in Anspruch [Jardine u. a. 2006, Teti u. a. 2010].

Der hohe Aufwand, der für die Entwicklung eines intelligenten Prozessüberwachungssystems betrieben werden muss, gilt als ein wesentliches Hemmnis für die flächendeckende Verbreitung solcher Systeme. So finden die intelligenten Prozessüberwachungssysteme bis heute in der Industrie nur geringe Akzeptanz. Die geringe Nachfrage führt dazu, dass nur sehr wenige kommerzielle Lösungen angeboten werden. Das große Potential der intelligenten Prozessüberwachung bleibt somit in der Praxis noch weitgehend ungenutzt [Liang u. a. 2004, Kitazawa u. a. 2011].

1.2 Zielsetzung und Vorgehensweise

Die Entwicklung eines Prozessüberwachungssystems für einen Fertigungsprozess ist selbst für Experten sehr aufwändig. Auch heute noch nimmt die Erstellung solch eines Systems mehrere Personenjahre in Anspruch. Ziel dieser Arbeit ist die Erarbeitung und Implementierung eines Verfahrens, das solche Prozessüberwachungssysteme automatisch erzeugt. Dieses Verfahren soll dabei so beschaffen sein, dass auch Nicht-Experten Überwachungssysteme erstellen können. Zudem soll die Erzeugung solch eines Systems im Vergleich zum Ist-Zustand signifikant beschleunigt werden.

Hierbei soll ein Ansatz verfolgt werden, der auf einem generischen Prozessüberwachungssystem basiert. Dieses System soll so gestaltet sein, dass es schnell an jeden zu überwachenden Fertigungsprozess angepasst werden kann. Der gewählte Lösungsansatz wird in dieser Arbeit detailliert beschrieben.

Im folgenden Kapitel wird zunächst der aktuelle Stand von Wissenschaft und Technik vorgestellt. Diesen Betrachtungen schließt sich eine differenzierte Anforderungsanalyse für das zu realisierende Verfahren an. Das konkret erarbeitete Verfahren wird im Kapitel 4 ausführlich beschrieben. Kapitel 5 beleuchtet die Leistungsfähigkeit des Verfahrens, indem es automatisch erzeugte Prozessüberwachungssysteme mit Systemen vergleicht, die von Experten erstellt wurden. Die Arbeit schließt mit einer kurzen Zusammenfassung und einem Ausblick auf weitere lohnende Forschungsarbeiten.

Kapitel 2

Stand der Forschung

Dieses Kapitel beleuchtet den aktuellen Stand der für diese Arbeit relevanten Forschungsfelder. Zunächst soll ein Überblick über die maschinellen Lernverfahren gegeben werden. Diese Lernverfahren bilden den Kern der intelligenten Prozessüberwachungssysteme, die anschließend beschrieben werden. Es folgen Betrachtungen aktueller heuristischer Optimierungsverfahren und aktueller Verfahren für die Modellauswahl. Ausgehend von diesen Betrachtungen wird im letzten Abschnitt geschildert, inwieweit intelligente Prozessüberwachungssysteme bereits heute automatisch erzeugt werden können.

2.1 Überwachtes maschinelles Lernen

In der Wissenschaft und Technik verwendet man oft Modelle, um komplexe Sachverhalte darzustellen. Das gewählte Modell soll dabei einerseits die auftretenden Phänomene sinnvoll beschreiben. Andererseits soll es auch möglich sein, mit die-

sem Modell belastbare Vorhersagen zu treffen. Maschinelles Lernen hat das Ziel, zu einer gegebenen Menge von Daten automatisch ein Modell zu erzeugen, das diese Daten beschreibt. Der vorliegende Abschnitt beschäftigt sich speziell mit den überwachten maschinellen Lernverfahren. Dieser Typ von Lernverfahren ist weit verbreitet und kommt in zahlreichen intelligenten Prozessüberwachungssystemen zur Anwendung [Liang u. a. 2004, Abellan-Nebot u. a. 2010].

2.1.1 Grundbegriffe

Dem überwachten maschinellen Lernen liegt folgende Aufgabenstellung zugrunde: Gegeben ist eine Menge $\mathcal{X}$ von Eingangsvariablen. Diese Eingangsvariablen haben einen nicht näher bekannten Einfluss auf einen Ausgabewert. Man weiß, dass der Ausgabewert in der Menge $\mathcal{Y}$ liegt. Ziel des überwachten maschinellen Lernens ist es nun, ein Modell zu finden, das für jede mögliche Eingangsvariable den zugehörigen Ausgabewert vorhersagt. Der Wertebereich $\mathcal{Y}$ des Ausgabewertes kann dabei kontinuierlich oder diskret sein. Die Vorhersage kontinuierlicher Ausgabewerte nennt man *Regression*. Die Vorhersage diskreter Ausgabewerte heißt *Klassifikation*. Eine Klassifikation ordnet also einen Eingabewert einer von endlich vielen Klassen zu. Die Eingangsvariablen sind dabei nur selten skalar. Vielmehr bestehen sie meist aus mehreren Werten, die zu einem n-Tupel $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{X}$ zusammengefasst werden. Die Komponenten solch eines n-Tupels heißen *Kenngrößen* [Hastie u. a. 2009].

Das eigentliche Lernen findet in der sogenannten *Lernmaschine* statt. Die Lernmaschine ist ein Algorithmus, der ein geeignetes Lernmodell bestimmt und parametriert. Solch ein *Lernmodell* ist dabei eine Funktion $f : \mathcal{X} \to \mathcal{Y}$, die für jeden Eingangswert $\boldsymbol{x} \in \mathcal{X}$ einen Näherungswert des erwarteten Ausgabewertes $y \simeq f(\boldsymbol{x})$ liefert [Jankowski u. a. 2006, Hastie u. a. 2009].

Beim überwachten maschinellen Lernen wird der Lernmaschine eine Menge $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$ von Lerndatensätzen vorgegeben. Jeder dieser Datensätze $(\boldsymbol{x}, y) \in \mathcal{D}$ besteht dabei aus zwei Teilen: einer Eingangsvariablen $\boldsymbol{x}$ und dem zugehörigen Ausgabewert y. Auf Basis der Lerndatensätze sucht die Lernmaschine ein geeignetes Lernmodell f und passt dieses über dessen freie Parameter an die vorgegebenen Lerndaten an [Jankowski u. a. 2006].

Die Datensätze, die für die Erzeugung eines Lernmodells zur Verfügung gestellt werden, können üblicherweise nur einen Bruchteil der tatsächlich möglichen Eingangsdatensätze abdecken. Das Lernmodell muss daher auch für solche Eingabetupel sinnvolle Ergebnisse liefern, die nicht in den Lerndaten enthalten waren. Diese Fähigkeit nennt man *Verallgemeinerung* [Hastie u. a. 2009].

Die Verallgemeinerungsfähigkeit eines Modells ist ein wesentliches Maß für dessen Güte. Zur Berechnung dieser Güte wird zunächst eine *Schadensfunktion* $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_0^+$ festgelegt. Diese Schadensfunktion vergleicht eine vom Lernmodell f ermittelte Ist-Prognose $f(\boldsymbol{x})$ mit der zu $\boldsymbol{x}$ gehörigen Soll-Prognose y. Das Ergebnis dieser Funktion ist ein Fehlerwert $L(\, y, f(\boldsymbol{x})\,)$, der größer wird, je stärker $f(\boldsymbol{x})$ von y abweicht. Summiert man die Fehlerwerte über die Funktionswerte der Verteilungsfunktion P von $\mathcal{D}$ auf, so ergibt sich die Maßzahl

$$R\,[f] := \int L(\, y, f(\boldsymbol{x})\,)\, \mathrm{d}P(\boldsymbol{x}, y). \tag{2.1}$$

Man nennt $R\,[f]$ den *Verallgemeinerungsfehler* oder auch das *erwartete Risiko* von f. Ziel der Lernmaschine ist es, das Lernmodell f so auszuwählen und zu parametrieren, dass das zugehörige erwartete Risiko minimal wird.

In der Praxis ist die Verteilungsfunktion P nicht bekannt. Die Lernmaschine muss das erwartete Risiko daher abschätzen. Hierzu wählt sie t Datensätze $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_t, y_t) \in \mathcal{D}$ aus der Menge der vorgegeben Lerndaten aus. Auf Ba-

sis dieser Teilmenge der Lerndatensätze berechnet sie einen Näherungswert für das erwartete Risiko. Ein gängiger Näherungswert ist hierbei das *empirische Risiko*

$$R_{\text{emp}}[f] := \frac{1}{t}\sum_{i=1}^{t} L(\, y_i, f(\boldsymbol{x_i})\,). \tag{2.2}$$

Das empirische Risiko ist also der Mittelwert der Schadenswerte für die ausgewählten t Datensätze [Jankowski u. a. 2006].

2.1.2 Leistungsfähige Verfahren

Künstliche neuronale Netze waren lange Zeit das dominierende Lernmodell für überwachtes maschinelles Lernen. Auch heute noch kommen neuronale Netze in intelligenten Prozessüberwachungssystemen zum Einsatz [Abellan-Nebot u. a. 2010, Teti u. a. 2010]. In den letzten Jahren wurden allerdings neue leistungsfähige Lernmodelle entwickelt. Diese Lernmodelle schneiden in Leistungsvergleichen regelmäßig besser ab als neuronale Netze [Guyon u. a. 2006a, Lutz 2006, Guyon u. a. 2008a, Guyon u. a. 2008b]. Mit *LogitBoost* und den *Support Vector Machines* werden im Folgenden zwei dieser Verfahren beschrieben. Zudem wird mit dem *naiven Bayes-Klassifikator* ein einfaches und dennoch sehr leistungsfähiges Klassifikationsverfahren vorgestellt. Für die folgenden Betrachtungen sei die Ergebnismenge $\mathcal{Y} = \{c_1, \ldots, c_z\}$ jeweils diskret.

Naiver Bayes-Klassifikator

Der Ausgangspunkt für Bayes-Klassifikatoren ist der Satz von Bayes. Für eine gegebene Eingangsvariable $\boldsymbol{x} \in \mathcal{X}$ besagt der Satz von Bayes folgendes:

$$P(c_i|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|c_i)P(c_i)}{P(\boldsymbol{x})}. \tag{2.3}$$

Dabei ist $P(\boldsymbol{x})$ die Wahrscheinlichkeit für das Auftreten eines bestimmten Tupels $\boldsymbol{x}$. Der Wert $P(c_i)$ gibt die Wahrscheinlichkeit an, mit der ein beliebiger Eingabewert zur Klasse c_i gehört. Man nennt diese Wahrscheinlichkeit auch die *A-Priori-Wahrscheinlichkeit* von c_i. Die sogenannte *klassenbedingte Wahrscheinlichkeit* $P(\boldsymbol{x}|c_i)$ beschreibt die Wahrscheinlichkeit, mit welcher $\boldsymbol{x}$ in der Klasse c_i vorkommt. Aus den eben beschriebenen Wahrscheinlichkeiten lässt sich nun die *A-Posteriori-Wahrscheinlichkeit* $P(c_i|\boldsymbol{x})$ berechnen. Die A-Posteriori-Wahrscheinlichkeit ist die Wahrscheinlichkeit, mit der das Eingangstupel $\boldsymbol{x}$ zur Klasse c_i gehört.

Unter Verwendung des Satzes von Bayes kann man den sogenannten *optimalen Bayes-Klassifikator* $f_{\text{obc}} : \mathcal{X} \to \mathcal{Y}$ definieren durch

$$f_{\text{obc}}(\boldsymbol{x}) := \underset{c_i \in \mathcal{Y}}{\operatorname{argmax}} \; P(c_i|\boldsymbol{x}) \qquad \text{für alle } \boldsymbol{x} \in \mathcal{X}. \tag{2.4}$$

Dieser Klassifikator sucht für jedes Eingabetupel $\boldsymbol{x}$ die Klasse c_i, zu der $\boldsymbol{x}$ mit der größten Wahrscheinlichkeit gehört. Der optimale Bayes-Klassifikator würde für jede Klassifikationsaufgabe stets das bestmögliche Ergebnis liefern. Allerdings lässt sich das Verfahren in der Praxis nicht anwenden. Für die Berechnung der A-Posteriori-Wahrscheinlichkeiten müsste man alle Wahrscheinlichkeitsdichten kennen, die dem Problem zugrunde liegen. Diese Wahrscheinlichkeitsdichten sind in der Regel aber nicht bekannt [Jankowski u. a. 2006].

Ein Verfahren, das sich auch im praktischen Einsatz bewährt hat, ist der *naive Bayes-Klassifikator*. Dieser Klassifikator geht von der vereinfachenden Annahme aus, dass die einzelnen Kenngrößen der Eingangstupel linear unabhängig sind. Durch diese Annahme verliert der Klassifikator die Fähigkeit, stets das optimale Ergebnis zu liefern. Dafür vereinfacht sich die Berechnung der A-Posteriori-Wahrscheinlichkeit enorm. Man erhält:

$$
\begin{aligned}
f_{nbc}(\boldsymbol{x}) &= \underset{c_i \in \mathcal{Y}}{\operatorname{argmax}}\ P(c_i|\boldsymbol{x}) = \underset{c_i \in \mathcal{Y}}{\operatorname{argmax}}\ \frac{P(\boldsymbol{x}|c_i)P(c_i)}{P(\boldsymbol{x})} \\
&= \underset{c_i \in \mathcal{Y}}{\operatorname{argmax}}\ P(c_i) \cdot \prod_{j=1}^{n} P(x_j|c_i).
\end{aligned} \tag{2.5}
$$

Der naive Bayes-Klassifikator wird von einer Lernmaschine parametriert, indem diese aus den gegebenen Lerndatensätzen die Werte für die bedingten Wahrscheinlichkeiten $P(x_j|c_i)$ für alle möglichen Klassen $c_i \in \mathcal{Y}$ ermittelt.

Diskrete Kenngrößen x_j können nur endlich viele Werte $w_1, \ldots, w_s$ annehmen. Daher lassen sich die bedingten Wahrscheinlichkeiten in diesem Fall sehr einfach abschätzen. Die Lernmaschine setzt jede Einzelwahrscheinlichkeit $P(x_j = w_k|c_i)$ auf den Wert der relativen Häufigkeit, mit der x_j in der Klasse c_i den Wert w_k annimmt. Diese relativen Häufigkeiten sind besonders dann gute Abschätzungen, wenn die Menge $\mathcal{D}$ die tatsächliche Wahrscheinlichkeitsverteilung der Lerndaten widerspiegelt.

Kenngrößen mit kontinuierlichem Definitionsbereich können ebenfalls mittels relativer Häufigkeiten ausgewertet werden. Allerdings muss man in diesem Fall die Werte zunächst in diskrete Klassen einteilen. Alternativ kann man die Wahrscheinlichkeiten $P(x_j|c_i)$ durch Standard-Verteilungen annähern. Meist verwendet man in diesen Fällen die Normalverteilung N und wählt

$$
P(x_j|c_i) \sim N(\mu_j^{c_i}, \sigma_j^{c_i}). \tag{2.6}
$$

Die Werte $\mu_j^{c_i}$ bzw. $\sigma_j^{c_i}$ setzt man auf den Mittelwert bzw. die Standardabweichung der j-ten Kenngröße innerhalb der Lerndatensätze, die der Klasse c_i zugeordnet werden sollen [Jankowski u. a. 2006, Boullé 2006, Boullé 2009].

Die wesentlichen Vorteile des naiven Bayes-Klassifikators sind dessen Einfachheit und die vergleichsweise geringe Zeitkomplexität. Zudem zeigen Untersuchungen, dass das Verfahren für viele Klassifikationsaufgaben sehr gute Ergebnisse

liefert. Er empfiehlt sich insbesondere dann, wenn nur wenige Lerndatensätze zur Verfügung stehen [Hand u. a. 2001]. Der naive Bayes-Klassifikator wird auch auf dem Gebiet der Prozessüberwachung seit einigen Jahren erfolgreich eingesetzt [Elangovan u. a. 2010, Muralidharan u. a. 2012].

LogitBoost mit Entscheidungsbäumen

LogitBoost ist ein Klassifikationsverfahren aus der Gruppe der *Boosting-Algorithmen*. Beim Boosting wird eine Folge von k *Basisklassifikatoren* zu einem einzigen leistungsfähigen Klassifikator zusammengesetzt. Die so erzeugten Klassifikatoren liefern sehr gute Ergebnisse für eine Vielzahl von Klassifikationsaufgaben [Schapire u. a. 1999, Dettling u. a. 2003, Caruana u. a. 2006, Khoshgoftaar u. a. 2010].

Die Boosting-Algorithmen erzeugen ihre Basisklassifikatoren sukzessive. Dabei wird die Lerndatenmenge für jeden Erzeugungsvorgang so modifiziert, dass jeweils diejenigen Datensätze besonders gewichtet sind, die mit den aktuell vorhandenen Basisklassifikatoren falsch klassifiziert werden. Das Klassifikationsergebnis eines Boosting-Verfahrens ist die gewichtete Mehrheitsentscheidung all seiner Basisklassifikatoren [Friedman u. a. 2000, Guyon 2009].

Für die Gewichtung der falsch klassifizierten Datensätze gibt es zwei Varianten: Reweighting und Resampling. Beim *Reweighting* wird jedem Element der Datensatzmenge ein Gewicht zugeordnet, das in der Lernmaschine des Basisklassifikators ausgewertet wird. Im Gegensatz dazu wird beim *Resampling* jeweils eine neue Datensatzmenge erstellt, indem einzelne Datensätze der bisherigen Lerndatenmenge ausgewählt werden. Dabei werden Datensätze mit höherem Gewicht mit höherer Wahrscheinlichkeit ausgewählt [Seiffert u. a. 2008].

Das LogitBoost-Verfahren arbeitet bei der Klassifikation ähnlich wie die Bayes-Klassifikatoren. Bayes-Klassifikatoren ordnen eine Eingangsvariable $\boldsymbol{x} \in \mathcal{X}$ immer derjenigen Klasse $c_i \in \mathcal{Y}$ zu, in die sie am wahrscheinlichsten gehört. Auch das

LogitBoost-Verfahren arbeitet nach diesem Prinzip. Wie die Bayes-Klassifikatoren ermittelt auch LogitBoost für ein zu klassifizierendes $\boldsymbol{x} \in \mathcal{X}$ zunächst alle A-Posteriori-Wahrscheinlichkeiten $P(c_1|\boldsymbol{x}), \ldots, P(c_z|\boldsymbol{x})$. Als Ergebnis liefert LogitBoost dann ebenfalls die Klasse mit der höchsten Wahrscheinlichkeit. Was LogitBoost allerdings stark von den Bayes-Klassifikatoren unterscheidet, ist das Vorgehen bei der Ermittlung der benötigten A-Posteriori-Wahrscheinlichkeiten. Während die Bayes-Verfahren die A-Posteriori-Wahrscheinlichkeiten nach dem Satz von Bayes berechnen, schätzt LogitBoost diese Wahrscheinlichkeiten mithilfe seiner Basisklassifikatoren ab.

Sind die $b_j : \mathcal{X} \to \{\pm 1\}$ mit $j = 1, \ldots, k$ die erzeugten Basisklassifikatoren. Dann bildet LogitBoost z Funktionen $F_i : \mathcal{X} \to \mathbb{R}$, die gegeben sind durch

$$F_i(\boldsymbol{x}) := \sum_{j=1}^{k} \beta_{ij} b_j(\boldsymbol{x}) \qquad \text{mit } i = 1, \ldots, z. \tag{2.7}$$

Dabei ist F_i ein Maß für die Stärke der Zugehörigkeit zur Klasse c_i. Je größer $F_i(\boldsymbol{x})$ für ein $\boldsymbol{x} \in \mathcal{X}$, desto höher ist die Wahrscheinlichkeit, dass $\boldsymbol{x}$ zur Klasse c_i gehört. Hieraus kann man die gesuchten A-Posteriori-Wahrscheinlichkeiten ableiten. Die Wahrscheinlichkeit, dass ein gegebenes $\boldsymbol{x} \in \mathcal{X}$ zur Klasse c_i gehört ist

$$P(c_i|\boldsymbol{x}) = \frac{e^{F_i(\boldsymbol{x})}}{\sum_{j=1}^{z} e^{F_j(\boldsymbol{x})}}. \tag{2.8}$$

Das Lernmodell b muss vor der Erstellung eines Klassifikators vorgegeben werden. Für die Anpassung des LogitBoost-Klassifikators müssen die Gewichte β_{ij} bestimmt und die Basisklassifikatoren b_j erstellt werden. Dies geschieht mit einem iterativen Quasi-Newton-Verfahren [Friedman u. a. 2000, Bühlmann u. a. 2007, Li 2010].

Typische Basisklassifikatoren für Boosting-Verfahren sind *Entscheidungsbäume*. Solche Entscheidungsbäume setzen sich aus Knoten und Blättern zusam-

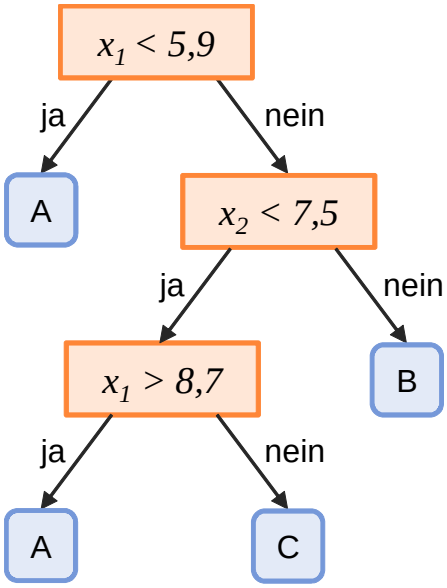


Abbildung 2.1: Beispielhafter Entscheidungsbaum für die Klassen A, B und C

men. Jeder Knoten entspricht einer Entscheidungsregel, die jeweils eine Kenngröße auswertet. Die Blätter repräsentieren eine Klassenzuordnung. Abbildung 2.1 zeigt beispielhaft solch einen Entscheidungsbaum. Mit einem gegebenen Entscheidungsbaum können Kenngrößentupel klassifiziert werden, indem man ausgehend vom Wurzelknoten entlang des Baumes abwärts geht. In jedem Knoten wird geprüft, ob eine bestimmte Kenngröße die entsprechende Regel erfüllt. Abhängig vom Ausgang dieser Prüfung wird dann das entsprechende Nachfolgeelement besucht. Das Verfahren endet, wenn man ein Blatt erreicht. Die Klasse, die dem erreichten Blatt zugeordnet ist, ist das Ergebnis der Klassifikation. Entscheidungsbäume werden auf Basis von Lerndatensätzen automatisch erzeugt. Hierfür gibt es mehrere effiziente Verfahren [Quinlan 1986, Jankowski u. a. 2006]. LogitBoost mit Entscheidungsbäumen als Basislerner gilt derzeit als eines der leistungsfähigsten Klassifikationsmodelle [Guyon 2009, Lutz 2006].

Support Vector Machines

Die *Support Vector Machines* sind Klassifikationsmodelle aus der Klasse der *Kernel-Verfahren*. Sie können in ihrer Urform lediglich auf Klassifikationsprobleme mit zwei Zielklassen angewendet werden. Das Verfahren lässt sich jedoch auch auf z Zielklassen ausweiten, indem man z Support Vector Machines kombiniert.

Die Grundidee der Support Vector Machines ist folgende: Für die gegebene Menge der Lerndatensätze wird ein mehrdimensionaler Hilbertraum ermittelt, in welchem diese Datensätze möglichst gut linear separierbar sind. Lineare Separierbarkeit bedeutet dabei, dass es eine Trennebene gibt, sodass links der Ebene die Datensätze der einen Klasse und rechts der Ebene die Datensätze der anderen Klasse liegen. Die Lernmaschine der Support Vector Machine platziert diese Trennebene so, dass zwischen der Ebene und den verwendeten Lerndatensätzen ein möglichst breiter Rand bleibt. Die Trennebene soll also möglichst mittig zwischen den Klassen liegen. Da der Hilbertraum eine beliebige Dimension haben kann, werden die Trennebenen als lineare Hyperebenen beschrieben. Für die spätere Klassifikation wird nun lediglich geprüft, auf welcher Seite der Ebene der zu klassifizierende Datensatz liegt. Die lineare Entscheidungsgrenze im Hilbertraum führt im Allgemeinen zu einer nicht-linearen Entscheidungsgrenze im ursprünglichen Eingangsraum. Dieses Prinzip ist in Abbildung 2.2 dargestellt [Schölkopf u. a. 2002].

Der zu verwendende Hilbertraum $\mathcal{H}$ ist nicht direkt definiert. Er wird durch die sogenannte *Kernel-Funktion* $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ induziert. Dies geschieht durch die Festlegung, dass für alle $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$ die Gleichung

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j) \tag{2.9}$$

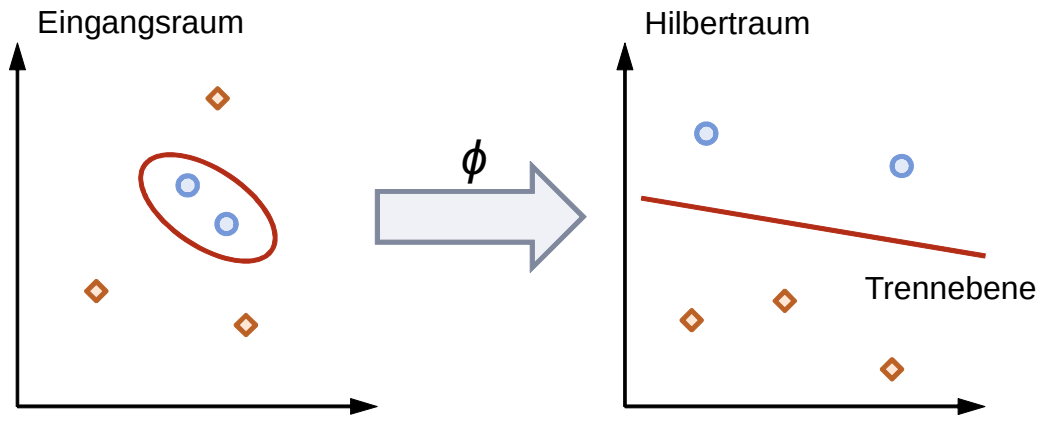


Abbildung 2.2: Grundprinzip der Support Vector Machines [Schölkopf u. a. 2002]

erfüllt sein muss. Der Wert $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ entspricht also dem Skalarprodukt der Bilder von $\boldsymbol{x}_i$ und $\boldsymbol{x}_j$ unter der implizit definierten Funktion $\phi : \mathcal{X} \to \mathcal{H}$ [Schölkopf u. a. 2002, Hofmann u. a. 2008].

Gängige Kernel-Funktionen sind der Gauß-Kernel K_g, der polynomiale Kernel K_p, der lineare Kernel K_l und der Sigmoid-Kernel K_s. Diese Kernel sind wie folgt definiert [Hofmann u. a. 2008, Hsu u. a. 2010]:

$$K_g(\boldsymbol{x}_i, \boldsymbol{x}_j) := e^{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \cdot \sigma^{-2}} \tag{2.10}$$

$$K_p(\boldsymbol{x}_i, \boldsymbol{x}_j) := (c + \boldsymbol{x}_i^T \boldsymbol{x}_j)^q \tag{2.11}$$

$$K_l(\boldsymbol{x}_i, \boldsymbol{x}_j) := \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{2.12}$$

$$K_s(\boldsymbol{x}_i, \boldsymbol{x}_j) := \tanh(\sigma \boldsymbol{x}_i^T \boldsymbol{x}_j + c) \tag{2.13}$$

Für die einzelnen Kernel sind dabei die jeweiligen Parameter σ, c und q problemabhängig zu wählen.

In einem Hilbertraum $\mathcal{H}$ kann eine Hyperebene durch einen Normalenvektor $\boldsymbol{w} \in \mathcal{H}$ und einen Offset $b \in \mathbb{R}$ definiert werden. Alle Punkte $\boldsymbol{p} \in \mathcal{H}$ in der Hyperebene erfüllen dann die Normalengleichung

$$\boldsymbol{w} \cdot \boldsymbol{p} + b = 0. \tag{2.14}$$

Der linke Term der Normalengleichung liefert einen Wert, der genau dann größer Null ist, wenn $\boldsymbol{p}$ auf der Ebenenseite liegt, in die $\boldsymbol{w}$ zeigt. Die Support Vector Machines nutzen diese Eigenschaft wie folgt: Sei $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ die gewählte Kernel-Funktion und $\phi : \mathcal{X} \to \mathcal{H}$ die dadurch induzierte Abbildung in den Hilbertraum. Sei die Trennebene in $\mathcal{H}$ durch $\boldsymbol{w} \in \mathcal{H}$ und $b \in \mathbb{R}$ gegeben, und sei das Urbild $\boldsymbol{w}' := \phi^{-1}(\boldsymbol{w})$ bekannt. Dann verwenden die Support Vector Machines die Vorhersagefunktion $f : \mathcal{X} \to \{+, -\}$, die für alle $\boldsymbol{x} \in \mathcal{X}$ definiert ist durch

$$\begin{aligned} f(\boldsymbol{x}) &:= \operatorname{sgn}(\boldsymbol{w} \cdot \phi(\boldsymbol{x}) + b) \\ &= \operatorname{sgn}(\phi(\boldsymbol{w}') \cdot \phi(\boldsymbol{x}) + b) \\ &= \operatorname{sgn}(K(\boldsymbol{w}', \boldsymbol{x}) + b). \end{aligned} \tag{2.15}$$

Intern wird dem erhaltenen Vorzeichen dann die entsprechende Klassenbezeichnung zugeordnet [Schölkopf u. a. 2002].

Um zu einer gegeben Menge $\mathcal{D} \subseteq \mathcal{X} \times \{+, -\}$ von Lerndaten die optimal trennende Hyperebene zu finden, muss der Lernalgorithmus ein Optimierungsproblem lösen. Für die optimal trennende Hyperebene wird die Länge des zugehörigen Normalenvektors $\boldsymbol{w}$ minimal. Die Zielfunktion $\tau : \mathcal{H} \to \mathbb{R}^+$ dieser Optimierung ist deshalb definiert durch

$$\tau(\boldsymbol{w}) := \frac{1}{2}\|\boldsymbol{w}\|^2 \qquad \text{für alle } \boldsymbol{w} \in \mathcal{H}. \tag{2.16}$$

Der Optimierungsalgorithmus löst dann das Problem

$$\min_{\boldsymbol{w}' \in \mathcal{X},\, b \in \mathbb{R}} \tau(\phi(\boldsymbol{w}')) \tag{2.17}$$

mit der Nebenbedingung

$$y(\phi(\boldsymbol{w}') \cdot \phi(\boldsymbol{x}) + b) = y(K(\boldsymbol{w}', \boldsymbol{x}) + b) \geq 1 \qquad \text{für alle } (\boldsymbol{x}, y) \in \mathcal{D}. \tag{2.18}$$

Da $\|\boldsymbol{w}\|^2 = \boldsymbol{w} \cdot \boldsymbol{w} = \phi(\boldsymbol{w}') \cdot \phi(\boldsymbol{w}') = K(\boldsymbol{w}', \boldsymbol{w}')$ gilt, können die Funktionswerte von τ in Gleichung 2.17 auch ohne direkte Kenntnis von ϕ berechnet werden.

In der Praxis kommt es vor, dass sich die zu separierenden Datenmengen überlappen. In diesem Fall kann es keine Trennebene geben. Daher wird die Nebenbedingung aus Gleichung 2.18 gelockert. Man erlaubt für jeden der m Lerndatensätze $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m) \in \mathcal{D}$ eine Verletzung dieser Nebenbedingung, bestraft diese Verletzung aber mit einem Fehlerwert ξ_i. Als neue Nebenbedingung ergibt sich so

$$y_i(K(\boldsymbol{w}', \boldsymbol{x}_i) + b) \geq 1 - \xi_i \qquad \text{für alle } (\boldsymbol{x}_i, y_i) \in \mathcal{D},\ i = 1, \ldots, m. \tag{2.19}$$

Die Abweichungen werden bestraft, indem die Summe der Fehlerwerte in die Zielfunktion einfließt. Die Zielfunktion ist nun $\tau : \mathcal{H} \times \mathbb{R}^m \to \mathbb{R}^+$, definiert durch

$$\tau(\boldsymbol{w}, \boldsymbol{\xi}) := \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{m} \xi_i \qquad \text{für alle } \boldsymbol{w} \in \mathcal{H}, \boldsymbol{\xi} \in \mathbb{R}^m. \tag{2.20}$$

Die Wahl eines geeigneten $\boldsymbol{\xi} \in \mathbb{R}^m$ ist dabei ebenfalls Gegenstand der Optimierung. Der Faktor $C > 0$ ist ein Parameter, mit dem sich die Gewichtung der Fehlersumme einstellen lässt. Dieser Parameter gibt der Klasse von Support Vector Machines, die als Zielfunktion die Gleichung 2.20 verwenden, ihren Namen. Man nennt diese *C-Support Vector Machines*, oder kurz C-SVMs [Schölkopf u. a. 2002, Chen u. a. 2005].

Eine zweite Möglichkeit zur Behandlung von Abweichungen ist die ν-Parametrierung. Hier wird der Parameter C durch einen Parameter $\nu \in (0, 1]$ ersetzt. Zusätzlich wird der Randparameter $\rho \in \mathbb{R}$ eingeführt, der selbst eine Variable des

Optimierungsproblems ist. Mit diesen Parametern ergibt sich die Optimierungsaufgabe

$$\min_{\boldsymbol{w}' \in \mathcal{X}, \boldsymbol{\xi} \in \mathbb{R}^m \; b, \rho \in \mathbb{R}} \tau(\phi(\boldsymbol{w}'), \boldsymbol{\xi}, \rho) := \frac{1}{2}\|\phi(\boldsymbol{w}')\|^2 - \nu\rho + \frac{1}{2}\sum_{i=1}^{m} \xi_i \tag{2.21}$$

mit der Nebenbedingung

$$y_i(K(\boldsymbol{w}', \boldsymbol{x}_i) + b) \geq \rho - \xi_i \qquad \text{für alle } (\boldsymbol{x}_i, y_i) \in \mathcal{D},\ i = 1, \ldots, m. \tag{2.22}$$

Der in der Zielfunktion verwendete Parameter ν gibt den entsprechenden Verfahren den Namen *ν-Support Vector Machines* (ν-SVMs) [Schölkopf u. a. 2002, Chen u. a. 2005].

2.2 Intelligente Prozessüberwachung

Unter *intelligenten Prozessüberwachungssystemen* sollen im Folgenden Prozessüberwachungssysteme verstanden werden, die Lernmodelle verwenden, um den aktuellen Prozesszustand aus den verfügbaren Maschinensignalen abzuleiten. Die intelligenten Prozessüberwachungssysteme gelten als sehr leistungsstark und flexibel [Abellan-Nebot u. a. 2010, Kitazawa u. a. 2011].

2.2.1 Allgemeiner Aufbau

Die derzeit verfügbaren intelligenten Prozessüberwachungssysteme haben vorwiegend den in Abbildung 2.3 dargestellten Grundaufbau. In diesem Aufbau werden zunächst physikalische Größen des überwachten Prozesses sensorisch erfasst. Die so gewonnenen Signale werden dann zur Auswertung an ein Software-System übergeben. Die folgenden Betrachtungen beschränken sich auf die Verarbeitungsschritte innerhalb dieses Software-Systems. Daher bezieht sich die Bezeichnung *Prozessüberwachungssystem* stets auf dieses Software-System.

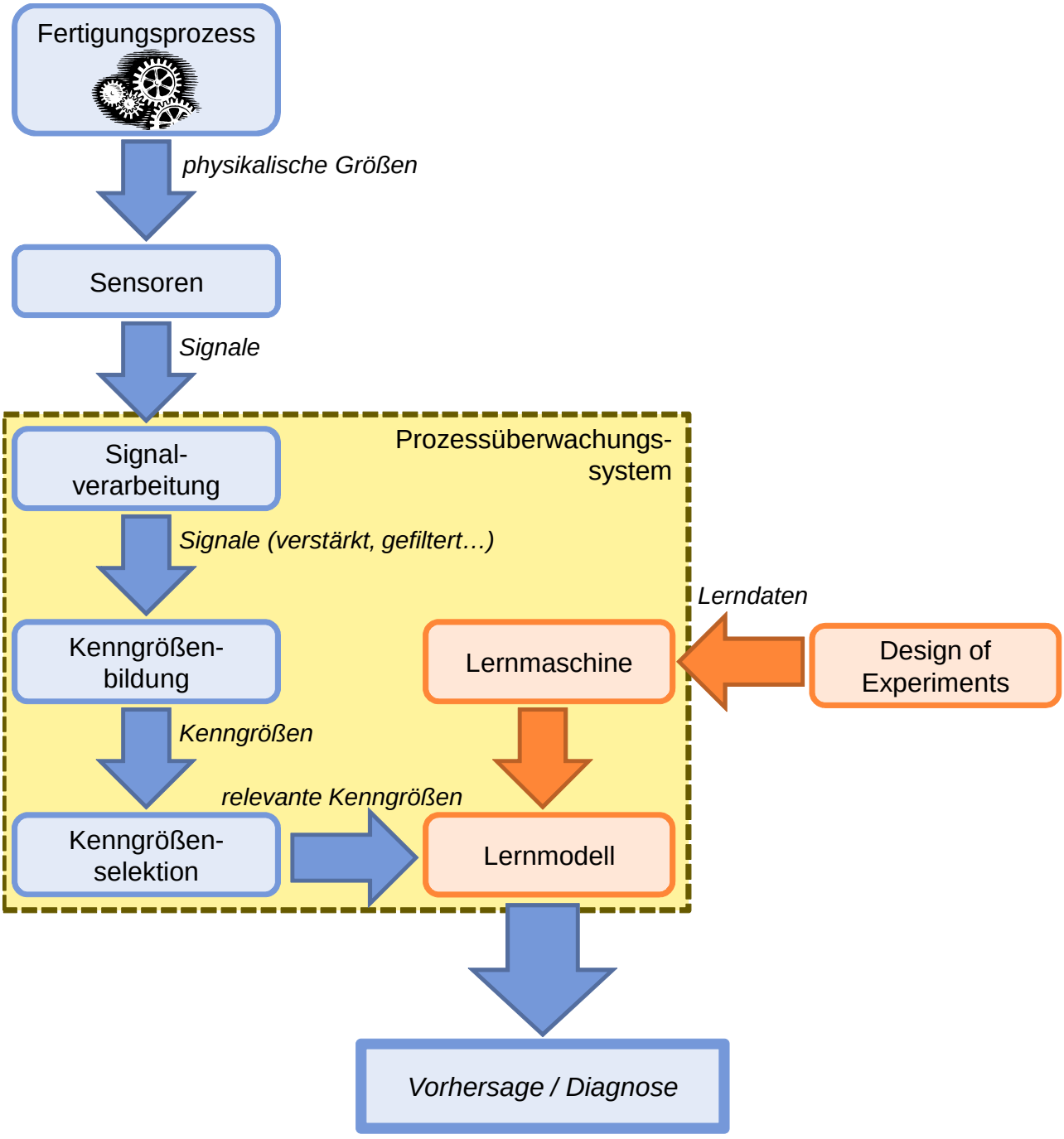


Abbildung 2.3: Aufbau intelligenter Prozessüberwachungssysteme [Abellan-Nebot u. a. 2010]

Das Prozessüberwachungssystem bereitet die Signaldaten zunächst mit Methoden der digitalen Signalverarbeitung auf. Im Rahmen der *Kenngrößenbildung* werden aus den aufbereiteten Signalen die relevanten Informationen extrahiert und durch numerische Werte beschrieben. Diese numerischen Werte heißen *Kenngrößen*. Anschließend werden in der *Kenngrößenselektion* aus allen gebildeten Kenngrößen die aussagekräftigsten ausgewählt. Die ausgewählten Kenngrößen werden zu einem n-Tupel zusammengefasst und an das Prozessmodell übergeben. Das Prozessmodell ordnet diesem Kenngrößentupel dann eine Prognose oder eine Diagnose zu [Abellan-Nebot u. a. 2010].

2.2.2 Funktionsweise der Teilkomponenten

Signalverarbeitung

Sensordaten, die in der industriellen Fertigung erfasst werden, sind in der Regel mit mechanischen, akustischen oder elektrischen Störungen behaftet. Aus diesem Grund müssen die erfassten Signale vor der Analyse meist noch aufbereitet werden [Ghosh u. a. 2007]. Innerhalb des Überwachungssystems liegen die Signale in digitaler Form vor. Hier wird nun mit digitalen Filtern versucht, möglichst alle Störungen aus den Signalen zu entfernen. Dabei kommen vor allem klassische Filter wie Bandpässe [Chen u. a. 2007, Jemielniak 2000] oder Median-Filter zum Einsatz [Doymaz u. a. 2001, Jardine u. a. 2006].

Ein weiterer möglicher Schritt der Signalverarbeitung ist die Signalsegmentierung. Hierbei werden aus den Signalen die tatsächlich relevanten Bereiche extrahiert. Die Daten außerhalb dieser relevanten Bereiche werden verworfen. So wird die Menge der Sensordaten reduziert. Ein wesentlicher Nachteil der Signalsegmentierung ist allerdings, dass dadurch eine spätere Analyse der Signale im Frequenz- oder Zeit-Frequenz-Bereich unmöglich wird [Abellan-Nebot u. a. 2010].

Kenngrößenbildung

Die vom Überwachungssystem auszuwertenden Eingangsdaten tragen Informationen über den überwachten Prozess. Bei der Kenngrößenbildung werden diese Informationen extrahiert und durch aussagekräftige Maßzahlen – die Kenngrößen – beschrieben. Die Eingangsdaten können dabei skalare Werte oder Signale sein. Skalare Werte, wie z. B. Umgebungstemperatur, Luftfeuchtigkeit und Zykluszeit, können direkt als Kenngrößen verwendet werden [Jardine u. a. 2006]. Bei Signalen ist die Kenngrößenbildung dagegen aufwändiger. Sie liegen in Form von Zeitreihen vor, die den zeitlichen Verlauf einer gemessenen Größe beschreiben [Schlittgen u. a. 2001]. So können beispielweise Vibrationen und akustische Signale dargestellt werden oder auch Temperatur- und Druckverläufe in einem Spritzgießwerkzeug.

In einer Zeitreihe sind oft nicht einzelne Werte relevant; vielmehr liegen die gesuchten Informationen im zeitlichen Verlauf der Merkmalswerte verborgen. Zur Kenngrößenbildung muss die Zeitreihe deshalb zusammenhängend betrachtet werden. Hierzu gibt es zahlreiche Methoden [Abellan-Nebot u. a. 2010, Jemielniak u. a. 2012]. Abhängig vom Anwendungsfall liefern einige dieser Methoden sinnvolle Kenngrößen, andere nicht. Die Auswahl geeigneter Methoden setzt daher Expertenwissen und ein tiefes Verständnis des überwachten Prozesses voraus [Jardine u. a. 2006].

Die Methoden zur Erzeugung von Kenngrößen aus Zeitreihen können in drei Klassen aufgeteilt werden: Analyse im Zeitbereich, Analyse im Frequenzbereich und Analyse im Zeit-Frequenzbereich [Jardine u. a. 2006, Teti u. a. 2010].

Analyse im Zeitbereich: Gängige Kenngrößen, die im Zeitbereich berechnet werden, sind u. a. Mittelwerte, Standardabweichungen, Minima und Maxima, Integrale und Steigungen sowie Minimum-Maximum-Abstände und Minimum-Maximum-Amplituden [Ertekin u. a. 2003, Teti u. a. 2010].

Komplexere Überwachungssysteme setzen auch Zeitserienmodelle ein. Hierbei werden parametrisierbare Zeitserienmodelle an die zu untersuchende Zeitserie angepasst. Die Kenngrößen werden dann aus den Parameterwerten der angepassten Modelle berechnet. In der Prozessüberwachung werden häufig die Zeitserienmodelle *Auto Regressive (AR)* und *Auto Regressive Moving Average (ARMA)* eingesetzt [Pöyhönen u. a. 2004, Elbestawi u. a. 2006].

Über diese Modelle hinaus finden sich in der Literatur noch zahlreiche weitere Verfahren zur Kenngrößenbildung im Zeitbereich. Viele dieser Verfahren sind speziell auf den überwachten Prozess zugeschnitten [Abellan-Nebot u. a. 2010, Teti u. a. 2010].

Analyse im Frequenzbereich: Die Analyse im Frequenzbereich ermöglicht die Betrachtung spezieller Komponenten des Frequenzspektrums. Hierzu muss die Zeitserie zunächst in den Frequenzbereich übertragen werden. Die gängigste Methode hierfür ist die schnelle Fourier-Transformation (FFT). Im Rahmen einer Spektrumsanalyse wird entweder das ganze Spektrum betrachtet, oder es werden einzelne Frequenzkomponenten genauer untersucht. Die Kenngrößen werden dann auf Basis der betrachteten Frequenzkomponenten gebildet [Abellan-Nebot u. a. 2010, Teti u. a. 2010].

Ein gebräuchliches Verfahren für die Frequenzanalyse ist die Auswertung des Leistungsspektrums. Das Leistungsspektrum liefert den jeweiligen Anteil der auf die Frequenz bezogenen Leistung des Signals in vorgegebenen Frequenzbändern [Press 2007]. Einige Überwachungssysteme verwenden auch alternative Verfahren, die speziell auf den jeweiligen Anwendungsfall zugeschnittene Kenngrößen liefern [Verl u. a. 2009, Jardine u. a. 2006].

Zeit-Frequenzbereich: Analysen im Frequenzbereich sind nur dann sinnvoll, wenn die im Signal vorkommenden Frequenzen über die Zeit hinweg kon-

stant bleiben. Diese Voraussetzung ist aber bei der Analyse von Maschinendaten oft nicht gegeben. In diesen Fällen bietet sich die Analyse im Zeit-Frequenzbereich an. Zur Überführung der Zeitreihen in den Zeit-Frequenzbereich wird dabei fast immer die diskrete Wavelet-Transformation verwendet [Jardine u. a. 2006, Mandrikova u. a. 2011].

Im Anschluss an die diskrete Wavelet-Transformation können die gesuchten Kenngrößen abgeleitet werden. Hierbei werden die berechneten Wavelet-Koeffizienten oft direkt als Kenngrößen verwendet. In einigen Systemen werden die Kenngrößen aber auch durch statistische Auswertungen (z. B. Mittelwert, Varianz, Maximalwert) dieser Koeffizienten ermittelt [Zhu u. a. 2009, Abellan-Nebot u. a. 2010].

Vergleichende Untersuchungen zeigen, dass sich mit vielen einfachen Kenngrößen in der Regel die besten Vorhersageergebnisse erzielen lassen. Vorhersagen auf Basis komplexer und mit Expertenwissen erzeugter Kenngrößen liefern dagegen weniger gute Resultate [Guyon 2009].

Kenngrößenselektion

Die Zahl der Kenngrößen, die aus einem oder mehreren Signalen erzeugt werden, kann sehr groß sein. Viele dieser Kenngrößen stehen aber in keinem Zusammenhang mit der vorherzusagenden Größe. Eine Kenngrößenselektion wählt aus der Menge aller Kenngrößen eine Teilmenge aus. Diese Teilmenge soll nur diejenigen Kenngrößen enthalten, die tatsächlich Hinweise auf die Zielgröße geben [Duch 2006, Guyon u. a. 2003].

Vor allem traditionelle Lernmodelle wie neuronale Netze und Bayes-Klassifikatoren profitieren von einer Kenngrößenselektion. Die meisten neuronalen Netze können nur sehr schlecht verallgemeinern, wenn die Dimension der Eingangsdaten

viel größer ist als die Anzahl der Lerndatensätze. Somit kann eine Kenngrößenselektion die Anzahl der benötigten Lerndatensätze erheblich reduzieren [Abellan-Nebot u. a. 2010]. Auch der naive Bayes-Klassifikator profitiert von der Kenngrößenselektion, da die reduzierten Kenngrößensätze der angenommenen linearen Unabhängigkeit zwischen den einzelnen Kenngrößen oft recht nahe kommen [Hand u. a. 2001].

Die Verfahren zur Kenngrößenselektion können in zwei Klassen eingeteilt werden: Filter und Wrapper. Beide Klassen nutzen Lerndatensätze, um die relevanten Kenngrößen zu bestimmen. Sie unterscheiden sich allerdings hinsichtlich der Bewertung der selektierten Kenngrößenmengen [Kohavi u. a. 1997].

Die *Wrapper*-Verfahren bewerten eine gewählte Menge von Kenngrößen anhand der Prognosegüte des später verwendeten Lernmodells. Das Lernmodell wird auf Basis der bereitgestellten Lerndatensätze erzeugt. Dabei werden nur die Kenngrößen berücksichtigt, die in der gewählten Kenngrößenmenge enthalten sind. Somit muss für die Bewertung jeder Kenngrößenmenge ein eigenes Lernmodell generiert werden. Die Erzeugung von Lernmodellen ist üblicherweise rechenintensiv. Daher benötigen Wrapper-Verfahren für die Auswahl einer geeigneten Kenngrößenmenge meist sehr lange. Dies gilt insbesondere dann, wenn die Datensätze eine hohe Dimension haben [Kohavi u. a. 1997, Ferreira u. a. 2012].

Filtermethoden bewerten die Kenngrößenmengen unabhängig vom verwendeten Lernmodell mittels Relevanzfunktionen. Sei $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$ die Menge der Lerndatensätze, sei $g : \mathcal{X} \to \mathbb{R}^n$ die Funktion, die aus den Eingangsdaten die Kenngrößen erzeugt, und sei $\mathcal{K} := \{1, 2, \ldots, n\}$ die Menge der Indices dieser Kenngrößen. Dann liefert eine Relevanzfunktion $R_{\mathcal{D},g} : \mathcal{P}(\mathcal{K}) \to \mathbb{R}$ eine Maßzahl für die Stärke des Einflusses einer Kenngrößenteilmenge $S \subseteq \mathcal{K}$ auf die vorherzusagende Zielgröße. Die Kenngrößen werden abhängig von der berechneten Relevanz ausgewählt.

Filtermethoden haben eine geringe Zeitkomplexität und liefern gleichzeitig gute Ergebnisse [Liu u. a. 2005, Duch 2006, Abellan-Nebot u. a. 2010].

Das einfachste Filter-Verfahren ist das *Ranking*. Hier wird die Relevanz für jede Kenngröße $k \in \mathcal{K}$ einzeln berechnet. Anschließend werden die Kenngrößen nach ihrer Relevanz sortiert: $R_{\mathcal{D},g}(\{k_1\}) \geq R_{\mathcal{D},g}(\{k_2\}) \geq \cdots \geq R_{\mathcal{D},g}(\{k_n\})$. Die Kenngrößen mit geringer Relevanz werden verworfen. Trotz ihrer Einfachheit erzielen Ranker in der Praxis oft gute Ergebnisse [Duch 2006, Guyon 2008].

Da Ranker die Kenngrößen meist isoliert betrachten, können sie Datenredundanzen schwer erkennen. Die Vorwärtsselektion und die Rückwärtselimination sind alternative Verfahren, die Kenngrößensätze mit geringer Redundanz bevorzugen.

Die *Vorwärtsselektion* beginnt mit einer leeren Kenngrößenmenge $S = \emptyset$. Anschließend fügt sie dieser Menge iterativ einzelne Kenngrößen hinzu. In jedem Iterationsschritt sucht die Vorwärtsselektion diejenige Kenngröße $s \in \mathcal{K} \setminus S$ in der Menge der noch nicht selektierten Kenngrößen, für die die Menge $S \cup \{s\}$ den größten Relevanzwert $R_{\mathcal{D},g}(S \cup \{s\})$ liefert. Ist dieser Relevanzwert größer als die Relevanz der bisherigen Kenngrößenmenge S, dann wird s zu S hinzugefügt. Andernfalls bricht die Vorwärtsselektion ab und liefert S als Ergebnis. Die Vorwärtsselektion fügt also solange Kenngrößen zu S hinzu, bis sich die Relevanz der Kenngrößenmenge nicht mehr verbessert.

Die *Rückwärtselimination* wählt zunächst alle Kenngrößen und setzt $S = \mathcal{K}$. Dann entfernt sie iterativ einzelne Kenngrößen aus dieser Menge. Hierzu sucht die Rückwärtselimination in jedem Iterationsschritt eine bisher selektierte Kenngröße $s \in S$, sodass der Relevanzwert $R_{\mathcal{D},g}(S \setminus \{s\})$ maximal wird. Ist dieser Relevanzwert nicht kleiner als die Relevanz der bisherigen Menge S, dann wird s aus S entfernt. Andernfalls bricht die Rückwärtselimination ab und liefert S als

Ergebnis. Die Rückwärtselimination entfernt also Kenngrößen, solange dies ohne Verschlechterung der Relevanz möglich ist [Guyon u. a. 2006b, Guyon 2008].

Nachfolgend werden einige Relevanzfunktionen vorgestellt, die auch in intelligenten Prozessüberwachungssystemen Einsatz finden. Die verbreitetste Relevanzfunktion basiert auf dem linearen Korrelationskoeffizienten. Dieser Koeffizient kann auf Basis der auszuwertenden Lerndaten empirisch berechnet werden. Der Korrelationskoeffizient eignet sich aber lediglich für die Bewertung einzelner Kenngrößen. Ist $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$ die Menge der Lerndaten und ist $g : \mathcal{X} \to \mathbb{R}^n$ die Funktion für die Kenngrößenerzeugung, dann ist die zugehörige *korrelationsbasierte Relevanzfunktion* $\varrho_{\mathcal{D},g} : \mathcal{K} \to \mathbb{R}$ definiert durch

$$\varrho_{\mathcal{D},g}(k) := \frac{\sum_{i=1}^{m}(\, g(\boldsymbol{x}_i)_k - \bar{g}(\boldsymbol{x})_k\,) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m}(\, g(\boldsymbol{x}_i)_k - \bar{g}(\boldsymbol{x})_k\,)^2 \cdot \sum_{i=1}^{m}(y_i - \bar{y})^2}} \qquad \text{für alle } k \in \mathcal{K}. \quad (2.23)$$

Dabei sind $\bar{g}(\boldsymbol{x})_k$ und $\bar{y}$ die Mittelwerte der Kenngröße k bzw. der Zielgröße. Der Ausdruck $g(\boldsymbol{x}_i)_k$ liefert die Kenngröße k aus dem i-ten Lerndatensatz [Duch 2006, Teti u. a. 2010].

Die Relevanzfunktion von Hall ist eine Erweiterung der korrelationsbasierten Relevanzfunktion auf mehrere Kenngrößen. Bei der Berechnung der Relevanz der übergebenen Kenngrößenmenge wird sowohl die Korrelation der einzelnen Kenngrößen mit der Zielgröße als auch die Korrelation der einzelnen Kenngrößen untereinander berücksichtigt. Die höchsten Relevanzwerte ergeben sich, wenn alle Kenngrößen stark mit der Zielgröße korrelieren, die Korrelation zwischen den einzelnen Kenngrößen aber gering ist [Hall 1999].

Eine weitere Relevanzfunktion für eine einzelne Kenngröße ist die *Relief-Funktion* $\beta_{\mathcal{D},g} : \mathcal{K} \to \mathbb{R}$. Diese Funktion wertet für jeden Lerndatensatz auch dessen nächste Nachbarn im Kenngrößenraum aus. Sie ist definiert durch

$$\beta_{\mathcal{D},g}(k) := \frac{\sum_{i=1}^{m}\sum_{j=1}^{c} |g(\boldsymbol{x}_i)_k - g(\boldsymbol{x}_{M_{i,j}})_k|}{\sum_{i=1}^{m}\sum_{j=1}^{c} |g(\boldsymbol{x}_i)_k - g(\boldsymbol{x}_{H_{i,j}})_k|} \qquad \text{für alle } k \in \mathcal{K}. \quad (2.24)$$

Die Relief-Funktion benötigt zu jedem Lerndatensatz $(\boldsymbol{x}_i, y_i) \in \mathcal{D}$ die c nächsten Eingabetupel $H_{i,1}, H_{i,2}, \ldots, H_{i,c} \in \mathcal{X}$ aus der Menge der Lerndaten, deren erwartetes Ergebnis ebenfalls y_i ist. Zudem benötigt man die c nächsten Eingabetupel $M_{i,1}, M_{i,2}, \ldots, M_{i,c} \in \mathcal{X}$ aus der Menge der Lerndaten, die ein anderes erwartetes Ergebnis als y_i haben [Kononenko 1994, Guyon 2008].

Die *1R-Relevanzfunktion* nutzt den simplen 1R-Klassifikator zur Abschätzung der Relevanz einzelner Kenngrößen. Der 1R-Klassifikator ist ein Entscheidungsbaum, der Datensätze abhängig vom Wert einer einzelnen Kenngröße klassifiziert. Zur Berechnung der Relevanz werden die Lerndaten in eine Menge von Trainingsdatensätzen und eine Menge von Verifikationsdatensätzen aufgeteilt. Die Relevanz der betrachteten Kenngröße ergibt sich aus der Prognosegüte des 1R-Klassifikators, der mit den Trainingsdatensätzen erzeugt wurde [Holte 1993, Hall 1999].

Die *konsistenzbasierte Relevanzfunktion* definiert ein Maß für die Konsistenz von Datensätzen. Zwei Datensätze gelten hier als inkonsistent, wenn sie in allen ausgewählten Kenngrößen übereinstimmen, die erwarteten Prognoseergebnisse $y \in \mathcal{Y}$ sich aber unterscheiden. Stimmen n Datensätze in allen Kenngrößen überein, dann wird ermittelt, welches Prognoseergebnis $y_{\max} \in \mathcal{Y}$ diesen n Datensätzen am häufigsten zugeordnet ist. Ist $c_{\max}$ ein Zähler, der angibt, wie oft das häufigste Prognoseergebnis $y_{\max}$ in den n Datensätzen auftritt, so wird für diese n Datensätze eine Inkonsistenzzahl $z_i = n - c_{\max}$ berechnet. Die Relevanz des zu bewertenden Kenngrößensatzes ist dann die Summe aller Inkonsistenzzahlen geteilt durch die Anzahl der Lerndatensätze [Liu u. a. 1996].

Lernmodelle

Abellan-Nebot und Romero Subirón betrachten in einer umfassenden Literaturrecherche intelligente Prozessüberwachungssysteme, die im Zeitraum von

2002 bis 2007 entwickelt wurden. Dabei zeigt sich, dass mehr als zwei Drittel dieser Systeme zur Modellbildung neuronale Netze oder Neuro-Fuzzy-Netze verwenden [Abellan-Nebot u. a. 2010]. Übereinstimmend stellen auch Jardine, Lin und Banjevic sowie Teti, Jemielniak, O'Donnell und Dornfeld fest, dass neuronale Netze auf dem Gebiet der Maschinenüberwachung derzeit noch das vorherrschende Lernmodell sind [Jardine u. a. 2006, Teti u. a. 2010].

Allerdings zeigt Ribeiro, dass Support Vector Machines bei der Qualitätsüberwachung von Spritzgießprozessen bessere Ergebnisse und eine bessere Laufzeit erzielen als neuronale Netze [Ribeiro 2005]. In neueren Überwachungssystemen werden auch zunehmend kernel-basierte Lernmodelle eingesetzt [Cho u. a. 2005, Lei Guo u. a. 2009, Kim u. a. 2011, Qu u. a. 2011, Matić u. a. 2012, Demetgul 2012].

Besonders bemerkenswert ist, dass alle hier betrachteten Überwachungssysteme mit einem festen Typen von Lernmodellen arbeiten. Dem Autor ist kein Überwachungssystem bekannt, das das Lernmodell abhängig vom Umfang und der Gestalt der verfügbaren Lerndaten auswählt.

2.3 Heuristische Optimierungsverfahren

Optimierungsverfahren suchen die beste Lösung für eine vorgegebene Problemstellung. Viele Optimierungsaufgaben sind aber nicht analytisch lösbar. In diesen Fällen bedient man sich heuristischer Verfahren, die Näherungslösungen für solche Probleme ermitteln [Michalewicz u. a. 2004].

Mit der Partikelschwarm-Optimierung und der Artificial-Bee-Colony-Optimierung werden zunächst zwei Verfahren zur Bestimmung globaler Extremwerte von Funktionen vorgestellt. Dem schließt sich die Beschreibung der genetischen Pro-

grammierung an. Hierbei handelt es sich um eine Methode zur automatischen Erzeugung von Computer-Programmen, die ein vorgegebenes Problem lösen.

2.3.1 Partikelschwarm-Optimierung

Die *Partikelschwarm-Optimierung* ist ein Verfahren zur Bestimmung des globalen Maximums einer Funktion $f : \mathbb{R}^n \to \mathbb{R}$. Die Optimierung wird dabei erreicht, indem das Sozialverhalten eines Fisch- oder Vogelschwarms simuliert wird. Innerhalb solch eines Schwarms kann sich jedes Individuum nach eigenem Willen bewegen. Gleichzeitig wird die Bewegung eines Individuums aber auch vom Verhalten der anderen Schwarmmitglieder beeinflusst. Durch diese Gruppendynamik entsteht die sogenannte Schwarmintelligenz. Diese hält den Schwarm trotz unterschiedlicher individueller Interessen zusammen [Kennedy u. a. 1995].

Die Partikelschwarm-Optimierung simuliert das Schwarmverhalten, indem sie einen Schwarm durch den Lösungsraum $\mathbb{R}^n$ bewegt. Innerhalb dieses Lösungsraumes ist jede Position eine mögliche Lösung des zugrunde liegenden Optimierungsproblems. Der Schwarm wird so gesteuert, dass er als Ganzes danach strebt, die Position $\boldsymbol{x} \in \mathbb{R}^n$ zu finden, für die f den maximalen Wert annimmt [Kameyama 2009].

Das Verfahren arbeitet wie folgt: Zunächst werden Partikel $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \mathbb{R}^n$ zufällig im Suchraum platziert. Anschließend werden die einzelnen Partikel in Bewegung gesetzt. Hierzu wird jedem Partikel $\boldsymbol{x}_i$ ein zufällig gewählter Geschwindigkeitsvektor $\boldsymbol{v}_i$ zugeordnet. Dieser Geschwindigkeitsvektor bestimmt neben der momentanen Geschwindigkeit auch die Richtung, in die sich der Partikel im Lösungsraum bewegt. Im Anschluss an diese Initialisierung wird das gesuchte Maximum iterativ ermittelt. Dabei werden in jedem Iterationsschritt zunächst die Geschwindigkeitsvektoren aller Partikel neu berechnet. Anschließend werden die Partikel abhängig von ihrem jeweiligen Geschwindigkeitsvektor bewegt. Die neue

Position des i-ten Partikels ist nun $\boldsymbol{x}_i + \boldsymbol{v}_i$. Am Ende jedes Iterationsschrittes werden für alle Partikelpositionen die zugehörigen Funktionswerte $f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_k)$ berechnet. Die Optimierung endet, wenn einer der berechneten Funktionswerte ein vorgegebenes Abbruchkriterium erfüllt [Poli 2008, Pedersen u. a. 2010].

Die Schwarmintelligenz wird bei der Partikelschwarm-Optimierung durch die Berechnung der Geschwindigkeitsvektoren realisiert. Diese Berechnung ist so gestaltet, dass der jeweilige Partikel zunächst seine ursprüngliche Bewegungsrichtung beibehalten möchte. Gleichzeitig wird der Partikel von zwei elastischen Kräften beeinflusst. Eine dieser Kräfte zieht ihn mit zufällig gewählter Stärke in die Richtung der besten Position, die der Partikel bisher im Lösungsraum gefunden hat. Die zweite Kraft zieht ihn – ebenfalls mit zufälliger Stärke – in die Richtung der global besten Position, die über alle Schwarmpartikel hinweg bisher gefunden wurde. Die Geschwindigkeit $\boldsymbol{v}_i$ des Partikels $\boldsymbol{x}_i$ im Iterationsschritt $t+1$ lässt sich somit durch folgende Formel ausdrücken:

$$\boldsymbol{v}_i[t+1] = \omega \boldsymbol{v}_i[t] + \psi_1 R_1 \left(\hat{\boldsymbol{x}}_s - \boldsymbol{x}_i[t]\right) + \psi_2 R_2 \left(\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i[t]\right). \tag{2.25}$$

Dabei ist ω das sogenannte *Trägheitsgewicht*, das beeinflusst, wie stark die bisherige Geschwindigkeit mit in die neue Geschwindigkeit eingeht. Die Variable $\hat{\boldsymbol{x}}_s$ enthält die beste Position, die dem Schwarm bisher bekannt ist. Die Variable $\hat{\boldsymbol{x}}_i$ enthält die beste Position, die bisher vom aktuellen Partikel selbst besucht wurde. Die Parameter ψ_1 und ψ_2 bestimmen zusammen mit den Zufallszahlen R_1 und R_2, wie stark die beiden Kräfte jeweils berücksichtigt werden [Poli 2008, Kennedy u. a. 1995]. Zusammenfassend lässt sich die Partikelschwarm-Optimierung durch den Algorithmus 1 beschreiben [Elbeltagi u. a. 2005].

Gegenüber vielen anderen Optimierungsverfahren hat die Partikelschwarm-Optimierung wesentliche Vorteile: Das Verfahren konvergiert schnell und liefert für viele Probleme sehr gute Ergebnisse. Zudem lässt sich der Algorithmus leicht implementieren, und er hat wenige Parameter, die zu setzen sind. Wegen dieser

Algorithmus 1 : Partikelschwarm-Optimierung

```
// Initialisierung
Wähle zufällige Startpositionen x_1, x_2, ..., x_k ∈ R^n;
Wähle zufällige Startgeschwindigkeiten v_1, v_2, ..., v_k ∈ R^n;

Setze die beste Position x̂_i = x_i für jeden Partikel i = 1, 2, ..., k;
Setze die beste Position des Schwarms x̂_s = argmax_{x_i} f(x_i);

// Iteration
while Abbruchkriterium nicht erfüllt do
    for i = 1 to k do
        Wähle Zufallszahlen R_1, R_2;
        Setze v_i = ω v_i + ψ_1 R_1 (x̂_s − x_i) + ψ_2 R_2 (x̂_i − x_i);
        Setze die neue Position x_i = x_i + v_i;
        if f(x_i) > f(x̂_i) then
            Setze x̂_i = x_i;
        end
        if f(x_i) > f(x̂_s) then
            Setze x̂_s = x_i;
        end
    end
end
```

positiven Eigenschaften hat sich das Verfahren schnell verbreitet. Es kommt derzeit in vielen technischen und nicht-technischen Anwendungen zum Einsatz [Cui u. a. 2005, Poli 2008, Elbeltagi u. a. 2005].

2.3.2 Artificial-Bee-Colony-Optimierung

Die *Artificial-Bee-Colony-Optimierung* ist ebenfalls ein Algorithmus zur Bestimmung des globalen Maximums einer Funktion $f : \mathbb{R}^n \to \mathbb{R}$. Auch die Artificial-

Bee-Colony-Optimierung macht sich das Konzept der Schwarmintelligenz zunutze. Allerdings wird hier das Zusammenwirken der Einzelindividuen wesentlich differenzierter betrachtet als bei der Partikelschwarm-Optimierung.

Bei der Artificial-Bee-Colony-Optimierung wird die Futtersuche eines Bienenvolkes simuliert. Die erfolgreiche Strategie der Futtersuche realer Honigbienen basiert im Wesentlichen auf folgenden Verhaltensformen [Karaboga 2005]:

Kommunikation: Bienen teilen ihre Informationen über Futterquellen mit den anderen Bienen durch den Bienentanz.

Positives Feedback: Je größer die Nektarmenge einer Futterquelle ist, desto größer ist die Wahrscheinlichkeit, dass auch weitere Bienen dorthin fliegen.

Negatives Feedback: Futterquellen mit geringer Nektarmenge werden aufgegeben.

Zufällige Streuung: Späher schwärmen in willkürlich gewählte Richtungen aus und suchen dort nach neuen Futterquellen.

Um dieses Verhalten nachzubilden, betrachtet die Artificial-Bee-Colony-Optimierung die möglichen Lösungen des Optimierungsproblems als Futterquellen. Einer Futterquelle $\boldsymbol{x} \in \mathbb{R}^n$ wird deren Funktionswert $f(\boldsymbol{x})$ als Nektarmenge zugeordnet. Außerdem unterscheidet der Algorithmus drei Gruppen von Bienen: beschäftigte Arbeiterbienen, unbeschäftigte Arbeiterbienen und Späher. Die beschäftigten Arbeiterbienen haben bereits eine Futterquelle entdeckt und beuten diese aus. Die unbeschäftigten Arbeiterbienen warten zunächst im Bienenstock auf Informationen von den beschäftigten Arbeiterbienen, die auf lohnende Futterquellen hinweisen. Die Späher schwärmen ohne Rückmeldung von anderen Bienen aus und suchen willkürlich nach Futterquellen. Algorithmus 2 beschreibt das Zusammenwirken der einzelnen Bienen-Typen [Karaboga 2005, Karaboga u. a. 2007].

Algorithmus 2 : Artificial-Bee-Colony-Optimierung

Wähle zufällige Futterquellen $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k \in \mathbb{R}^n$;
Setze Zähler $t_1 = t_2 = \cdots = t_k = 0$;

while *Abbruchkriterium nicht erfüllt* **do**
  Ermittle beste Lösung $\boldsymbol{x}_{\text{best}} = \text{argmax}_{\boldsymbol{x} \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, \boldsymbol{x}_{\text{best}}\}} f(\boldsymbol{x})$;
  for $i = 1$ **to** k **do** `// beschäftigte Arbeiterbienen senden`
    Wähle zufällig Futterquelle $\boldsymbol{x}_j \in \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k\}$;
    Besuche neue Futterquelle $\boldsymbol{x}'_i = \frac{1}{2}(\boldsymbol{x}_i + \boldsymbol{x}_j)$;
    if $f(\boldsymbol{x}'_i) > f(\boldsymbol{x}_i)$ **then**
      Setze $\boldsymbol{x}_i = \boldsymbol{x}'_i$; Setze $t_i = 0$;
    else
      Erhöhe t_i um 1;
    end
  end
  Berechne Besuchswahrscheinlichkeiten $p_i = \frac{f(\boldsymbol{x}_i)}{\sum_{j=1}^{k} f(\boldsymbol{x}_j)}$ für $i = 1, \ldots, k$;
  for $i = 1$ **to** $\frac{k}{2}$ **do** `// unbeschäftigte Arbeiterbienen senden`
    Wähle zufällig $\boldsymbol{x}_r$ abhängig von den Wahrscheinlichkeiten $p_1, \ldots, p_k$;
    Wähle zufällig Futterquelle $\boldsymbol{x}_j \in \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k\}$;
    Besuche neue Futterquelle $\boldsymbol{x}'_r = \frac{1}{2}(\boldsymbol{x}_r + \boldsymbol{x}_j)$;
    if $f(\boldsymbol{x}'_r) > f(\boldsymbol{x}_r)$ **then**
      Setze $\boldsymbol{x}_r = \boldsymbol{x}'_r$; Setze $t_r = 0$;
    else
      Erhöhe t_r um 1;
    end
  end
  for $i = 1$ **to** k **do** `// Späher aussenden`
    if $t_i > maxNumTrials$ **then**
      Ersetze $\boldsymbol{x}_i$ durch zufällige Futterquelle $\boldsymbol{x} \in \mathbb{R}^n$; Setze $t_i = 0$;
    end
  end
end

Bei der Artificial-Bee-Colony-Optimierung wird eine Menge $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k\}$ von momentan besuchten Futterquellen verwaltet. Iterativ wird dann zunächst das Aussenden der beschäftigten Arbeiterbienen simuliert. Hierzu werden die Umgebungen der momentan verwendeten Futterquellen durchsucht, indem man sich ausgehend von der jeweiligen Futterquelle in Richtung einer beliebigen anderen Futterquelle bewegt.

Hat die neue Futterquelle eine größere Nektarmenge als die bisherige, so wird die neue Futterquelle verwendet. Die bisherige Futterquelle wird in diesem Fall verworfen. Anschließend wird für jede Futterquelle die Wahrscheinlichkeit berechnet, mit der sie von einer bisher unbeschäftigten Arbeiterbiene besucht werden soll. Diese Wahrscheinlichkeit ist abhängig von der Nektarmenge der Futterquelle. Nach der Berechnung wird die Hälfte der Futterquellen unter Berücksichtigung dieser Wahrscheinlichkeiten nach dem Zufallsprinzip ausgewählt. Die ausgewählten Futterquellen werden von den unbeschäftigten Arbeiterbienen besucht. Hierbei wird die Nachbarschaft dieser Futterquellen analog zum Besuch der beschäftigten Arbeiterbienen ausgewertet.

Für jede Futterquelle wird mitgezählt, wie oft sie bereits von einer Arbeiterbiene besucht wurde, ohne dass in der Nachbarschaft eine bessere Futterquelle gefunden wurde. Nach einer vorgegebenen Anzahl von vergeblichen Besuchen wird die entsprechende Futterquelle verlassen und durch eine zufällig gewählte Futterquelle ersetzt. So wird das Verhalten der Späher simuliert [Karaboga 2005, Karaboga u. a. 2007].

Die Artificial-Bee-Colony-Optimierung findet in der Wissenschaft aktuell große Beachtung [Narasimhan 2009, Alatas 2010, Subotic u. a. 2010, Akay u. a. 2012, Parpinelli u. a. 2011, Banharnsakuna u. a. 2011]. Sie gilt als sehr leistungsfähig und schneidet in Vergleichstests sehr gut ab [Karaboga u. a. 2007, Teodorović 2009, Karaboga u. a. 2009].

2.3.3 Genetische Programmierung

Die *genetische Programmierung* ist eine Methode zur automatisierten Erzeugung von Computer-Programmen. Das Verfahren basiert auf der von Charles Darwin beschriebenen biologischen Evolution [Poli u. a. 2008]. Darwin beobachtete, dass die Natur einen Überschuss an Lebewesen produziert, die Populationen der einzelnen Spezies aber dennoch nicht wachsen. Er folgerte, dass viele Nachkommen sterben bevor sie sich fortpflanzen. Darwin wusste, dass Lebewesen einer Spezies unterschiedliche Merkmale aufweisen und diese an ihre Nachkommen vererben. Auf dieser Basis stellte er die These auf, dass sich gerade diejenigen Individuen fortpflanzen, die aufgrund ihrer speziellen Merkmale am besten mit ihren Lebensbedingungen zurechtkommen. Bei der Fortpflanzung vererben die Individuen dabei ihre Merkmale. Somit führt die natürliche Selektion dazu, dass sich eine Spezies im Laufe der Zeit an ihre Umgebung anpasst [Darvin 1872].

Algorithmus 3 : Genetische Programmierung

Erzeuge aus den Primitiven eine zufällige Population von Programmen;

repeat

  Führe jedes Programm aus und bewerte dessen Fitness;
  Wähle ein oder zwei Programme aus mit einer Wahrscheinlichkeit, die deren Fitness entspricht;
  Wende eine genetische Operation auf die ausgewählten Programme an;

until *Abbruchkriterium ist erfüllt*;

Das Programm mit der höchsten Fitness ist das gesuchte Ergebnis.

Die genetische Programmierung stellt die biologische Evolution nach, indem sie zu einer vorgegebenen Problemstellung eine Population von Computer-Programmen verwaltet. Jede Generation von Programmen erzeugt eine Menge von Nachkommen. Dabei werden Merkmale der Elterngeneration an die Kindgenera-

tion vererbt. Aus der Menge der Eltern und der Nachkommen werden diejenigen Individuen ausgewählt, die die neue Population bilden sollen. Das Verfahren endet, wenn ein Programm der aktuellen Population eine bestimmte Güte erreicht hat (siehe Algorithmus 3). Im Kontext der genetischen Programmierung wird diese zu bestimmende Güte die *Fitness* des Individuums genannt [Poli u. a. 2008].

Die genetische Programmierung bewertet ein Programm, indem sie es ausführt und das erhaltene Ergebnis mit einem vorgegebenen Soll-Ergebnis vergleicht. Je besser das Ist-Ergebnis und das Soll-Ergebnis übereinstimmen, desto höher ist der Fitness-Wert des entsprechenden Programms. Die Programme mit hohen Fitnesswerten werden mit höherer Wahrscheinlichkeit für die Fortpflanzung ausgewählt. Die ausgewählten Programme produzieren nun Nachwuchs für die nächste Generation. Gewöhnlich wird dieser Nachwuchs mithilfe zweier genetischer Operationen erzeugt: der Kreuzung und der Mutation. Bei der *Kreuzung* wird ein Kind-Programm erzeugt, indem zufällig gewählte Teile zweier Eltern-Programme kombiniert werden. Die *Mutation* erzeugt ein Kind-Programm durch das Verändern eines einzelnen Eltern-Programms an einer zufällig gewählten Stelle [Poli u. a. 2008, Espejo u. a. 2010].

Bei der genetischen Programmierung werden die Programme üblicherweise in Form von Syntaxbäumen dargestellt. Abbildung 2.4 zeigt solch einen Syntaxbaum für das Programm

$$\min(5 \cdot x + y,\ x + x).$$

Die Variablen und Konstanten eines Programms (im Beispielprogramm: x, y und 5) sind dabei die Blätter des Syntaxbaumes. Man nennt diese Blätter auch *Terminale*. Die arithmetischen Operationen (im Beispielprogramm: $+$, $\cdot$ und min) bilden die internen Knoten des Baumes. Diese Operationen nennt man die *Funktionen*. Terminale und Funktionen zusammen heißen *Primitive* [Poli u. a. 2008].

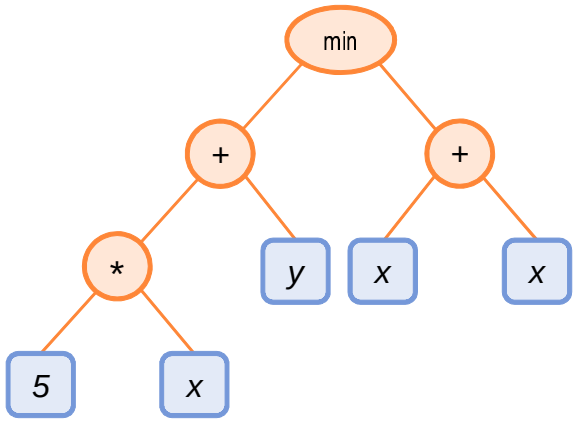


Abbildung 2.4: Beispielhafter Operatorbaum

Die Kreuzung und die Mutation basieren ebenfalls auf der Syntaxbaum-Darstellung. Die am häufigsten verwendete Form der Kreuzung ist die *Teilbaum-Kreuzung*. Hierbei wird in jedem der beiden Elternbäume ein zufälliger Knoten ausgewählt. Anschließend wird der Kindbaum erzeugt, indem der Teilbaum unterhalb des gewählten Knotens eines Elternteils durch den Teilbaum unterhalb des gewählten Knotens des anderen Elternteils ersetzt wird. Die meistgenutzte Form der Mutation wählt ebenfalls einen zufälligen Knoten innerhalb des einen Eltern-Baums. Der Teilbaum unterhalb des gewählten Knotens wird dann abgeschnitten und durch einen zufällig erzeugten Teilbaum ersetzt [Poli u. a. 2008].

2.4 Modellauswahl

Wie bereits in Abschnitt 2.1.1 beschrieben, findet beim maschinellen Lernen der eigentliche Lernvorgang in der Lernmaschine statt. Im Folgenden wird der Ablauf dieses Lernvorganges beschrieben.

2.4.1 Grundbegriffe

Die Bestimmung des besten Lernmodells für eine vorgegebene Menge von Datensätzen nennt man *Modellauswahl*. Die Modellauswahl im engeren Sinne bezieht sich lediglich auf die Auswahl geeigneter Parameter und Hyperparameter für ein bereits bestimmtes Lernmodell. Die Hyperparameter beschreiben die Konfiguration des anzupassenden Lernmodells. Bei einem Kernel-Verfahren können die Hyperparameter beispielsweise den Typen des gewählten Kernels festlegen, bei einem Boosting-Verfahren die Anzahl und den Typen der schwachen Lerner. Die Parameter umfassen die Werte, mit denen das konfigurierte Modell an die Lerndatensätze angepasst werden kann [Kearns u. a. 1997, Cawley u. a. 2007b, Salibian-Barrera u. a. 2008].

Einige neuere Veröffentlichungen haben eine allgemeinere Sichtweise, die auch in der vorliegenden Arbeit Anwendung finden soll. Hier geht man davon aus, dass zusätzlich zur Menge der Lernverfahren auch jeweils eine Menge von Verfahren für die Vorverarbeitung, die Kenngrößenerzeugung und die Kenngrößenselektion gegeben ist. Die Aufgabe der Modellauswahl ist es dann, diese Verfahren so zusammenzustellen, dass der Verallgemeinerungsfehler des Gesamtsystems minimal wird. Die Konfiguration des Gesamtsystems wird dabei ebenfalls durch Parameter und Hyperparameter beschrieben [Guyon 2009, Guyon u. a. 2010].

Zur formalen Darstellung des Modellauswahl-Problems wählt man eine Parameter-Hyperparameter-Notation. Für $k, n \in \mathbb{N}$ sei dabei $\boldsymbol{\alpha} \in \mathbb{R}^k$ der Vektor der Parameter des Lernmodells, $\boldsymbol{\theta} \in \mathbb{R}^n$ der Vektor der Hyperparameter. Dann repräsentiert die entsprechend parametrierte Funktion

$$f_{\boldsymbol{\alpha},\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y} \tag{2.26}$$

das gesamte Lernmodell [Guyon 2009].

Die Aufteilung in Parameter und Hyperparameter ermöglicht es, die Hyperparameter in einem Optimierungsverfahren zu modifizieren und gleichzeitig die Standard-Lernmaschine des jeweiligen Lernmodells für die Parameterauswahl zu verwenden [Guyon 2009, Cawley u. a. 2010].

2.4.2 Problem der Überanpassung

Über die Jahre wurden zahlreiche Verfahren zur Modellauswahl entwickelt. Diese Verfahren beherzigen zumeist ein Grundprinzip, das von dem englischen Wissenschaftler William von Ockham bereits im 14. Jahrhundert angewandt wurde. Dieses Prinzip ist unter dem Namen „Ockhams Rasierer“ bekannt. Es schreibt vor, dass Modelle immer so einfach wie möglich gehalten werden sollen. Komplexe Lernmodelle tendieren beim maschinellen Lernen nämlich zur *Überanpassung*. Bei einer Überanpassung wird das Lernmodell so stark an die Lerndaten angepasst, dass es auf den Lerndaten zwar sehr gute Ergebnisse liefert, seine Generalisierungsfähigkeit für unbekannte Daten aber verliert. Die Vermeidung von Überanpassung ist ein grundlegendes Problem der Modellauswahl [Wei Chu 2006, Guyon 2009].

Die Idee von Ockhams Rasierer findet auch in der klassischen Statistik Anwendung. Hier lässt sich das Prinzip formell beschreiben. Ein Lernmodell $f : \mathcal{X} \to \mathcal{Y}$ kann als Schätzfunktion für den Erwartungswert $E(y|\boldsymbol{x})$ betrachtet werden. Die Genauigkeit einer Schätzfunktion wird durch deren mittleren quadratischen Fehler ausgedrückt. Im Folgenden bezeichne $f_{\mathcal{D}}$ ein Lernmodell, dass unter Verwendung einer Menge $\mathcal{D}$ von Lerndatensätzen ausgewählt wurde. Die Menge $\mathcal{D}$ umfasse dabei m Datensätze. Außerdem bezeichne $E_{\mathcal{D}}(\cdot)$ den Erwartungswert über alle

möglichen Datensatzmengen $D \subseteq \mathcal{D}$ der Größe m. Dann lässt sich der mittlere quadratische Fehler $\delta_{\mathcal{D}}$ der Funktion $f_{\mathcal{D}}$ berechnen durch [Geman u. a. 1992]

$$\begin{aligned} \delta_{\mathcal{D}} &= E_{\mathcal{D}}((f_{\mathcal{D}}(\boldsymbol{x}) - E(y|\boldsymbol{x}))^2) \\ &= (\, E_{\mathcal{D}}(f_{\mathcal{D}}(\boldsymbol{x})) - E(y|\boldsymbol{x})\,)^2 + E_{\mathcal{D}}((f_{\mathcal{D}}(\boldsymbol{x}) - E_{\mathcal{D}}(f_{\mathcal{D}}(\boldsymbol{x})))^2). \end{aligned} \quad (2.27)$$

Der erste Term aus Gleichung 2.27 heißt *Schätzfehler* (auf englisch: Bias). Eine Schätzfunktion ist also dann mit einem Schätzfehler behaftet, wenn der Durchschnittswert $E_{\mathcal{D}}(f_{\mathcal{D}}(\boldsymbol{x}))$ ihrer Vorhersagen vom Sollwert $E(y|\boldsymbol{x})$ abweicht. Der zweite Term ist die *Varianz*. Diese Komponente gibt an, wie stark die Schätzfunktion um ihren Erwartungswert streut. Oft haben Modelle mit geringem Schätzfehler eine hohe Varianz und umgekehrt. Bei der Modellauswahl muss daher meist zwischen dem Schätzfehler und der Varianz abgewogen werden. Diese Abwägung ist als *Bias/Variance Tradeoff* bekannt [Bartlett u. a. 2002, Guyon 2009].

2.4.3 Bewertung von Lernmodellen

Die Modellauswahl ist ein Optimierungsproblem. Man sucht Parameter $\boldsymbol{\alpha} \in \mathbb{R}^k$ und Hyperparameter $\boldsymbol{\theta} \in \mathbb{R}^n$, sodass für eine Schadensfunktion $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_0^+$ das erwartete Risiko

$$R\,[f_{\boldsymbol{\alpha},\boldsymbol{\theta}}] := \int L(\, y, f_{\boldsymbol{\alpha},\boldsymbol{\theta}}(\boldsymbol{x})\,)\; dP(\boldsymbol{x}, y) \quad (2.28)$$

des entsprechend konfigurierten Lernmodells $f_{\boldsymbol{\alpha},\boldsymbol{\theta}}$ minimal wird [Guyon 2009].

Dieses Optimierungsproblem kann mit heuristischen Optimierungsverfahren gelöst werden. Die Zielfunktion dieser Optimierung wäre dann das erwartete Risiko. Allerdings wurde bereits in Abschnitt 2.1.1 dargelegt, dass die Berechnung des erwarteten Risikos im Allgemeinen nicht möglich ist. Man muss das Risiko daher mit Lerndatensätzen abschätzen [Molinaro 2005, Guyon 2009].

Eine bereits vorgestellte Näherungsfunktion für das erwartete Risiko ist das *empirische Risiko* $R_{\text{emp}}[f]$. Zur Berechnung des empirischen Risikos wählt man t Datensätze $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_t, y_t) \in \mathcal{D}$ aus der Menge der vorgegeben Lerndaten aus und berechnet den Mittelwert der Schadenswerte für diese Datensätze. Als Näherung des erwarteten Risikos aus Gleichung 2.28 ergibt sich damit [Jankowski u. a. 2006]

$$R_{\text{emp}}[f_{\boldsymbol{\alpha},\boldsymbol{\theta}}] := \frac{1}{t}\sum_{i=1}^{t} L(\, y_i, f(\boldsymbol{x_i})\,). \tag{2.29}$$

Das empirische Risiko gilt allerdings als sehr grobe Abschätzung. Vor allem im Falle einer Überanpassung kann es große Abweichungen zwischen $R_{\text{emp}}[f]$ und $R[f]$ geben. Dieser Abweichung kann man entgegenwirken, indem man der Risikoberechnung eine Komponente hinzufügt, die komplexe Lernmodelle bestraft. Man erhält so das *strukturelle Risiko*

$$R_{\text{struct}}[f_{\boldsymbol{\alpha},\boldsymbol{\theta}}] := R_{\text{emp}}[f_{\boldsymbol{\alpha},\boldsymbol{\theta}}] + p(\boldsymbol{\theta}). \tag{2.30}$$

Die Funktion $p : \mathbb{R}^n \to \mathbb{R}$ ist dabei ein Maß für die Komplexität des Lernmodells, das durch die Hyperparameter $\boldsymbol{\theta}$ beschrieben wird. Durch die Addition dieses Komplexitätswertes werden einfachere Modelle bevorzugt.

Eine einfach zu implementierende Variante des strukturellen Risikos ist die *Regularisierung*. Hierbei wird für das jeweilige Lernmodell ein Regularisierer definiert. Dies ist gewöhnlich die Norm $\|\boldsymbol{a}\|$ eines Parametervektors $\boldsymbol{a}$, der intern im jeweiligen Lernmodell verwendet wird. Man erhält dann das *regularisierte empirische Risiko*

$$R_{\text{reg}}[f_{\boldsymbol{\alpha},\boldsymbol{\theta}}] := R_{\text{emp}}[f_{\boldsymbol{\alpha},\boldsymbol{\theta}}] + \gamma\|\boldsymbol{a}|. \tag{2.31}$$

In obiger Gleichung ist $\gamma \in \mathbb{R}^+$ der sogenannte Regularisierungsparameter, mit dem sich der Bias/Variance-Tradeoff beeinflussen lässt. Die Wahl eines geeigneten Wertes für γ ist allerdings schwierig. Die Festlegung des Wertes erfolgt gewöhnlich

gemeinsam mit der Wahl der Hyperparameter [Bousquet u. a. 2002, Bousquet u. a. 2004, Guyon 2009].

2.4.4 Validierung

Ein gewähltes Lernmodell wird üblicherweise unter Verwendung von Testdaten bewertet. Hierzu werden einige der Lerndatensätze als Testdaten reserviert. Die reservierten Testdaten werden nicht für die Modellauswahl verwendet. Sie dienen einzig und allein der Bewertung des jeweiligen Lernmodells. Im Folgenden sei m die Anzahl der Datensätze in der Menge $\mathcal{D}$ der Lerndatensätze und s sei die Anzahl der Datentupel, die zur Modellauswahl verwendet werden. Dann stehen für die Bewertung des Modells noch $t = m - s$ Datensätze zur Verfügung. In der Praxis werden normalerweise 10% bis 30% der Daten als Testdaten reserviert [Guyon 2009].

Lernmodelle, die auf Basis einer einfachen Validierung ausgewählt werden, haben meist hohe Varianzen. Diese Varianzen lassen sich reduzieren, wenn die Modellbewertung auf eine breitere Datenbasis gestellt wird. Eine Möglichkeit hierfür ist das Sammeln zusätzlicher Lerndatensätze. Dies ist aber oft mit hohem Aufwand verbunden. Eine andere Möglichkeit ist der Einsatz einer *Kreuzvalidierung*. Die Kreuzvalidierung ist ein Verfahren zur Verbesserung der Modellbewertung. Die am häufigsten verwendete Form ist die k-fache Kreuzvalidierung. Hierbei werden die verfügbaren Lerndatensätze in k paarweise disjunkte Mengen aufgeteilt. In jeder Iteration wird das Modell mit einer anderen Kombination aus $k-1$ Teilmengen trainiert. Die nicht verwendete Teilmenge wird für die Risiko-Abschätzung verwendet. Die Gesamtbewertung des Lernmodells ist dann der Durchschnitt der k Einzelbewertungen [Bengio u. a. 2004, Cawley u. a. 2007a].

Ein Spezialfall der Kreuzvalidierung ist die *k-fache stratifizierte Kreuzvalidierung*. Bei der stratifizierten Kreuzvalidierung werden die Lerndatensätze so aufge-

teilt, dass jede der k Teilmengen annähernd die gleiche Klassenverteilung aufweist. Dadurch wird die Varianz der Abschätzung nochmals deutlich verringert [Guyon 2009].

Wegen ihrer Einfachheit und Effektivität ist die Kreuzvalidierung das verbreitetste Verfahren zur Modellvalidierung [Arlot u. a. 2010, Cawley u. a. 2010]. Mittlerweile gibt es schnellere Validierungsverfahren. Diese liefern allerdings meist weniger belastbare Bewertungen als die Kreuzvalidierung [Varewyck u. a. 2011].

2.4.5 Suchstrategien

Neben einer geeigneten Bewertungsmethode benötigt man für die Modellauswahl eine Suchstrategie, mit welcher der Parameter- und Hyperparameter-Raum durchsucht werden soll. Erfolgversprechend ist hier vor allem die *zweistufige Optimierung*. Bei diesem Verfahren werden zwei Inferenzstufen unterschieden. Auf Stufe 1 wird der Parametervektor $\boldsymbol{\alpha}$ angepasst, auf Stufe 2 der Hyperparameter-Vektor $\boldsymbol{\theta}$. Die Wahl der Hyperparameter nennt man *Modellkonfiguration*, die Wahl der Parameter heißt *Training* [Bennett u. a. 2006, Cawley u. a. 2010, Özöğür-Akyüz u. a. 2011].

Für die Modellkonfiguration sind heuristische Optimierungsverfahren sehr gut geeignet. Escalante, Montes und Sucar erzielen mit einer Partikelschwarm-Optimierung gute Resultate. Allgemein zeigt sich, dass populationsbasierte Optimierungsverfahren robust gegenüber der Überanpassung sind [Escalante u. a. 2009, Escalante u. a. 2010].

Auch Gorissen, Dhaene und De Turck verwenden ein heuristisches Optimierungsverfahren. Der von ihnen verwendete genetische Algorithmus ist aber zum einen in der Implementierung wesentlich komplexer als die populationsbasierten Optimierungsverfahren. Zum anderen ist das Verfahren so unflexibel, dass das

Einbinden zusätzlicher Lernmodelle mit hohem Aufwand verbunden ist [Gorissen u. a. 2009, Escalante u. a. 2009].

Einige neuere Verfahren ermöglichen eine Vorauswahl des Lernmodells anhand der Charakteristik der Eingangsdaten. Hierbei werden Regelwerke definiert, mit denen auf Basis struktureller und statistischer Merkmale der Lerndatensätze ein oder mehrere Lernmodelle vorgeschlagen werden. Solch eine Vorauswahl verkleinert den Hyperparameter-Raum und kann so die Modellauswahl beschleunigen [Guyon 2009, Song u. a. 2012, Reif u. a. 2014].

2.5 Ansätze zur automatisierten Erzeugung von Prozessüberwachungssystemen

Dieser Abschnitt beleuchtet den aktuellen Stand der Wissenschaft auf dem Gebiet der automatisierten Erzeugung und Konfiguration von Prozessüberwachungssystemen. Insgesamt scheint dieses Forschungsfeld bisher lediglich punktuell bearbeitet worden zu sein. So ist dem Autor bisher kein Verfahren bekannt, welches ein Prozessüberwachungssystem automatisch erzeugen kann.

Auch auf anderen Themenfeldern der digitalen Signalanalyse gibt es kaum Arbeiten zur automatischen Konfiguration entsprechender Systeme. Als einzige einschlägige Veröffentlichung ist hier [Iswandy u. a. 2009] bekannt. Dort wird die weitgehend automatisierte Konfiguration eines integrierten Multisensor-Systems für Embedded-Anwendungen beschrieben. Dieses Multisensor-System soll auch Klassifikationsaufgaben erfüllen können. Das zugehörige PC-basierte Konfigurationssystem stellt für die Signalvorverarbeitung, die Kenngrößenbildung, die Kenngrößenselektion und die Klassifikation jeweils eine Menge ausgewählter Methoden bereit. In einer Hardware-in-the-loop-Simulation werden im Rahmen eines Optimierungsverfahrens (Genetischer Algorithmus) die passenden Methoden ausge-

wählt und parametriert [Iswandy u. a. 2009]. Aufgrund der stark eingeschränkten Möglichkeiten bei der Kenngrößenbildung lässt sich dieses Verfahren nicht auf die automatisierte Erzeugung von Prozessüberwachungssystemen übertragen. Für die Anpassung eines Überwachungssystems an verschiedengestaltige Fertigungsprozesse werden für die Kenngrößenerzeugung wesentlich flexiblere Methoden benötigt.

Neben dem oben beschriebenen System gibt es Anwendungen, in denen wenigstens Teilkomponenten von Überwachungs- oder Analysesystemen automatisch erzeugt werden. Die entsprechenden Arbeiten werden im Folgenden gegliedert nach der jeweiligen Teilkomponente vorgestellt.

Vorverarbeitung: Die Vorverarbeitung hat das Ziel, die erfassten Sensorsignale aufzubessern. Für die automatisierte Verbesserung von Signalen gibt es nur sehr wenige Ansätze. ISWANDY und KÖNIG definieren eine Menge von klassischen Filterverfahren. Im Zuge der Modellauswahl wird dann eines dieser Filterverfahren ausgewählt. Bei Bedarf werden auch dessen Parameter bestimmt [Iswandy u. a. 2009]. Wesentlich weiter verbreitet ist der Einsatz morphologischer Filter, die über ein heuristisches Optimierungsverfahren konfiguriert werden [Yu u. a. 2000, Jelodar u. a. 2006, Sharif u. a. 2010]. Andere Sensorsysteme verzichten auf einen expliziten Vorverarbeitungsschritt. Die Vorverarbeitung und die Kenngrößenerzeugung werden hier zusammengefasst. Das kombinierte Problem wird dann mit genetischer Programmierung gelöst [Guo u. a. 2005a, Espejo u. a. 2010].

Kenngrößenerzeugung: Der weit überwiegende Teil der Prozessüberwachungssysteme verwendet keine automatische Kenngrößenerzeugung. In diesen Systemen werden die Kenngrößen mit Expertenwissen aus den Rohdaten abgeleitet. Die Ableitung der Kenngrößen erfolgt dann vorwiegend auf Basis auf-

wändiger Untersuchungen [Abellan-Nebot u. a. 2010]. Einige Überwachungssysteme setzen zur Kenngrößenerzeugung halbautomatische Verfahren ein. Hierbei wird mit Expertenwissen eine Menge von Rechenoperationen für die Kenngrößenbildung bestimmt. In einem Optimierungsverfahren werden dann lediglich die geeigneten Rechenoperationen ausgewählt und parametriert [Iswandy u. a. 2009, Yu 2011].

In den letzten Jahren wurde die Forschung auf dem Gebiet der automatischen Kenngrößenerzeugung intensiviert. Dabei zeigte sich, dass sich die genetische Programmierung auf vielen Anwendungsfeldern sehr gut zur automatischen Kenngrößenerzeugung eignet [Lin u. a. 2005, Lin u. a. 2008, Espejo u. a. 2010].

Bei der Kenngrößenerzeugung mit genetischer Programmierung werden die Berechnungsvorschriften für die einzelnen Kenngrößen als Operatorbäume dargestellt. Die internen Knoten repräsentieren dabei die verwendeten arithmetischen Operatoren und Funktionen. Die Blätter des Baumes sind mit den Eingangsdaten und mit Konstanten belegt. Zur Bewertung der erzeugten Kenngrößen benötigt man wieder ein Maß für die Güte der Kenngrößenmenge. Hierbei werden die bereits von der Kenngrößenauswahl bekannten Wrapperverfahren [Rizki u. a. 2002, Guo u. a. 2005b, Muharram u. a. 2005, Guo u. a. 2005a, Muni u. a. 2006] und Filterverfahren [Bhanu u. a. 2005, Estébanez u. a. 2005, Estébanez u. a. 2007] eingesetzt [Espejo u. a. 2010].

Klassifikation: Wie bereits in Abschnitt 2.2.2 beschrieben, ist dem Autor kein Prozessüberwachungssystem bekannt, für welches das zu verwendende Klassifikationsverfahren automatisch gewählt wird. Allgemein sind nur wenige sensordatenbasierte Systeme bekannt, die von einer automatischen Auswahl Gebrauch machen. Diese werden nachfolgend beschrieben.

Iswandy und König bestimmen in ihrem bereits vorgestellten Multisensor-System einen passenden Klassifikator. Hierbei wählen sie den Klassifikator aus einer vorgegeben Menge aus und trainieren diesen. Dabei stehen drei Typen von neuronalen Netzen und eine Support Vector Machine zur Auswahl [Iswandy u. a. 2009].

In [Azadeh u. a. 2013] wird ein Verfahren für die Fehlerdiagnose in Kreiselpumpen vorgestellt. Auch dieses Verfahren wählt aus vorgegebenen Lernmodellen das am besten geeignete aus. Hier stehen Support Vector Machines und ein neuronales Netz zur Auswahl [Azadeh u. a. 2013].

Auch Sakthivel, Nair und Sugumaran stellen ein System zur Fehlerdiagnose in Kreiselpumpen vor. Im Vergleich zu den oben beschriebenen Verfahren wählten sie einen grundlegend anderen Ansatz. In ihrem System wird der Klassifikator mittels *Gene Expression Programming* dynamisch programmiert. Das Gene Expression Programming ist eine leichte Abwandlung der genetischen Programmierung [Chi Zhou u. a. 2003, Sakthivel u. a. 2012].

Kapitel 3

Ableitung von Anforderungen

Dieses Kapitel beschreibt ausgehend vom Stand der Wissenschaft und Technik, welche Anforderungen dem hier vorgestellten System zugrunde liegen. Diese Betrachtungen bilden den Ausgangspunkt für den Lösungsansatz, der im nachfolgenden Kapitel vorgestellt wird.

3.1 Defizite der heutigen Situation

Wie bereits in Abschnitt 1.1 beschrieben, ist die Fertigung in Hochlohnländern vor allem dann wettbewerbsfähig, wenn die entsprechenden Fertigungsprozesse weitgehend automatisiert sind. Allerdings gibt es in Fertigungsprozessen meist ein komplexes Zusammenspiel zwischen Maschinen, Werkzeugen, Werkstücken, Material, Bedienern und der Umgebung. Solche Abläufe sind nur dann beherrschbar, wenn die relevanten Prozessvariablen sensorisch erfasst und überwacht werden. Eine automatisierte Fertigung ist somit nur sinnvoll, wenn zuverlässige Prozess-

überwachungssysteme existieren. Untersuchungen an mehreren Fertigungsprozessen zeigen, dass Prozesse, in denen ein Überwachungssystem zum Einsatz kommt, zwischen 10 % und 40 % weniger Ausschuss produzieren als unüberwachte Prozesse. Dabei ist noch nicht berücksichtigt, dass durch unkontrolliertes Prozessverhalten auch Werkzeuge, Maschinen oder gar Menschen zu Schaden kommen können [Byrne u. a. 1995, Kovač u. a. 2011].

Kern der Prozessüberwachung ist die schnelle Auswertung der erfassten Prozessvariablen. Hierbei gilt es, ausgehend von diesen Sensordaten zyklussynchron den aktuellen Prozesszustand zu prognostizieren. In [Kitazawa u. a. 2011] wird gezeigt, dass die Prognosegüte eines passend trainierten intelligenten Prozessüberwachungssystems die Prognosegüte herkömmlicher Überwachungssysteme deutlich übertrifft. Allgemein wird den intelligenten Prozessüberwachungssystemen in zahlreichen wissenschaftlichen Veröffentlichungen eine sehr hohe Leistungsfähigkeit bescheinigt [Liang u. a. 2004, Teti u. a. 2010].

Bei all ihrer Leistungsfähigkeit haben intelligente Prozessüberwachungssysteme aber heute noch zwei wesentliche Schwächen, die einem flächendeckenden Einsatz in Produktivsystemen entgegenstehen:

Benötigtes Expertenwissen: Nach heutigem Stand der Forschung müssen Prozessüberwachungssysteme noch weitgehend manuell an den zu überwachenden Fertigungsprozess angepasst werden. Im Rahmen dieser Anpassung müssen Verfahren für die Signalverarbeitung, die Kenngrößenbildung, die Kenngrößenselektion und die abschließende Klassifikation ausgewählt und passend parametriert werden. Diese Aufgaben setzen Expertenwissen auf dem Gebiet der digitalen Signalanalyse voraus.

Gleichzeitig verlangt insbesondere die Kenngrößenbildung detailliertes Wissen über die Zusammenhänge im Fertigungsprozess. Daher muss zusätzlich

ein Fachmann für das zu überwachende Produktionsverfahren hinzugezogen werden [Neher u. a. 2010, Neher 2012].

Hoher Zeitaufwand: Für die meisten Fertigungsprozesse sind zunächst auch den Experten nicht alle Zusammenhänge auf Sensordatenebene bekannt. Daher müssen die Kenngrößen üblicherweise mit aufwändigen Versuchen ermittelt werden. Diese Versuche können viele Monate in Anspruch nehmen. Oft ist damit auch eine Beeinträchtigung der Fertigungskapazität des zu überwachenden Prozesses verbunden [Kaupp u. a. 2012b, Neher 2012].

3.2 Anforderungen

Aus den oben beschriebenen Defiziten der heutigen Situation lassen sich die Anforderungen an ein System für die automatische Erzeugung von Prozessüberwachungssystemen ableiten. Das zu erstellende System soll in der Lage sein, auf Basis einer gegebenen Lerndatenmenge ein voll funktionsfähiges Prozessüberwachungssystem zu erzeugen. Das erzeugte Prozessüberwachungssystem soll den durch die Lerndaten beschriebenen Fertigungsprozesszyklus dann zyklussynchron überwachen können. Dieses grundlegende Systemkonzept ist in Abbildung 3.1 dargestellt.

Um nutzbringend eingesetzt werden zu können, muss das zu erstellende System folgende Anforderungen erfüllen:

Erzeugung ohne Expertenwissen: Das System soll so gestaltet sein, dass für die Erzeugung eines passenden Prozessüberwachungssystems kein Expertenwissen benötigt wird. Vielmehr soll die Anpassung auch durch einen geübten Anlagenbediener vorgenommen werden können.

Beschleunigung der Erzeugung: Mit dem System soll die Erzeugung von Prozessüberwachungssystemen signifikant beschleunigt werden. So sollen die

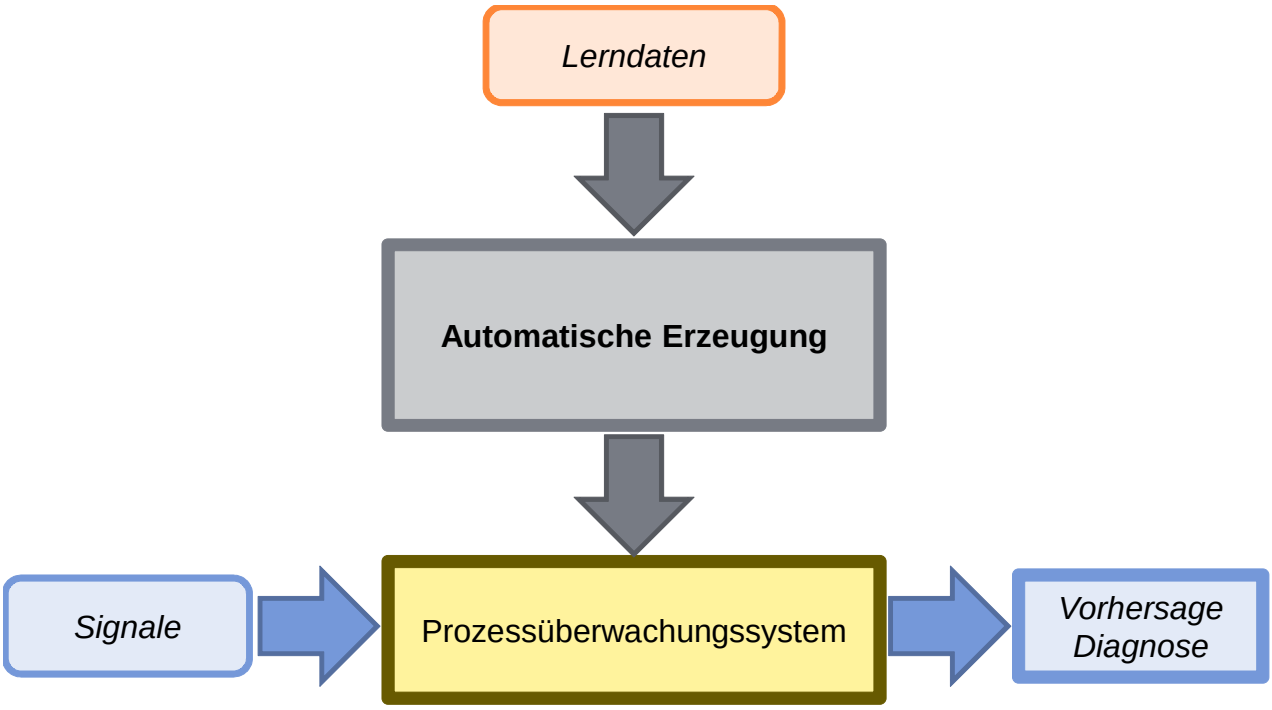


Abbildung 3.1: Systemkonzept

Gewinnung der benötigten Lerndaten und die anschließende automatische Erzeugung des Prozessüberwachungssystems nur wenige Tage in Anspruch nehmen.

Flexibilität: Das System soll ein sehr breites Spektrum von Produktionsprozessen abdecken können. Daher muss insbesondere die Kenngrößenerzeugung so flexibel gestaltet werden, dass sie aus verschiedensten Messgrößen aussagekräftige Kennzahlen ableiten kann.

Kapitel 4

Lösungsansatz

Für die automatisierte Konfiguration von Prozessüberwachungssystemen mussten neue Methoden erarbeitet und software-technisch umgesetzt werden. Im Folgenden wird die Konzeption des neuen Verfahrens und seiner Teilsysteme beschrieben.

4.1 Grundidee

Ziel dieser Arbeit ist die Entwicklung eines Verfahrens, das automatisch voll funktionsfähige intelligente Prozessüberwachungssysteme erzeugt. Der hier vorgestellte Lösungsansatz arbeitet nach dem in Abbildung 4.1 gezeigten Grundkonzept.

Den Kern dieses Konzeptes bildet ein *generisches Prozessüberwachungssystem*. Dieses System bietet die Infrastruktur für die Datenerfassung und bildet die benötigten Datenflüsse ab. Es enthält aber zunächst keinerlei Logik für die Verarbeitung und Bewertung der erfassten Daten. Diese Verarbeitungs- und Bewertungslogik wird in Form eines *Analysemodells* von außen vorgegeben. Das

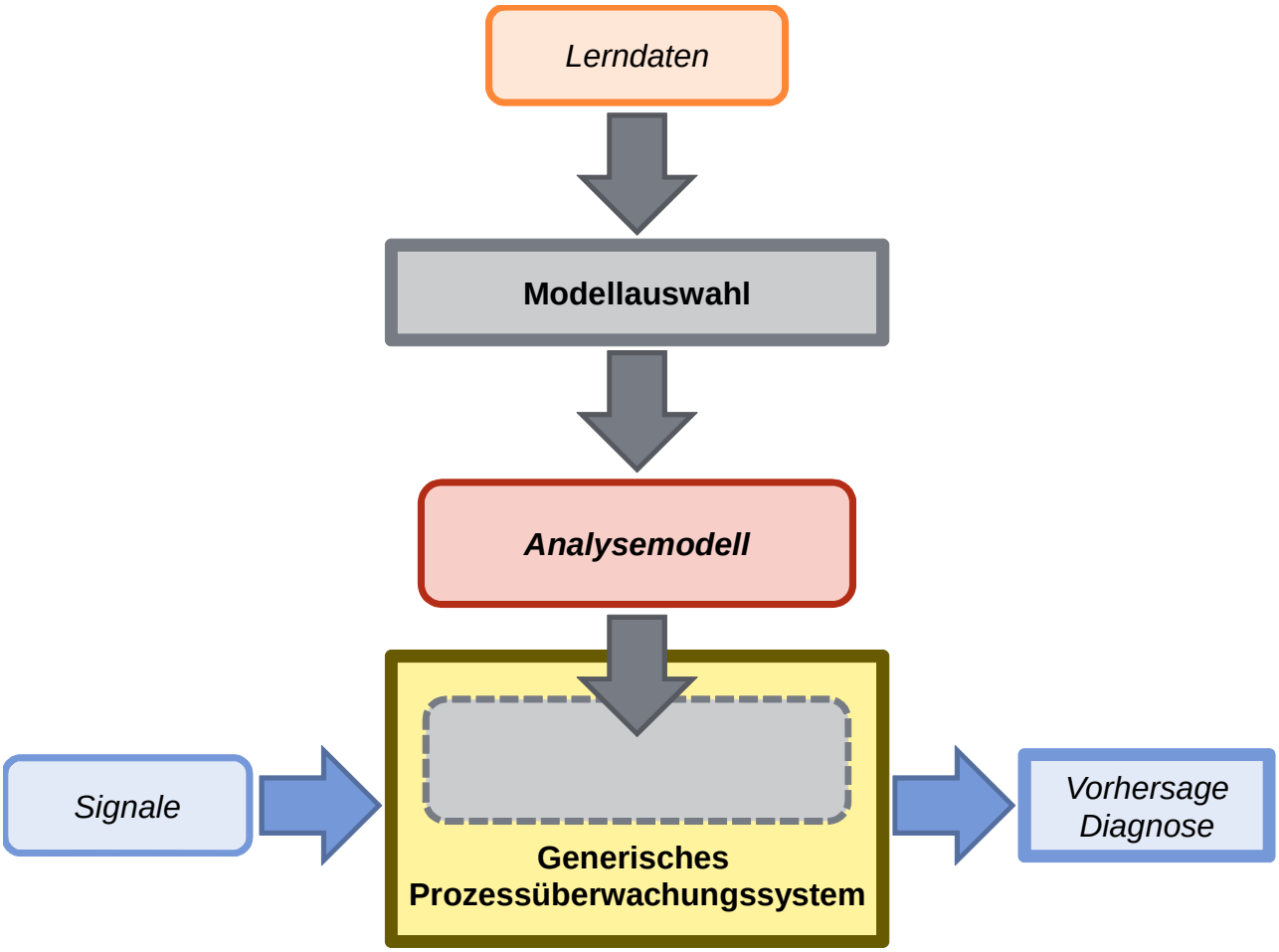


Abbildung 4.1: Konzept der automatischen Erzeugung von Prozessüberwachungssystemen

Analysemodell enthält dabei die zu verwendenden Verfahren für die Signalverarbeitung, die Kenngrößenbildung und die Kenngrößenselektion sowie ein passendes Lernmodell. Durch Setzen eines geeigneten Analysemodells lässt sich das generische Prozessüberwachungssystem an jeden Fertigungsprozess anpassen.

Das zu verwendende Analysemodell wird im Rahmen der *Modellauswahl* bestimmt. Die Modellauswahl ist so konzipiert, dass sie einen Satz von Lerndaten

aus dem Zielprozess auswertet. Auf Basis dieser Lerndaten wird dasjenige Analysemodell ausgewählt, welches das generische Prozessüberwachungssystem am besten an den Fertigungsprozess anpasst. Das mit dem entsprechenden Analysemodell konfigurierte generische Überwachungssystem ist dann das gewünschte Prozessüberwachungssystem für diesen Fertigungsprozess.

Die für die Anpassung benötigten Lerndatensätze können von einem geübten Bediener des Fertigungsprozesses mit vergleichsweise geringem Zeitaufwand erzeugt werden.

4.2 Generisches Prozessüberwachungssystem

4.2.1 Herauslösen des Analysemodells

In Abschnitt 2.2.1 wird der allgemeine Aufbau eines intelligenten Prozessüberwachungssystems beschrieben. Abbildung 2.3 zeigt mit der Signalverarbeitung, der Kenngrößenbildung, der Kenngrößenselektion und dem Lernmodell die typischen Teilkomponenten solch eines Prozessüberwachungssystems. Die dort dargestellte allgemeine Systemarchitektur bildet die Basis für die Erstellung des generischen Überwachungssystems, das hier beschrieben wird.

Bei der Konzeption des generischen Systems wurden die Teilkomponenten Signalverarbeitung, Kenngrößenbildung, Kenngrößenselektion und die Lernmaschine aus dem eigentlichen Überwachungssystem herausgelöst und durch Platzhalter ersetzt. Zudem wurden die Signalverarbeitung und die Kenngrößenbildung zu einer gemeinsamen Komponente verschmolzen. Das generische Überwachungssystem hat somit den in Abbildung 4.2 gezeigten Grundaufbau. Die gestrichelt umrandeten, grauen Blöcke stellen hierbei die Platzhalter im generischen Überwachungssystem dar.

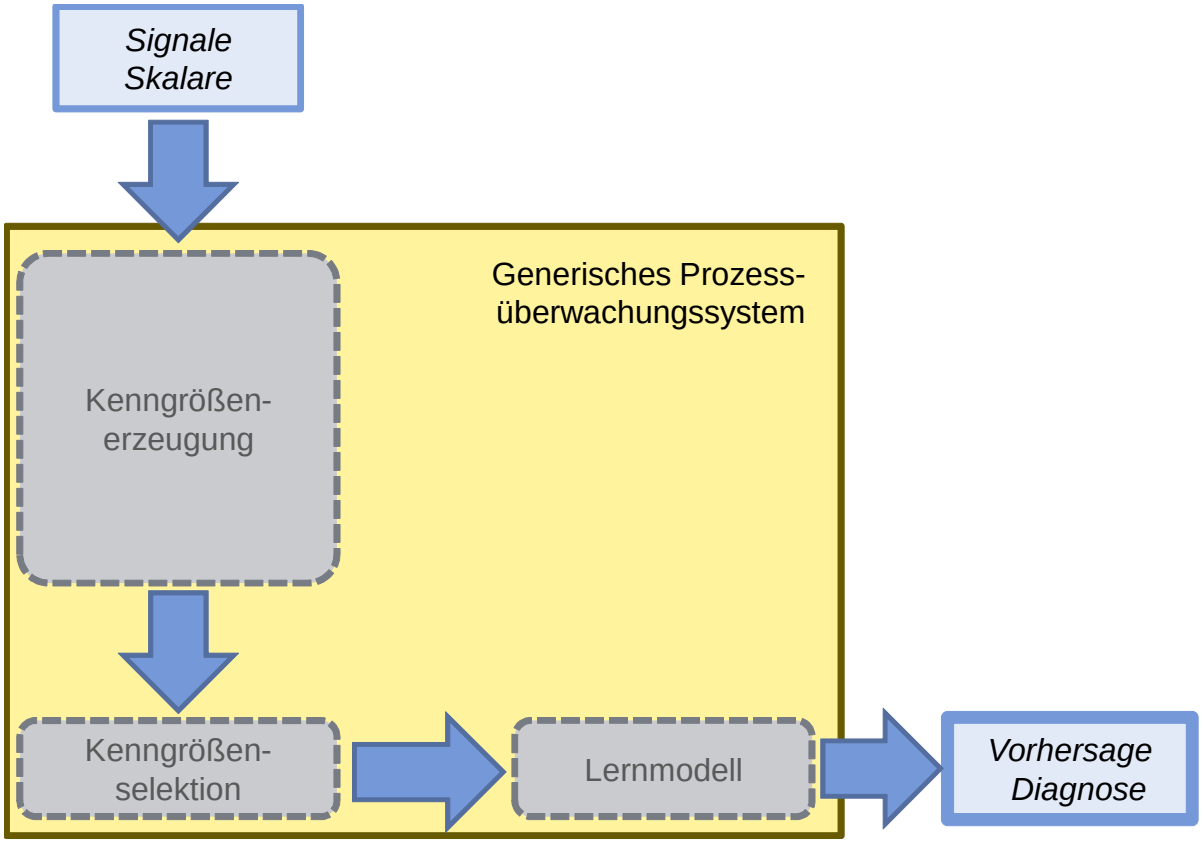


Abbildung 4.2: Grundaufbau des generischen Prozessüberwachungssystems

Software-technisch wurde der oben beschriebene Grundaufbau unter Verwendung des Entwurfsmusters *Strategy* realisiert. Hierbei handelt es sich um ein Verhaltensmuster aus dem Bereich der Software-Architektur. Das Strategy-Entwurfsmuster erlaubt die flexible Konfiguration von Software-Modulen, indem eine von mehreren alternativen Strategien dynamisch ausgewählt wird. Die einzelnen Strategien werden dabei als Software-Objekte realisiert, die nach außen hin die gleiche Schnittstelle anbieten [Gamma 1995].

Bei der Umsetzung des generischen Prozessüberwachungssystems wurde für jeden der drei Platzhalter aus Abbildung 4.2 eine passende Strategie-Schnittstelle

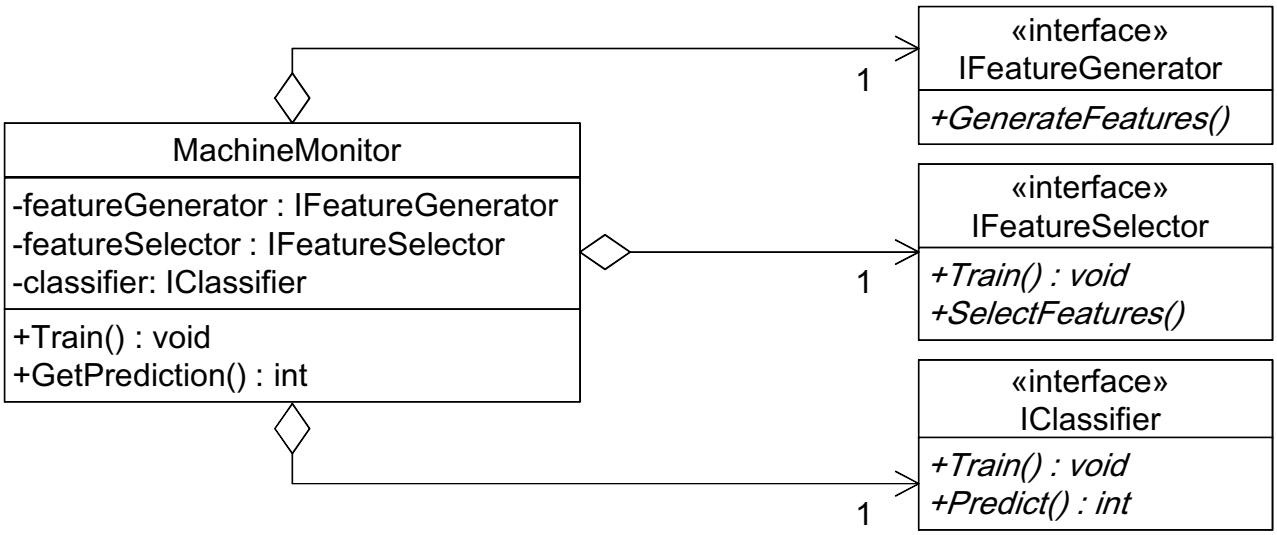


Abbildung 4.3: Statische Struktur des generischen Prozessüberwachungssystems

bereitgestellt. Die Kenngrößenbildung, die Kenngrößenselektion und die Lernmaschine werden über die Schnittstellen *IFeatureGenerator*, *IFeatureSelector* und *IClassifier* angesprochen. Die Klasse *MachineMonitor* realisiert das eigentliche generische Prozessüberwachungssystem. In dieser Klasse werden drei Attribute (*featureGenerator*, *featureSelector* und *classifier*) deklariert, die als Platzhalter für die konkreten Instanzen der einzelnen Strategien fungieren. Die Klasse *MachineMonitor* ist so konzipiert, dass die konkreten Strategien bereits bei der Erzeugung eines Prozessüberwachungssystems übergeben werden müssen. So ergibt sich für das generische Überwachungssystem die in Abbildung 4.3 gezeigte Klassenstruktur.

4.2.2 Prognose von Prozesszuständen

Das generische Prozessüberwachungssystem soll möglichst flexibel sein. Deshalb wurde es so gestaltet, dass es als Eingabedaten nicht nur Signale, sondern auch skalare Messgrößen berücksichtigen kann. Das System akzeptiert daher als Eingabe einen Datensatz, der beliebig viele diskret abgetastete Signale unterschiedlicher Länge sowie beliebig viele skalare Messgrößen enthalten kann.

Der Prognosevorgang eines konfigurierten generischen Überwachungssystems verläuft analog zur Prognose mit konventionell erzeugten Überwachungssystemen. Auch hier werden die Messwertdatensätze zunächst an die Kenngrößenerzeugung übergeben. Die dort erzeugten Kenngrößen werden direkt der Kenngrößenselektion zugeführt. Dort werden diejenigen Werte ausgewählt, die in der Trainingsphase als relevant identifiziert wurden. Die Menge der selektierten Werte wird an das parametrierte und trainierte Lernmodell übergeben, das dann die gewünschte Vorhersage oder Diagnose liefert.

4.3 Umsetzung des Analysemodells

Im hier vorgestellten System enthält das Analysemodell alle Informationen, die benötigt werden, um das generische Prozessüberwachungssystem an einen konkreten Fertigungsprozess anzupassen. Diese Informationen werden im Zuge der Modellauswahl gewonnen und im Analysemodell in Form einer Modellbeschreibung verwaltet.

Nachfolgend wird der Aufbau der Modellbeschreibung vorgestellt. Zudem wird gezeigt, wie ausgehend von dieser Modellbeschreibung ein funktionsfähiges Prozessüberwachungssystem erzeugt werden kann.

4.3.1 Grundkonzept

Im hier beschriebenen Verfahren fallen dem Analysemodell zwei Aufgaben zu: Zum einen soll es während der Modellauswahl die momentane Konfiguration aller enthaltenen Teilkomponenten verwalten. Und zum anderen soll es fertige Instanzen dieser Teilkomponenten liefern, die dann vom generischen Prozessüberwachungssystem direkt verwendet werden können. Um diesen Anforderungen gerecht zu werden, ist das Analysemodell so gestaltet, dass es intern mit einer *Modellbeschreibung* arbeitet. Diese Modellbeschreibung setzt sich aus drei *Konfigurationsbeschreibungen* zusammen. Jede dieser drei Beschreibungen spezifiziert dabei die Konfiguration einer einzelnen Teilkomponente. Darüber hinaus bietet das Analysemodell die Möglichkeit, aus den Konfigurationsbeschreibungen bei Bedarf die entsprechenden Software-Objekte zu erzeugen. Die so instanziierten Software-Objekte sind dann die Strategie-Objekte, die das generische Überwachungssystem für die Kenngrößenerzeugung, die Kenngrößenselektion und die Klassifikation verwenden kann (siehe Abschnitt 4.2.1).

Das Analysemodell erzeugt die Software-Objekte direkt aus den Konfigurationsbeschreibungen. Die so erzeugten Teilkomponenten sind somit lediglich konfiguriert. Die Kenngrößenselektion und die Klassifikation müssen vor ihrer Verwendung noch trainiert werden. Die Kenngrößenerzeugung hat nach der Konfiguration keine freien Parameter mehr. Deshalb ist für diese Teilkomponente kein Trainingsschritt mehr nötig.

Der Datenfluss zwischen den einzelnen Teilkomponenten wird im generischen Überwachungssystem abgewickelt. Daher muss das instanziierte Analysemodell in das generische Überwachungssystem eingesetzt werden. Das so entstehende untrainierte Überwachungssystem wird dann als Ganzes trainiert. Um ein funktionsfähiges Prozessüberwachungssystem aus einer Modellbeschreibung zu erzeugen, sind also die in Abbildung 4.4 gezeigten Schritte notwendig.

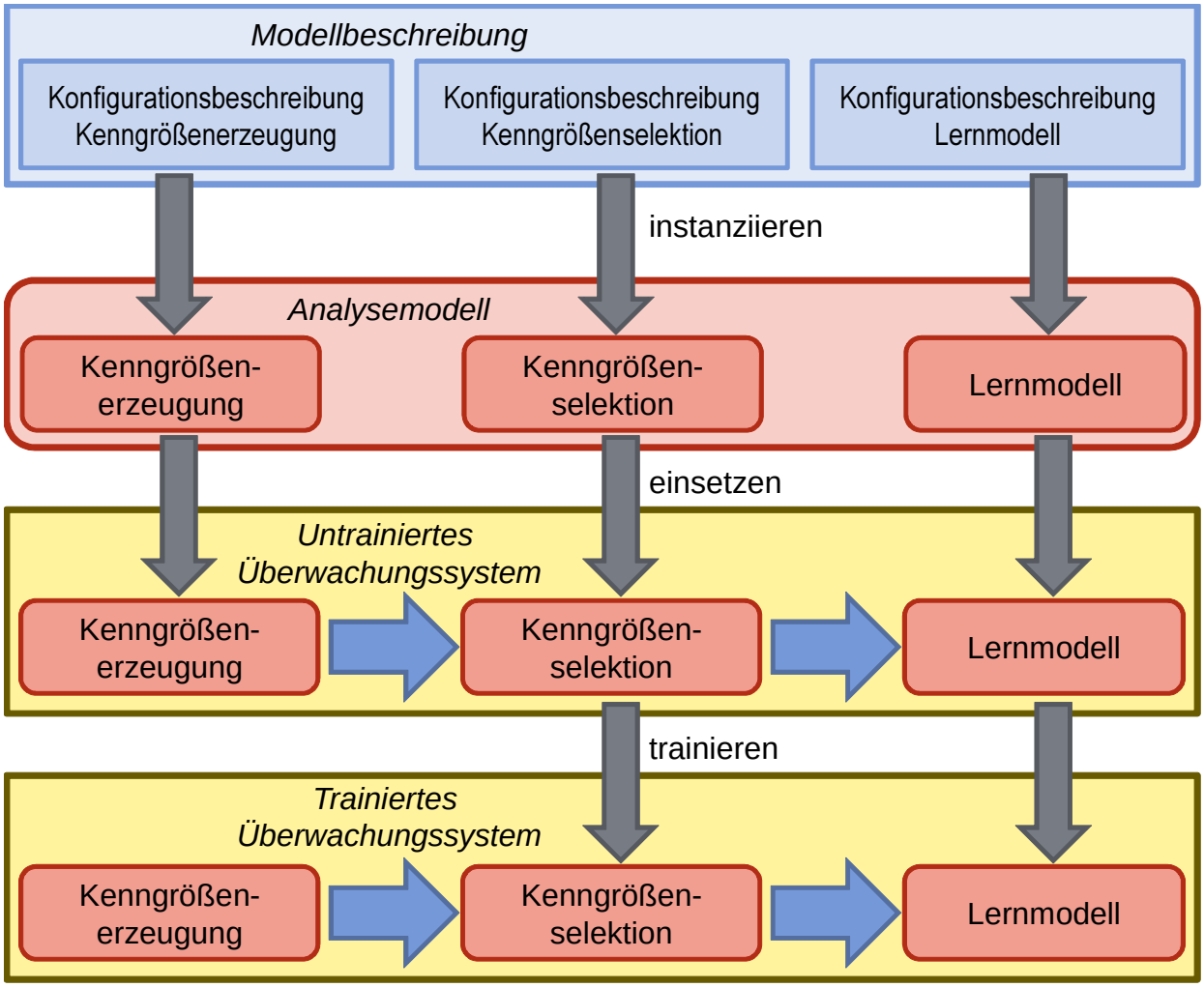


Abbildung 4.4: Ablauf der Erzeugung eines Prozessüberwachungssystems auf Basis einer Modellbeschreibung

4.3.2 Realisierung der Kenngrößenerzeugung

In vielen herkömmlichen Prozessüberwachungssystemen sind die Aufbereitung der Rohsignale und die Kenngrößenbildung voneinander getrennt (siehe Abschnitt 2.2.1). Im Rahmen der hier vorgestellten Arbeit wurden die Signalverarbeitung

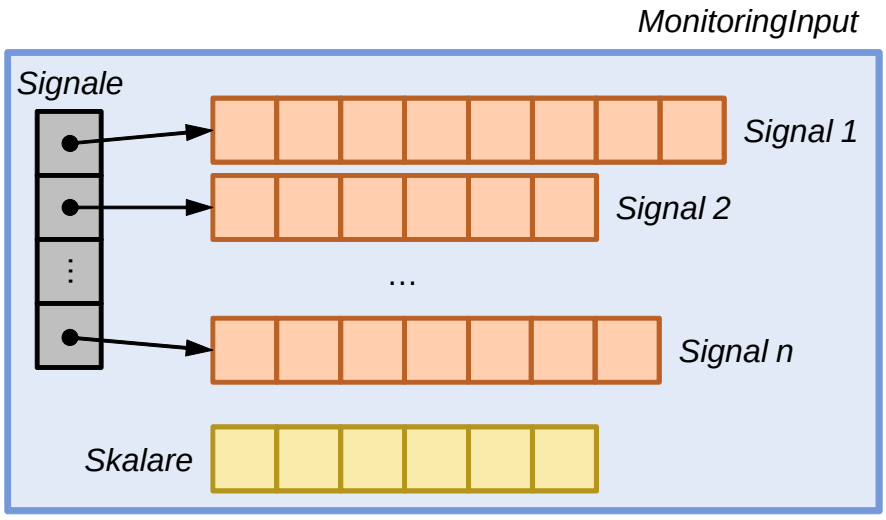


Abbildung 4.5: Struktur der Eingabedaten der Kenngrößenerzeugung

und die Kenngrößenbildung so konzipiert, dass sie automatisch erzeugt werden können. Der gewählte Aufbau ermöglicht es, die beiden Verarbeitungsschritte zu einer gemeinsamen Komponente zu verschmelzen. So entsteht das kombinierte Modul *Kenngrößenerzeugung*, das nachfolgend vorgestellt wird.

Dieser Abschnitt stellt den Aufbau der Konfigurationsbeschreibung einer Kenngrößenerzeugung vor. Zudem zeigt er, wie sich aus solch einer Konfigurationsbeschreibung der entsprechende Kenngrößenerzeuger instanziieren lässt.

Aufbau der Konfigurationsbeschreibung

Die Eingabe der Kenngrößenerzeugung ist ein Satz von Messdaten. Solch ein Datensatz hat in dem hier vorgestellten System die Klasse *MonitoringInput*. Dieser Datentyp ist so realisiert, dass die entsprechenden Instanzen eine variable Anzahl von Signalen beliebiger Länge und auch eine variable Anzahl von Skalaren enthalten können. Hierzu wird in der Datenstruktur ein Feld bereitgestellt, das die

Skalare aufnimmt. Außerdem gibt es dort ein Feld, in dem Referenzen auf die enthaltenen Signale abgelegt sind. Diese Signale sind Felder von Abtastwerten, die ebenfalls innerhalb der Klasse verwaltet werden (siehe Abbildung 4.5). Die Ausgabe der Kenngrößenerzeugung ist ein eindimensionales Feld reellwertiger Zahlen. Dieses Feld enthält die erzeugten Kenngrößen.

Für die Kenngrößenerzeugung stehen zwei alternative Verfahren zur Verfügung: die Identitätsfunktion und die dynamisch programmierten Kenngrößenerzeuger. Die *Identitätsfunktion* kopiert lediglich alle Messwerte des Eingabedatensatzes nacheinander in das Ausgabefeld. So projiziert sie die mehrdimensionalen Eingangsdaten in das eindimensionale Zielformat. Die enthaltenen Messwerte werden dabei nicht verändert. Es findet also weder eine Vorverarbeitung der Signale noch eine Berechnung komplexer Kenngrößen statt. Dieses Verfahren ist vor allem dann geeignet, wenn die Eingangsdaten verhältnismäßig wenige Messwerte enthalten und wenn die enthaltenen Signale zudem von hoher Qualität sind.

Die *dynamisch programmierten Kenngrößenerzeuger* werden während der Modellauswahl automatisch erstellt. Hierzu wurde im Rahmen der vorliegenden Arbeit eigens eine passende Programmiersprache konzeptioniert und realisiert. In dieser Programmiersprache werden die Kenngrößenerzeuger formuliert. Die neu entwickelte Programmiersprache befolgt das funktionale Programmierparadigma. Dementsprechend werden die Programme komplett durch ineinander verschachtelte Funktionsaufrufe beschrieben.

Jeder dynamisch programmierte Kenngrößenerzeuger generiert eine bestimmte Anzahl von Kenngrößen. Dabei wird die Berechnungsvorschrift jeder Kenngröße durch einen eigenen Operatorbaum (siehe Abschnitt 2.3.3) definiert. Für den Aufbau dieser Operatorbäume stellt die Programmiersprache eine Menge von Funktionen und Terminalen bereit. Die zur Verfügung stehende Funktionsmenge ist in Tabelle 4.1 beschrieben. Es ist zu erkennen, dass die Programmiersprache

Name	**Anzahl Parameter**	**Beschreibung**
Plus	2	*Addition* (nach oben und unten beschränkt durch Minimal- bzw. Maximalwert des Datentyps *double*)
Minus	2	*Subtraktion* (nach oben und unten beschränkt durch Minimal- bzw. Maximalwert des Datentyps *double*)
Mult	2	*Multiplikation* (nach oben und unten beschränkt durch Minimal- bzw. Maximalwert des Datentyps *double*)
Div	2	*Division* (nach oben und unten beschränkt durch Minimal- bzw. Maximalwert des Datentyps *double*; bei Division durch Null: Minimal- bzw. Maximalwert)
Sqr	1	*Quadratfunktion* (nach oben beschränkt durch Maximalwert des Datentyps *double*)
Sqrt	1	*Quadratwurzel* des Betrags des Parameters
Sin	1	*Sinus*
Cos	1	*Cosinus*
Asin	1	*Arcus-Sinus*
Acos	1	*Arcus-Cosinus*
Tan	1	*Tangens* (nach oben und unten beschränkt durch Minimal- bzw. Maximalwert des Datentyps *double*; bei ungültigem Wert: 0.0)
Tanh	1	*Tangens Hyperbolicus*
Recip	1	*Kehrwert* (nach oben und unten beschränkt durch Minimal- bzw. Maximalwert des Datentyps *double*; bei Division durch Null: Minimal- bzw. Maximalwert)
Log	1	*natürlicher Logarithmus* des Betrags des Parameters (nach oben und unten beschränkt durch Minimal- bzw. Maximalwert des Datentyps *double*)
Neg	1	*Negation*
Abs	1	*Betragsfunktion*

Tabelle 4.1: Menge der Funktionen der Kenngrößenerzeugung

lediglich ein minimales Gerüst mathematischer Grundfunktionen anbietet. Aus diesen Grundfunktionen werden bei Bedarf automatisch komplexere Operationen erzeugt. Die Ein- und Ausgabewerte der hier vorgestellten Funktionen sind jeweils vom Typ *double*. Die Funktionen sind intern so implementiert, dass die Ausgabewerte nach oben und nach unten beschränkt sind. So wird sichergestellt, dass die Ergebnisse auch tatsächlich im Wertebereich des Datentyps *double* liegen.

Die Programmiersprache stellt drei Typen von Terminalen zur Verfügung: Konstanten, Messwertvariablen und Bereichsvariablen. Bei den *Konstanten* handelt es sich um Werte, die unabhängig von den aktuellen Eingabedaten sind. Der Wert einer Konstanten muss bei der Programmerzeugung festgelegt werden.

Eine *Messwertvariable* ist ein Platzhalter für einen konkreten Messwert aus dem Eingabedatensatz. Innerhalb des Eingabedatensatzes hat jedes Signal einen festen Index. Auch die Abtastwerte innerhalb der Signale und die Skalare haben feste Indices. Eine Messwertvariable hält sich die Indices des Messwertes, den sie repräsentiert. Bei der Ausführung wird die Variable dann mit dem Messwert belegt, der im Eingabedatensatz an der entsprechenden Stelle abgelegt ist.

Eine *Bereichsvariable* steht für einen Wert, der aus einem ganzen Signal oder aus einem bestimmten Bereich eines Signals berechnet wird. Die Funktionen, die zur Berechnung solch einer Bereichsvariablen verwendet werden können, sind in Tabelle 4.2 aufgelistet. Die Funktion *StatMoment()* liefert einen der ersten vier statistischen Momente eines kompletten Signals. Dabei müssen das gewünschte Moment und das Signal, auf dem dieses Moment berechnet werden soll, als Parameter mitgegeben werden. Die Auswertung der statistischen Momente von Signalen hat vor allem in der Verschleißüberwachung große Bedeutung, da sich Unregelmäßigkeiten in Vibrationssignalen so zuverlässig erkennen lassen [Guo u. a. 2005a, Sakthivel u. a. 2012]. Die anderen Bereichsfunktionen betrachten jeweils einen eingeschränkten Bereich eines Signals. Als Parameter werden hier der Index

Name	**Anzahl Parameter**	**Beschreibung**
StatMoment	2	Einer der ersten vier statistischen Momente eines gesamten Signals; Auswahl des Signals und des Moments über Parameter
Average	3	Durchschnitt der Abtastwerte eines Signals im gegebenen Bereich
Minimum	3	Minimum der Abtastwerte eines Signals im gegebenen Bereich
Maximum	3	Maximum der Abtastwerte eines Signals im gegebenen Bereich
Integral	3	Summe der Abtastwerte eines Signals im gegebenen Bereich (nach oben und unten beschränkt durch Minimal- bzw. Maximalwert des Datentyps *double*)
MinMaxDiff	3	Differenz des Minimums und des Maximums der Abtastwerte im gegebenen Bereich (nach oben und unten beschränkt durch Minimal- bzw. Maximalwert des Datentyps *double*)
MinMaxDist	3	Zeitlicher Abstand zwischen dem Auftreten des Minimums und des Maximums im gegebenen Bereich

Tabelle 4.2: Menge der Bereichsfunktionen der Kenngrößenerzeugung

des zu analysierenden Signals sowie die Indices des ersten und letzten Abtastwertes des relevanten Signalbereichs übergeben. Die hier verwendeten Bereichsfunktionen entsprechen den klassischen Funktionen für die Signalanalyse im Zeitbereich (siehe Abschnitt 2.2.2).

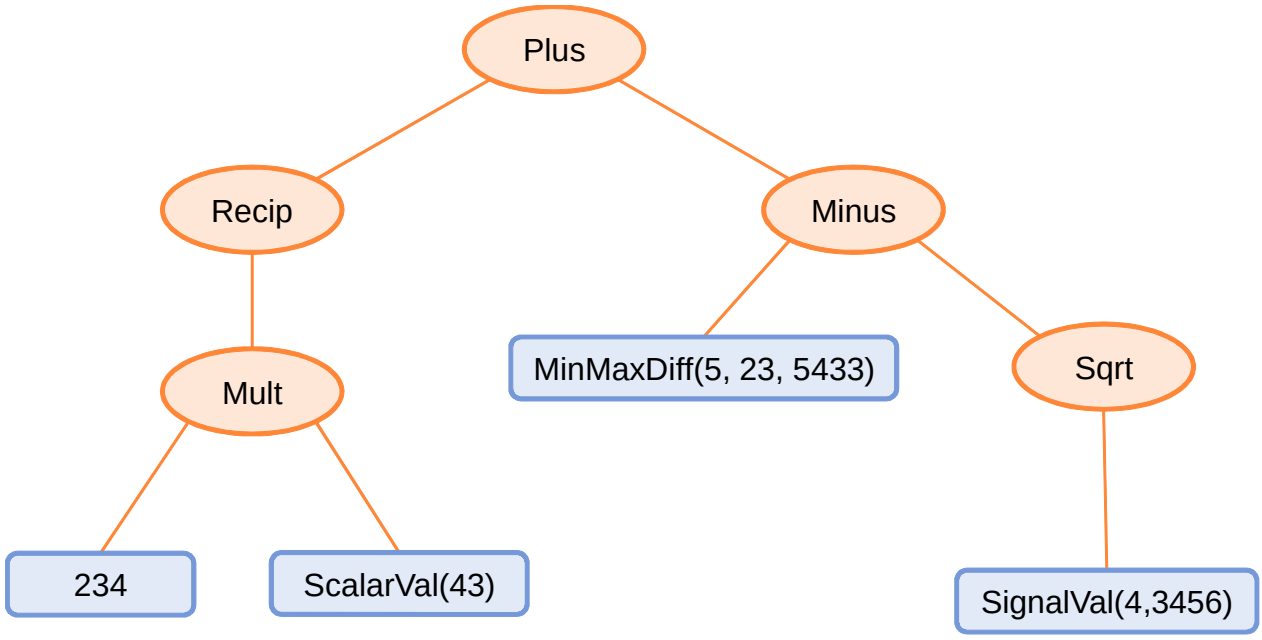


Abbildung 4.6: Beispielhafter Operatorbaum aus Primitiven der Programmiersprache für die Kenngrößenerzeuger

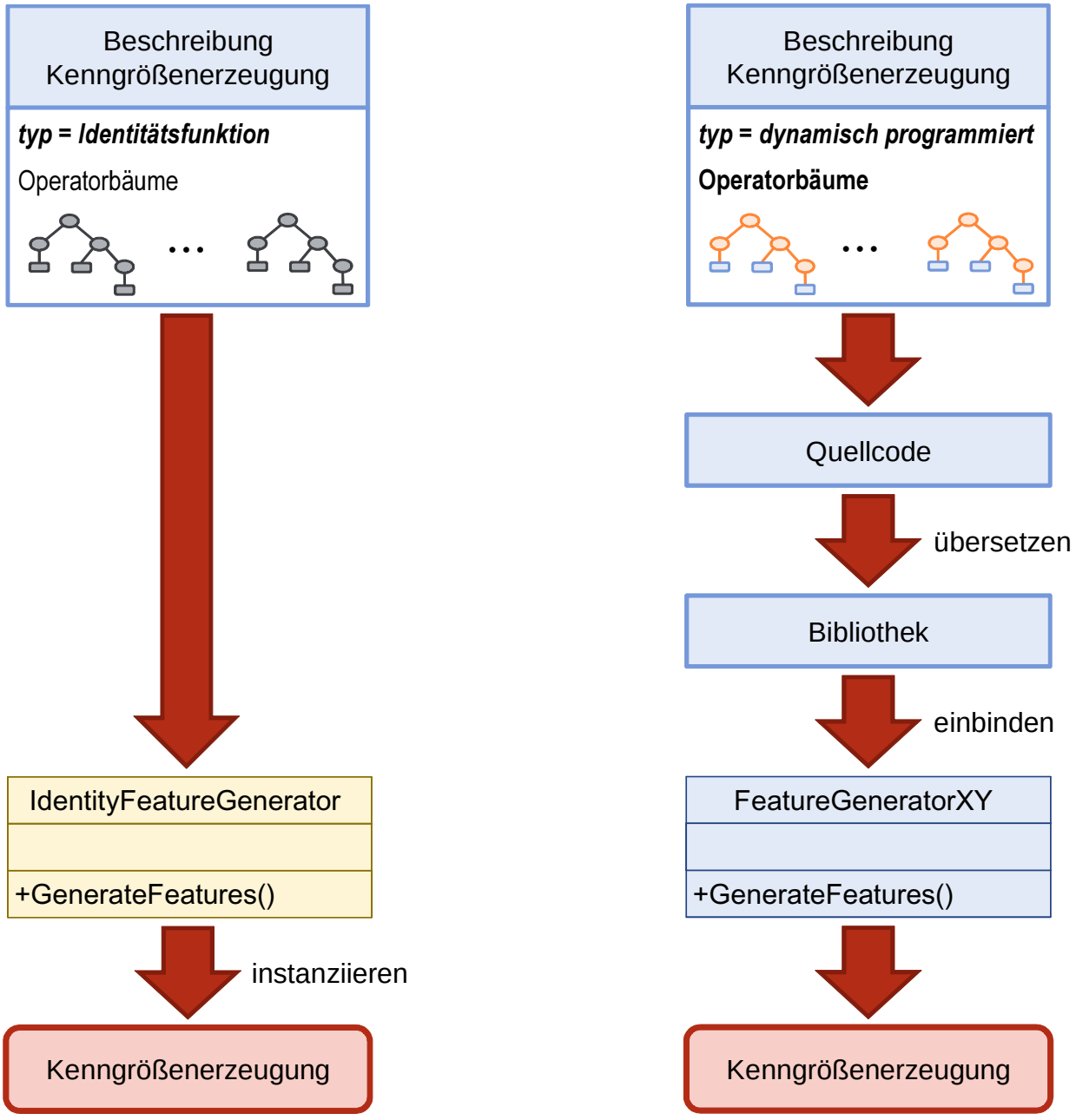
Abbildung 4.6 zeigt beispielhaft einen Operatorbaum, der aus den Primitiven der oben beschriebenen Programmiersprache gebildet wird. Die Terminale sind dort als blaue Rechtecke dargestellt. Das Terminal mit dem Inhalt 234 ist dabei eine Konstante (mit dem Wert 234). Das Terminal $ScalarVal(43)$ ist eine Messwertvariable, der bei der Kenngrößenerzeugung der Wert des 43-sten Skalares des jeweiligen Eingabedatensatzes zugewiesen wird. Auch das Terminal $SignalVal(4, 3456)$ ist eine Messwertvariable. Dieser Variablen wird bei der Kenngrößenerzeugung der 3456-ste Abtastwert des vierten Signals aus dem Eingabedatensatz zugewiesen. Beim Terminal $MinMaxDiff\,(5, 23, 5433)$ handelt es sich um eine Bereichsvariable. Dieser wird der Wert der Min-Max-Differenz des Bereichs vom 23-sten bis zum 5433-sten Abtastwert des fünften Signals zugewiesen.

Die Konfigurationsbeschreibung eines Kenngrößenerzeugers beinhaltet den Typen des Kenngrößenerzeugers (Identitätsfunktion oder dynamisch programmiert) und die Menge von Operatorbäumen, die den dynamisch programmierten Kenngrößenerzeuger repräsentiert.

Erzeugung von Instanzen

Das Vorgehen bei der Instanziierung eines Kenngrößenerzeugers ist abhängig von dessen Typ. Soll die Identitätsfunktion verwendet werden, so kann das für die Erzeugung zuständige Analysemodell einfach ein Objekt der Klasse *IdentityFeatureGenerator* instanziieren. Diese Klasse ist im hier vorgestellten System bereits definiert und kann ohne zusätzlichen Aufwand verwendet werden (siehe Abbildung 4.7(a)). Dies ist möglich, weil das gewünschte Verhalten der Identitätsfunktion zum Erstellungszeitpunkt des Systems bereits spezifiziert war.

Die Beschreibungen der dynamisch programmierten Kenngrößenerzeuger werden hingegen erst zur Ausführungszeit des Systems erstellt. Die entsprechende Software-Klasse kann daher im System noch nicht vorhanden sein. Sie muss dem System zur Laufzeit hinzugefügt werden. In diesem Fall folgt die Erstellung des Kenngrößenerzeugers dem in Abbildung 4.7(b) gezeigten Ablauf. Im Zuge der Quellcode-Synthese wird zunächst die durch die Operatorbäume beschriebene Funktionalität des Kenngrößenerzeugers in kompilierfähigen Quellcode überführt. Der Quellcode wird dabei so gestaltet, dass er eine Software-Klasse mit der vom Prozessüberwachungssystem geforderten Schnittstelle *IFeatureGenerator* beschreibt (siehe Abbildung 4.3). Die so erstellte Quellcode-Datei wird übersetzt und in eine Klassenbibliothek eingebunden. Diese Bibliothek wird vom System geladen, und die darin beschriebene dynamisch programmierte Software-Klasse wird importiert. In der Abbildung 4.7(b) trägt diese Klasse den Namen *FeatureGeneratorXY*. Die importierte Klasse kann nun instanziiert werden. Man erhält



(a) Identitätsfunktion

(b) Dynamisch programmierter Kenngrößenerzeuger

Abbildung 4.7: Ablauf der Instanziierung von Kenngrößenerzeugern für die verschiedenen Typen

so einen Kenngrößenerzeuger mit der Funktionalität, die in der Konfigurationsbeschreibung definiert wurde.

4.3.3 Realisierung der Kenngrößenselektion

Für die Kenngrößenselektion werden im hier vorgestellten System ausschließlich Filterverfahren verwendet. Wrapperverfahren sind aufgrund der hohen Dimension der Eingangsdatensätze zu laufzeitintensiv. Im Folgenden wird der Aufbau der Konfigurationsbeschreibung für die Kenngrößenselektion vorgestellt. Zudem wird gezeigt, wie aus Konfigurationsbeschreibungen lauffähige Instanzen erzeugt werden.

Aufbau der Konfigurationsbeschreibung

Das hier beschriebene System stellt für die Kenngrößenselektion vier alternative Verfahren zur Verfügung: Ranking, Vorwärtsselektion, Rückwärtselimination und die Identitätsfunktion, die alle Kenngrößen ungefiltert durchschleust.

Mit der Bereitstellung der Identitätsfunktion bietet das System die Möglichkeit, die vom Kenngrößenerzeuger generierten Werte direkt an das Lernmodell zu geben. Durch die Wahl der Identitätsfunktion lässt sich also die Kenngrößenselektion deaktivieren.

Für den Ranker, die Vorwärtsselektion und die Rückwärtselimination muss die zu verwendende Relevanzfunktion als Hyperparameter angegeben werden. Tabelle 4.3 listet die bereitgestellten Verfahren zur Kenngrößenselektion und deren Hyperparameter auf. Die Konfigurationsbeschreibung der Kenngrößenselektion beinhaltet den Typen des gewählten Verfahrens und gegebenenfalls den Wert des zugehörigen Hyperparameters.

Name	Hyperparameter
Identitätsfunktion	keine
Ranker	- Relevanzfunktion für einzelne Kenngrößen (Relief-Funktion, 1R) - Anzahl der zu selektierenden Kenngrößen
Vorwärtsselektion	Relevanzfunktion für Kenngrößenmengen (Hall-Funktion, konsistenzbasierte Relevanzfunktion)
Rückwärtselimination	Relevanzfunktion für Kenngrößenmengen (Hall-Funktion, konsistenzbasierte Relevanzfunktion)

Tabelle 4.3: Verwendete Verfahren zur Kenngrößenselektion und deren Hyperparameter

Erzeugung von Instanzen

Die Kenngrößenselektion wurde im hier vorgestellten System so implementiert, dass die einzelnen Verfahren nach dem Baukastenprinzip zusammengestellt werden können. Abbildung 4.8 zeigt die statische Struktur dieser Implementierung. Man erkennt, dass der Ranker und die Identitätsfunktion jeweils in eigenen Software-Klassen abgebildet sind. Die Vorwärtsselektion und die Rückwärtselimination ähneln sich in ihrem Ablauf sehr stark. Diese Verfahren werden daher in der Klasse *GreedySearch* zusammengefasst. Der Wert der boolschen Variablen *isForwardSelection* gibt dabei an, welches der beiden Verfahren ausgeführt werden soll.

Während die Identitätsfunktion keine Hyperparameter hat, muss für die anderen drei Verfahren noch die in der Konfigurationsbeschreibung festgelegte Relevanzfunktion berücksichtigt werden. Die Einbindung der Relevanzfunktionen in die Selektionsverfahren erfolgt wieder nach dem *Strategy*-Entwurfsmuster. Dies ermöglicht eine dynamische Kombination der Einzelkomponenten.

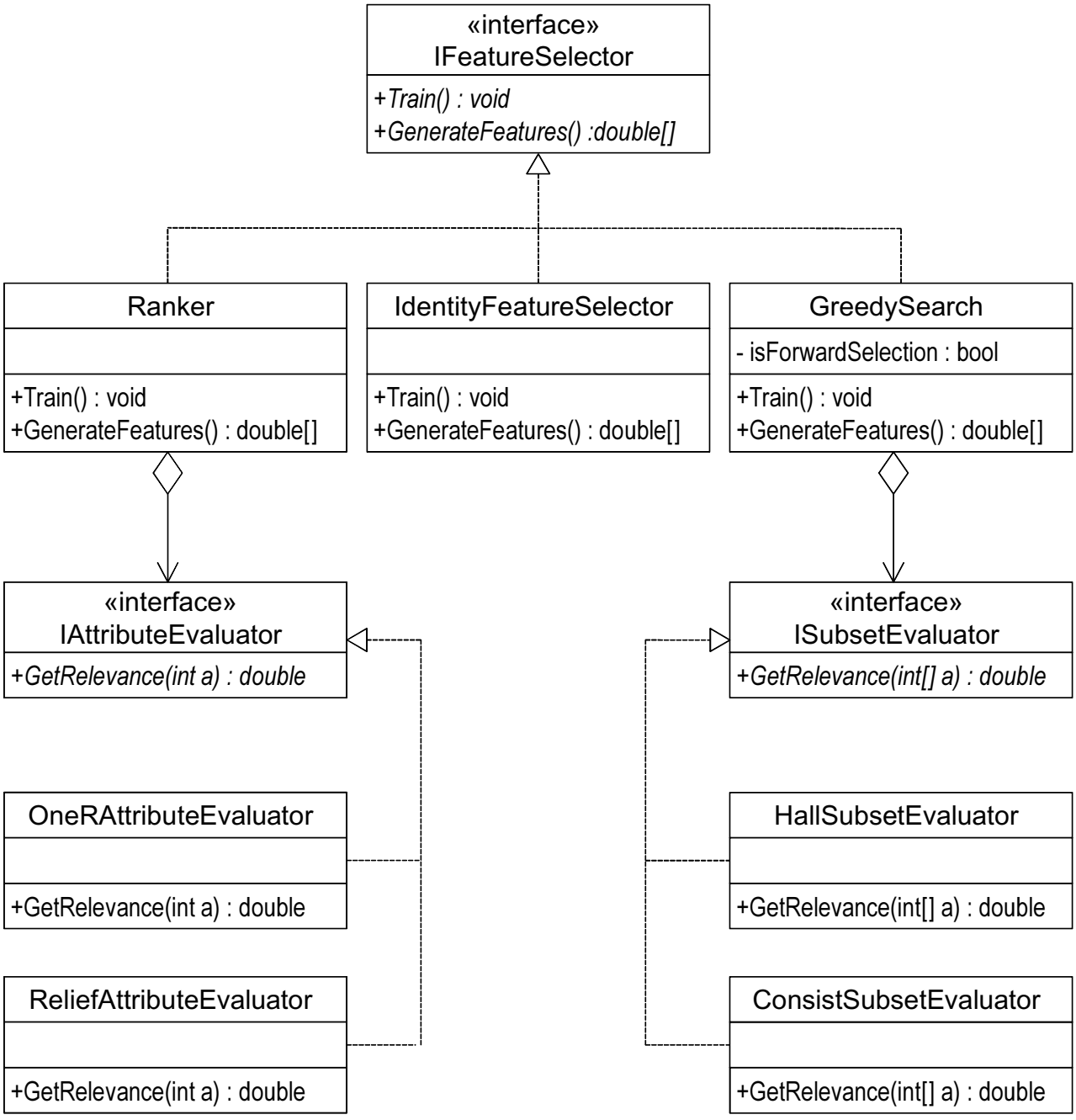


Abbildung 4.8: Statische Struktur der Verfahren für die Kenngrößenselektion

Soll ein Verfahren für die Kenngrößenselektion aus der entsprechenden Konfigurationsbeschreibung erzeugt werden, so muss im ersten Schritt wieder der Typ des gewählten Verfahrens ausgewertet werden. Die Identitätsfunktion kann direkt als Instanz der Klasse *IdentityFeatureSelector* generiert werden. Für die anderen Verfahren muss zunächst eine Instanz der Relevanzfunktion erzeugt werden, die in der Konfigurationsbeschreibung gefordert ist. Die erzeugte Instanz der Relevanzfunktion wird dann dem Software-Objekt des jeweiligen Verfahrens bei dessen Erzeugung übergeben und in dieses eingebunden. Nach dem Einbinden der Relevanzfunktion ist das erzeugte Objekt einsatzbereit.

4.3.4 Realisierung des Klassifikators

Das hier beschriebene System stellt drei Klassifikationsverfahren zu Verfügung: den naiven Bayes-Klassifikator, LogitBoost und die Support Vector Machines. Die Auswahl dieser drei Verfahren erfolgte auf Basis der Arbeiten von Guyon. Sie zeigt, dass sich mit dem naiven Bayes-Verfahren, einem Boosting-Verfahren und einem Kernel-Verfahren alle gängigen Klassifikationsaufgaben sehr gut abdecken lassen [Guyon 2009].

Aufbau der Konfigurationsbeschreibung

Zur Beschreibung des zu verwendenden Klassifikators muss zum einen das Klassifikationsverfahren festgelegt werden. Zum anderen sind bei Bedarf die spezifischen Hyperparameter des gewählten Verfahrens passend zu setzen. Die Werte der Hyperparameter sind ebenfalls in der Konfigurationsbeschreibung enthalten. Die Hyperparameter der einzelnen Klassifikationsverfahren sind in Tabelle 4.4 dargestellt.

Name	**Hyperparameter**
Naive Bayes	keine
LogitBoost	- maximale Höhe der Entscheidungsbäume - Anzahl der Iterationen für das Quasi-Newton-Verfahren - Auswahl Resampling oder Reweighting
Support Vector Machine	- Typ der Support Vector Machine (C-SVM, ν-SVM) - Kernel-Typ (linear, polynomial, Gauß, Sigmoid) - Parameter σ, c und q der einzelnen Kernel

Tabelle 4.4: Verwendete Klassifikationsverfahren und deren Hyperparameter

Erzeugung von Instanzen

Im hier beschriebenen System ist jedes der drei bereitgestellten Klassifikationsverfahren in einer eigenen Klasse implementiert. Diese Klassen sind so gestaltet, dass sie unabhängig von anderen Klassen erzeugt werden können. Somit ist die Instanziierung eines Klassifikators trivial. Es muss lediglich ein Objekt der in der Konfigurationsbeschreibung geforderten Klasse erzeugt werden. Die ebenfalls in der Konfigurationsbeschreibung enthaltenen Hyperparameter werden diesem Objekt direkt bei dessen Erzeugung übergeben.

4.3.5 Training des konfigurierten Analysemodells

Nachdem ein Analysemodell auf Basis seiner Modellbeschreibung instanziiert wurde, ist dieses Modell bereits vollständig konfiguriert (siehe Abschnitt 4.3.1). Vor seinem Einsatz in einem Prozessüberwachungssystem muss das Analysemodell

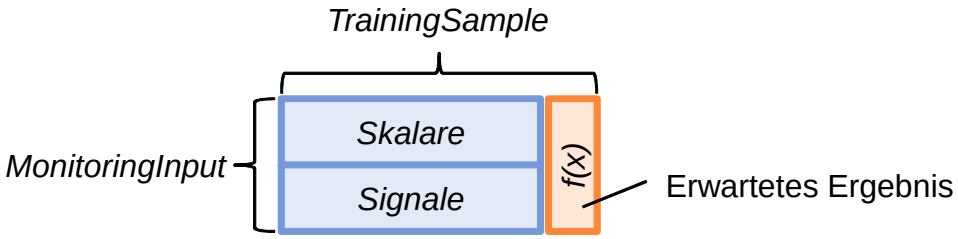


Abbildung 4.9: Aufbau der Lerndaten

aber noch trainiert werden. Hierzu wird das noch untrainierte Analysemodell in das generische Prozessüberwachungssystem eingesetzt. Dabei werden die Platzhalter innerhalb dieses Überwachungssystems mit den Strategie-Objekten belegt, die im instanziierten Analysemodell enthalten sind.

Das generische Überwachungssystem stellt eine Trainingsmethode bereit. Diese Methode übernimmt eine Menge von Lerndatensätzen. Die einzelnen Lerndatensätze sind dabei vom Typ *TrainingSample*. Dieser Datentyp enthält eine Eingabe für die Prognosemethode *GetPrediction*() des Prozessüberwachungssystems mit dem zugehörigen erwarteten Prognoseergebnis. Die enthaltene Eingabe für die Prognosemethode hat den Typ *MonitoringSample*. Dieser Datentyp enthält wiederum Signale und Skalare. Abbildung 4.9 zeigt den Aufbau eines einzelnen Lerndatensatzes.

Das Training des Überwachungssystems mit dem gesetzten Analysemodell läuft nach dem in Abbildung 4.10 dargestellten Schema. Das Verfahren für die Kenngrößenerzeugung ist zu diesem Zeitpunkt bereits vollständig parametriert. Lediglich die Kenngrößenselektion und das Lernmodell müssen noch angepasst werden.

Für die Anpassung der Kenngrößenselektion werden zunächst die Messwerte der Lerndatensätze von den erwarteten Prognoseergebnissen getrennt. Diese

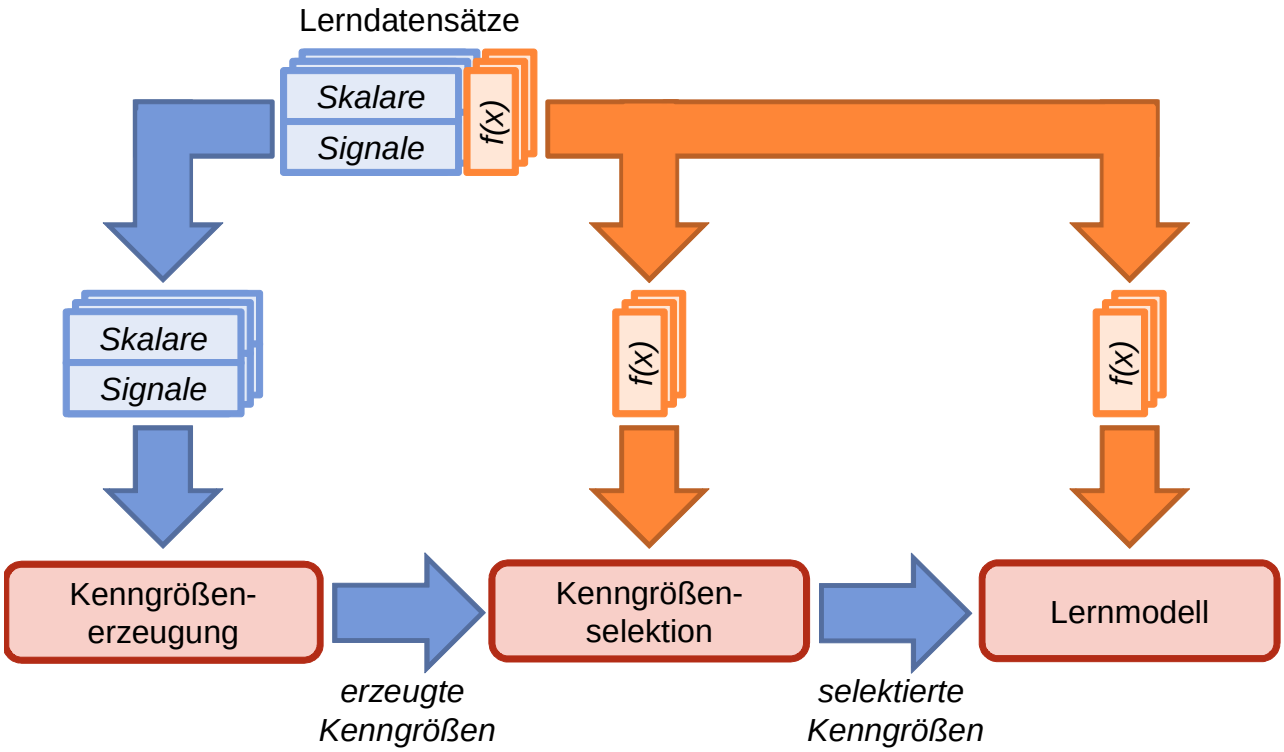


Abbildung 4.10: Datenflüsse während des Trainings eines Überwachungssystems

Messwerte werden an die Kenngrößenerzeugung übergeben. Dort werden für jeden Datensatz einzeln dessen Kenngrößen berechnet. Die so erzeugten Kenngrößen aller Datensätze werden zusammen mit den zugehörigen erwarteten Ergebnissen an die Trainingsmethode der Kenngrößenselektion übergeben. Dort werden die tatsächlich zu verwendenden Kenngrößen bestimmt. Nach diesem Schritt ist die Strategie für die Kenngrößenselektion fertig parametriert.

In einem zweiten Schritt wird die parametrierte Kenngrößenselektion auf die Kenngrößensätze aller Lerndatensätze angewandt. So wird für jeden Lerndatensatz der entsprechende Satz von Kenngrößen ausgewählt, der für die Erzeugung des endgültigen Lernmodells relevant ist. Die so gewonnenen Kenngrößensätze

werden zusammen mit den zugehörigen erwarteten Ergebnissen an die Lernmaschine übergeben. Dort wird aus den entsprechenden Daten letztendlich das fertige Lernmodell erzeugt. Nach Beendigung des Trainings ist das Prozessüberwachungssystem vollständig eingerichtet und einsatzbereit.

4.4 Umsetzung der Modellauswahl

Neben der Entwicklung des generischen Prozessüberwachungssystems lag der Fokus der Forschungsarbeiten auf der Umsetzung der Modellauswahl. Die Modellauswahl verfolgt das Ziel, dasjenige Analysemodell auszuwählen, welches das System am besten an die vorgegebene Problemstellung anpasst. Das Finden dieses Analysemodells ist ein Optimierungsproblem, das mit einem heuristischen Optimierungsverfahren gelöst werden kann.

Das hier vorgestellte System verwendet eine zweistufige Modellauswahl. Hierbei wird das Analysemodell im Rahmen des Optimierungsverfahrens zunächst konfiguriert. Diese Konfiguration entspricht der Auswahl der geeigneten Verfahren für die Kenngrößenerzeugung, die Kenngrößenselektion und die Klassifikation zusammen mit den jeweils passenden Hyperparametern. Im Anschluss an die Konfiguration wird das Analysemodell trainiert. Bei diesem Training werden die Parameter der Kenngrößenselektion und der Klassifikation mit den Standard-Algorithmen des jeweils gewählten Verfahrens bestimmt.

Nachfolgend werden die wesentlichen Aspekte der Modellauswahl des hier beschriebenen Systems beleuchtet.

4.4.1 Auswahl des Optimierungsverfahrens

Wie bereits beschrieben, ist die Auswahl eines geeigneten Analysemodells eine Optimierungsaufgabe. In Abschnitt 2.4.5 wird erklärt, dass sich vor allem po-

pulationsbasierte Optimierungsverfahren sehr gut für die Modellauswahl eignen. Daher soll solch ein Verfahren verwendet werden.

Mit der Partikelschwarm-Optimierung und der Artificial-Bee-Colony-Optimierung zählen momentan zwei populationsbasierte Verfahren zu den leistungsfähigsten heuristischen Optimierungsverfahren überhaupt. Insbesondere die Partikelschwarm-Optimierung ist weit verbreitet und wird für eine Vielzahl von Optimierungsproblemen erfolgreich eingesetzt [Cui u. a. 2005, Elbeltagi u. a. 2005, Poli 2008]. Allerdings setzen die Berechnungen innerhalb der Partikelschwarm-Optimierung voraus, dass der Lösungsraum ein Vektorraum ist. Dies ist bei der vorliegenden Problemstellung nicht der Fall. Hier bilden die möglichen Modellbeschreibungen den Lösungsraum. Diese Modellbeschreibungen sind komplexe Datenstrukturen, die neben einer Vielzahl numerischer, kategorialer und boolscher Werte auch ganze Operatorbäume enthalten. Somit ist die Partikelschwarm-Optimierung für das hier vorliegende Problem nicht geeignet.

Die Artificial-Bee-Colony-Optimierung ist in ihrer Urform ebenfalls auf die rein numerische Optimierung ausgelegt. Eine Analyse des Algorithmus zeigte aber, dass dieses Verfahren auch auf komplexe Datenstrukturen erweitert werden kann. Aus diesem Grund wird eine Artificial-Bee-Colony-Optimierung als Optimierungsverfahren für die Modellauswahl verwendet.

4.4.2 Erweiterung des Optimierungsverfahrens

Das ursprüngliche Artificial-Bee-Colony-Verfahren eignet sich ausschließlich für numerische Optimierungsaufgaben. Da der Lösungsraum der hier beschriebenen Aufgabenstellung aber weit komplexer ist, musste die Artificial-Bee-Colony-Optimierung entsprechend erweitert werden. Die Analyse des in Abschnitt 2.3.2 beschriebenen Algorithmus zeigt, dass im Rahmen der Optimierung auf dem Lösungsraum lediglich die beiden folgenden Operationen ausgeführt werden:

Zufällige Erzeugung: Bei der Initialisierung des Algorithmus und beim Aussenden der Späher werden neue Lösungskandidaten erzeugt und mit Zufallswerten initialisiert.

Erzeugung von Nachbarn: Bei der Auswertung der Futterquellen durch die beschäftigten und unbeschäftigten Arbeiterbienen werden neue Lösungskandidaten erzeugt, die im Lösungsraum zwischen zwei gegebenen Lösungen liegen.

Im Fall der Optimierung in einem Lösungsraum $\mathbb{R}^n$ entspricht die zufällige Erzeugung eines Lösungskandidaten dem Anlegen eines neuen Vektors $\boldsymbol{v} \in \mathbb{R}^n$, dessen Einzelwerte von einem Zufallszahlengenerator gesetzt werden. Die Erzeugung einer benachbarten Lösung ist im ursprünglichen Algorithmus durch das Bilden des arithmetischen Mittels der zwei betrachteten Lösungen realisiert.

Auf Basis obiger Betrachtungen können die Anforderungen an den Lösungsraum verallgemeinert werden. So muss der Lösungsraum einer Artificial-Bee-Colony-Optimierung nicht zwingend reellwertig sein. Es reicht aus zu fordern, dass der Lösungsraum die oben beschriebenen Operationen *zufällige Erzeugung* und *Erzeugung von Nachbarn* unterstützt. Der ursprüngliche Artificial-Bee-Colony-Algorithmus lässt sich mit geringem Aufwand an die so gewonnenen abstrakten Lösungsräume anpassen. Es genügt, den in Abschnitt 2.3.2 beschriebenen Algorithmus an den vier Stellen zu verallgemeinern, an denen Futterquellen zufällig erzeugt oder Nachbarn von Futterquellen ermittelt werden. Man erhält so die erweiterte Artificial-Bee-Colony-Optimierung, die durch Algorithmus 4 beschrieben wird.

Abbildung 4.11 zeigt die statische Grundstruktur, die für die Software-Implementierung der erweiterten Artificial-Bee-Colony-Optimierung gewählt wurde. In diesem Aufbau werden für die Realisierung des abstrakten Lösungsraumes

Algorithmus 4 : Erweiterte Artificial-Bee-Colony-Optimierung

Wähle zufällige Futterquellen $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k \in \mathcal{M}$;
Setze Zähler $t_1 = t_2 = \cdots = t_k = 0$;

while *Abbruchkriterium nicht erfüllt* **do**
  Ermittle beste Lösung $\boldsymbol{x}_{\text{best}} = \arg\max_{\boldsymbol{x} \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, \boldsymbol{x}_{\text{best}}\}} f(\boldsymbol{x})$;
  for $i = 1$ **to** k **do** `// beschäftigte Arbeiterbienen senden`
    Wähle zufällig Futterquelle $\boldsymbol{x}_j \in \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k\}$;
    Ermittle gemeinsamen Nachbarn $\boldsymbol{x}_i'$ von $\boldsymbol{x}_i$ und $\boldsymbol{x}_j$;
    if $f(\boldsymbol{x}_i') > f(\boldsymbol{x}_i)$ **then**
      Setze $\boldsymbol{x}_i = \boldsymbol{x}_i'$; Setze $t_i = 0$;
    else
      Erhöhe t_i um 1;
    end
  end
  Berechne Besuchswahrscheinlichkeiten $p_i = \frac{f(\boldsymbol{x}_i)}{\sum_{j=1}^{k}(\boldsymbol{x}_j)}$ für $i = 1, \ldots, k$;
  for $i = 1$ **to** $\frac{k}{2}$ **do** `// unbeschäftigte Arbeiterbienen senden`
    Wähle zufällig $\boldsymbol{x}_r$ abhängig von den Wahrscheinlichkeiten $p_1, \ldots, p_k$;
    Wähle zufällig Futterquelle $\boldsymbol{x}_j \in \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k\}$;
    Ermittle gemeinsamen Nachbarn $\boldsymbol{x}_r'$ von $\boldsymbol{x}_r$ und $\boldsymbol{x}_j$;
    if $f(\boldsymbol{x}_r') > f(\boldsymbol{x}_r)$ **then**
      Setze $\boldsymbol{x}_r = \boldsymbol{x}_r'$; Setze $t_r = 0$;
    else
      Erhöhe t_r um 1;
    end
  end
  for $i = 1$ **to** k **do** `// Späher aussenden`
    if $t_i > maxNumTrials$ **then**
      Ersetze $\boldsymbol{x}_i$ durch zufällige Futterquelle $\boldsymbol{x} \in \mathcal{M}$; Setze $t_i = 0$;
    end
  end
end

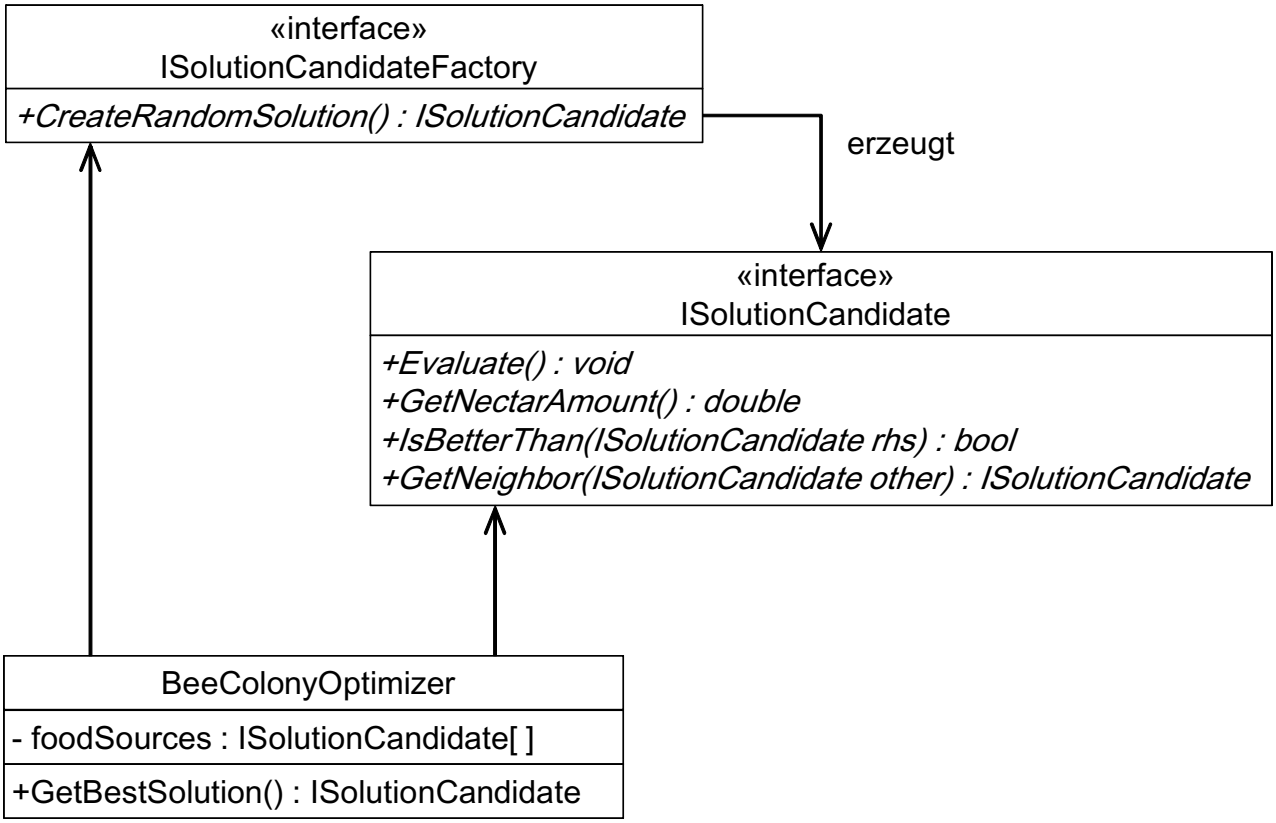


Abbildung 4.11: Statische Grundstruktur des erweiterten Artificial-Bee-Colony-Optimierers

zwei Schnittstellen bereitgestellt. Die Schnittstelle *ISolutionCandidate* definiert die Methode *GetNeighbor*() für die Erzeugung von Nachbarn. Die Erzeugung zufälliger Lösungskandidaten wurde nach dem *Abstract-Factory*-Erzeugungsmuster [Gamma 1995] realisiert. Dementsprechend ergibt sich eine zweite Schnittstelle *ISolutionCandidateFactory*, welche die Methode *CreateRandomSolution()* für die Erzeugung zufälliger Lösungskandidaten definiert.

4.4.3 Erzeugung und Modifikation von Lösungen

Der Lösungsraum der hier vorgestellten Modellauswahl ist die Menge aller Modellbeschreibungen. Im vorangegangenen Abschnitt wird der erweiterte Artificial-Bee-Colony-Optimierer beschrieben, der für die Modellauswahl entwickelt wurde. Dieser Optimierungsalgorithmus setzt voraus, dass auf dem Lösungsraum Operationen für die Erzeugung zufälliger Lösungen und für die Erzeugung von Nachbarn einer Lösung definiert sind. Nachfolgend wird beschrieben, wie diese Operationen auf der Menge der Modellbeschreibungen realisiert wurden.

Umsetzung in der Modellbeschreibung

Ein Analysemodell verwaltet die von ihm benötigten Daten in einer Modellbeschreibung. Diese Modellbeschreibung setzt sich aus den Konfigurationsbeschreibungen der drei Einzelkomponenten des Analysemodells zusammen. Neben diesen drei Konfigurationsbeschreibungen beinhaltet die Modellbeschreibung keine weiteren Daten. Entsprechend benötigen die geforderten Operationen für die Erzeugung zufälliger Lösungen und für die Erzeugung von Nachbarn kaum eigene Logik. Vielmehr werden die wesentlichen Aufgaben an die enthaltenen Konfigurationsbeschreibungen weitergereicht. Nachfolgend wird für die beiden Operationen die in der Modellbeschreibung verbliebene Programmlogik beschrieben.

Zufällige Erzeugung: Eine zufällig konfigurierte Modellbeschreibung wird erzeugt, indem für jede der drei Teilkomponenten – Kenngrößenerzeuger, Kenngrößenselektion und Klassifikation – eine zufällige Konfigurationsbeschreibung erzeugt wird. Die zufällig erzeugten Konfigurationsbeschreibungen werden zu der gewünschten zufälligen Modellbeschreibung kombiniert.

Erzeugung von Nachbarn: In der erweiterten Artificial-Bee-Colony-Optimierung werden zwei Modellbeschreibungen ausgewählt, für die ein gemeinsa-

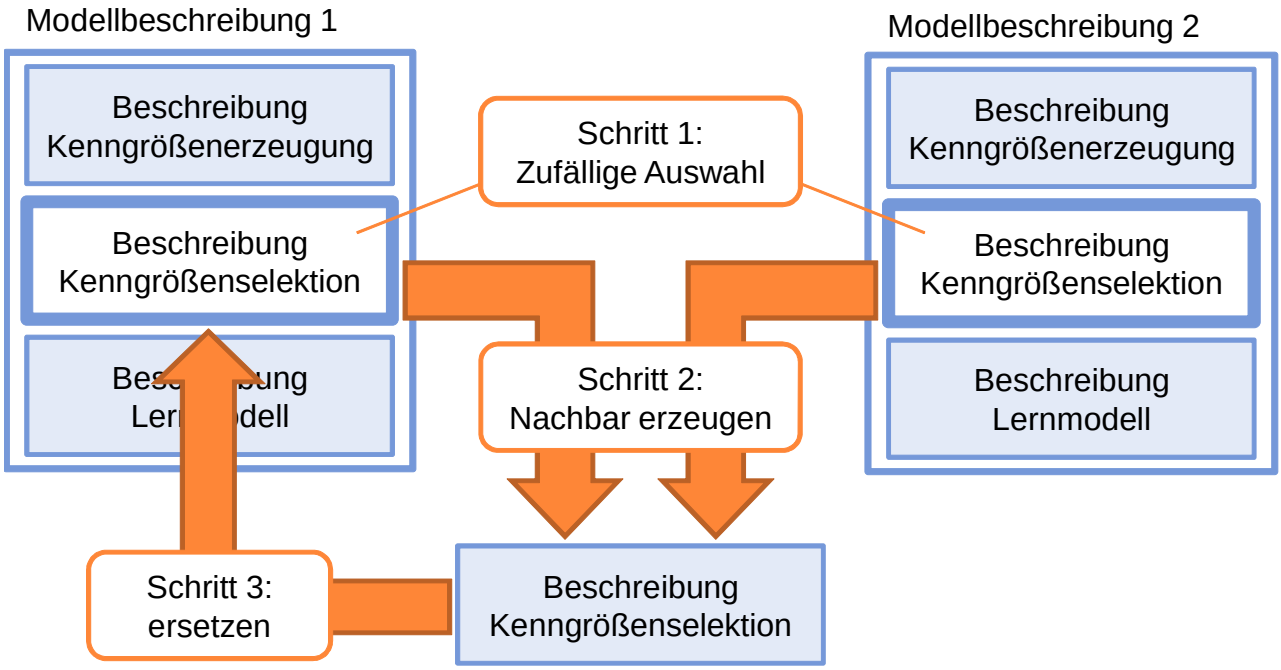


Abbildung 4.12: Vorgehen beim Erzeugen von Nachbarn einer Modellbeschreibung

mer Nachbar bestimmt werden soll (siehe Algorithmus 4). Die erste dieser Modellbeschreibungen entspricht der aktuell besuchten Lösung. Die zweite Modellbeschreibung wird im Algorithmus zufällig gewählt. Bei der Erzeugung eines gemeinsamen Nachbarn wird eine Modellbeschreibung gesucht, die im Lösungsraum zwischen den beiden ausgewählten Modellbeschreibungen liegt.

Der gesuchte gemeinsame Nachbar wird im hier vorgestellten Ansatz erzeugt, indem eine der drei enthaltenen Konfigurationsbeschreibungen der aktuell besuchten Lösung an die entsprechende Konfigurationsbeschreibung der zweiten Modellbeschreibung angepasst wird. Hierzu sieht das Konzept vor, dass für jede dieser Konfigurationsbeschreibungen gesondert ein Nachbar erzeugt werden kann. Die Modellbeschreibung wählt nun zufallsbasiert

die zu modifizierende Konfigurationsbeschreibung. Die gewählte Konfigurationsbeschreibung wird dann durch deren Nachbarn ersetzt. Dieser Ablauf wird in Abbildung 4.12 dargestellt.

Nachfolgend wird beschrieben, wie die zufällige Erzeugung und die Erzeugung von Nachbarn für die einzelnen Konfigurationsbeschreibungen umgesetzt wurden.

Umsetzung in der Beschreibung des Kenngrößenerzeugers

Wie bereits in Abschnitt 4.3.2 beschrieben, stehen für den Kenngrößenerzeuger zwei verschiedene Verfahren zur Auswahl. Die Identitätsfunktion ist der einzige Kenngrößenerzeuger, der bereits beim Start des Konfigurationssystems vorhanden ist. Dieser Kenngrößenerzeuger bildet den Kenngrößensatz, indem er einfach alle Messwerte des Eingangsdatensatzes unverändert in das Ausgabefeld schreibt. Alle anderen Kenngrößenerzeuger werden im Rahmen der Modellauswahl dynamisch programmiert und zur Laufzeit in das System eingebunden.

Im hier vorgestellten System beinhaltet die Konfigurationsbeschreibung des Kenngrößenerzeugers zum einen den gewünschten Typen des Kenngrößenerzeugers (Identitätsfunktion oder dynamisch programmiert). Zum anderen enthält sie eine Menge von Operatorbäumen, die den aktuellen Stand des dynamisch programmierten Kenngrößenerzeugers repräsentiert. Diese Operatorbaum-Menge wird nur dann berücksichtigt, wenn als Typ tatsächlich die dynamisch programmierten Kenngrößenerzeuger ausgewählt sind. Die Methoden für die zufällige Erzeugung dieser Konfigurationsbeschreibungen und für die Erzeugung der Nachbarn werden nachfolgend beschrieben.

Zufällige Erzeugung: Für die zufällige Erzeugung wird zunächst der Typ des Kenngrößenerzeugers zufällig gewählt. Anschließend wird eine zufällige Menge von Operatorbäumen generiert. Hierzu wird zunächst auf Zufallsbasis

entschieden, wie viele Operatorbäume die Konfigurationsbeschreibung beinhalten soll. Auch die maximale Höhe der Operatorbäume wird über eine Zufallszahl festgelegt. Abschließend werden die einzelnen Operatorbäume aus zufällig gewählten Funktionen und Terminalen aufgebaut.

Erzeugung von Nachbarn: Ein Nachbar einer bestehenden Konfigurationsbeschreibung eines Kenngrößenerzeugers wird generiert, indem diese Konfigurationsbeschreibung in Richtung einer zweiten Konfigurationsbeschreibung bewegt wird. Im Folgenden wird die aktuell zu modifizierende Konfigurationsbeschreibung des Kenngrößenerzeugers mit F_1 bezeichnet. Die oben genannte zweite Konfigurationsbeschreibung wird F_2 genannt.

Haben F_1 und F_2 verschiedene Typen von Kenngrößenerzeugern, dann wird F_1 modifiziert, indem der Typ von F_2 übernommen wird. Verwenden beide Typen bereits den gleichen Typen von Kenngrößenerzeugern, dann wird die Menge der Operatorbäume von F_1 verändert. Hierzu stehen fünf verschiedene Operationen bereit, von denen eine zufällig ausgewählt und ausgeführt wird. Dabei sind zwei der bereitgestellten Operationen unabhängig von F_2. Die anderen drei Operationen sind abhängig von F_2. Die fünf Operationen werden nachfolgend beschrieben.

Operatorbaum entfernen: Der gesuchte Nachbar wird erzeugt, indem ein zufällig gewählter Operatorbaum aus F_1 entfernt wird. Diese Operation ist unabhängig von F_2.

Operatorbaum mutieren: Der gesuchte Nachbar wird erzeugt, indem ein beliebiges Terminal eines zufällig gewählten Operatorbaumes aus F_1 mutiert wird. Terminale, die durch Bereichsfunktionen definiert sind, werden durch zufälliges Ändern des zu berücksichtigenden Signalbereichs mutiert. Alle anderen Typen von Terminalen werden bei

der Mutation durch ein zufällig erzeugtes Terminal ersetzt. Diese Operation ist unabhängig von F_2.

Baumanzahl anpassen: Der gesuchte Nachbar wird erzeugt, indem die Anzahl der Operatorbäume von F_1 an die Anzahl der Operatorbäume von F_2 angepasst wird. Beinhaltet F_1 mehr Operatorbäume als F_2, dann werden entsprechend viele zufällig gewählte Operatorbäume aus F_1 gelöscht. Enthält F_1 weniger Operatorbäume als F_2, dann werden entsprechend viele Operatorbäume in F_1 zufällig erzeugt.

Operatorbaum übernehmen: Der gesuchte Nachbar wird erzeugt, indem ein willkürlich gewählter Operatorbaum aus F_2 in F_1 übernommen wird.

Operatorbäume kreuzen: Der gesuchte Nachbar wird erzeugt, indem ein willkürlich gewählter Operatorbaum B_1 aus F_1 mit einem ebenfalls willkürlich gewählten Operatorbaum B_2 aus F_2 gekreuzt wird. In B_1 wird nun zufallsbasiert ein Knoten K_1 ausgewählt, in B_2 ein Knoten K_2. Nun wird ein neuer Operatorbaum erzeugt, indem zunächst B_1 kopiert wird. Anschließend wird im kopierten Operatorbaum der Teilbaum, der am Knoten K_1 beginnt, ersetzt durch den Teilbaum aus B_2, der am Knoten K_2 beginnt (siehe Abbildung 4.13). Dieses Vorgehen entspricht der Teilbaum-Kreuzung aus der genetischen Programmierung (siehe Abschnitt 2.3.3). Der neu erzeugte Operatorbaum ersetzt im erzeugten Nachbarn den Operatorbaum B_1.

Umsetzung in der Beschreibung der Kenngrößenselektion

Die Konfigurationsbeschreibung der Kenngrößenselektion besteht aus dem Typen des gewählten Selektionsverfahrens und aus den Werten der Hyperparameter aller

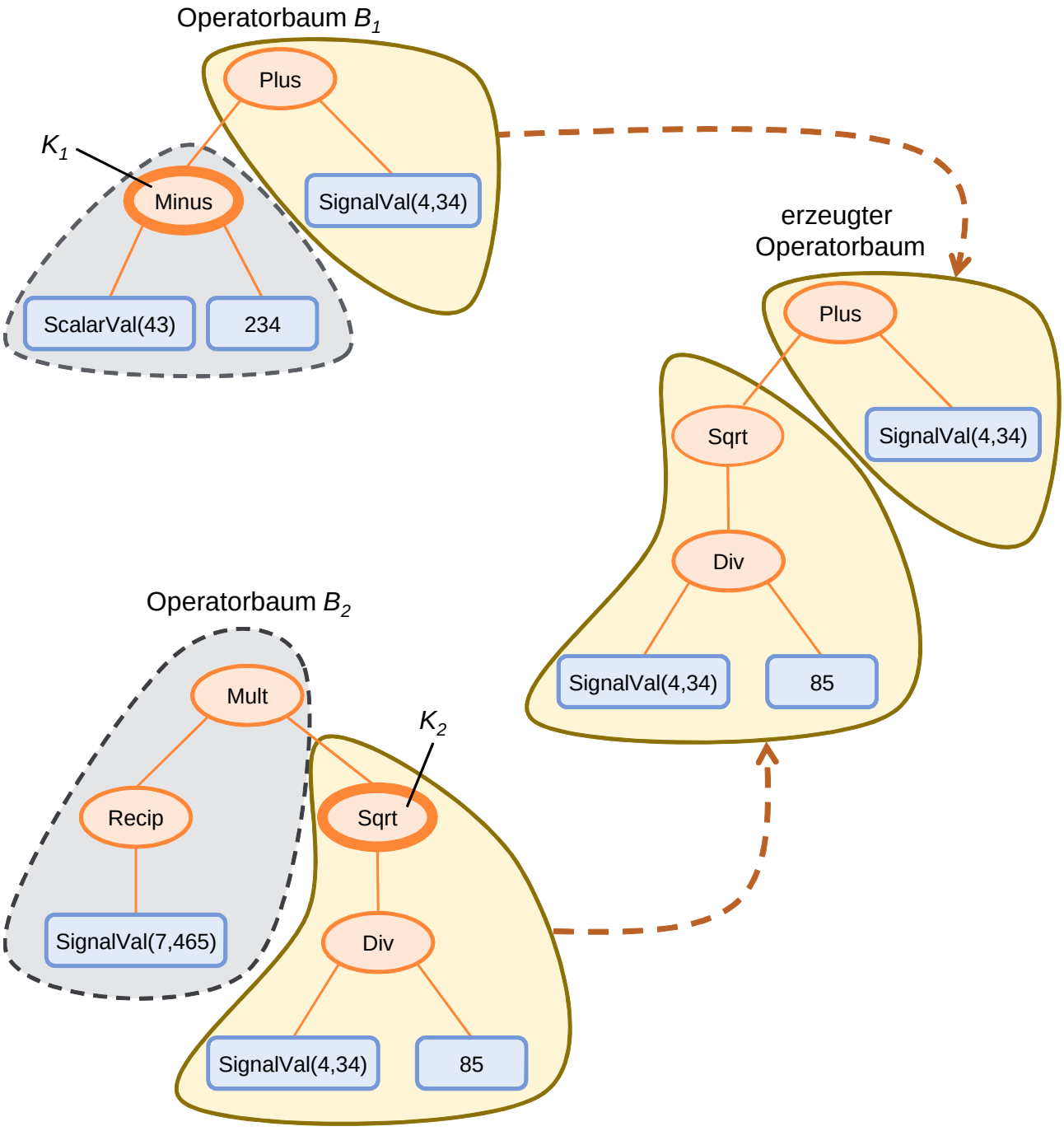


Abbildung 4.13: Prinzip des Kreuzens von Operatorbäumen

verfügbaren Selektionsverfahren. Die unterstützten Selektionsverfahren sowie deren Hyperparameter sind in Tabelle 4.3 dargestellt. Nachfolgend wird beschrieben, wie die Methoden für die zufällige Erzeugung und Modifikation solcher Konfigurationsbeschreibungen umgesetzt sind.

Zufällige Erzeugung: Für die zufällige Erzeugung einer Konfigurationsbeschreibung wird zunächst der Typ der Kenngrößenselektion zufällig gewählt (Identitätsfunktion, Ranker, Vorwärtsselektion, Rückwärtselimination). Die Hyperparameter aller Selektionsverfahren werden mit Zufallswerten aus deren jeweiligem Wertebereich belegt.

Erzeugung von Nachbarn: Für die Erzeugung eines Nachbarn wird eine zu modifizierende Konfigurationsbeschreibung S_1 in Richtung einer zweiten Konfigurationsbeschreibung S_2 bewegt. Dies geschieht wie folgt: Haben S_1 und S_2 verschiedene Typen von Kenngrößenselektionen, so wird der Typ von S_2 in S_1 übernommen. Andernfalls wird willkürlich einer der Hyperparameter des aktuell gewählten Verfahrens ausgewählt. Dieser Hyperparameter wird in S_1 auf den in S_2 enthaltenen Wert gesetzt.

Umsetzung in der Beschreibung des Klassifikators

Analog zur Konfigurationsbeschreibung der Kenngrößenselektion enthält auch die Konfigurationsbeschreibung des zu verwendenden Klassifikators im Wesentlichen den Typen des gewählten Klassifikationsverfahrens sowie die Hyperparametersätze aller verfügbaren Klassifikationsverfahren. Die bereitgestellten Klassifikationsverfahren und deren Hyperparameter sind in Tabelle 4.4 aufgelistet. Die Konfigurationsbeschreibung eines Klassifikators kann wie folgt erzeugt und modifiziert werden:

Zufällige Erzeugung: Eine zufällige Konfigurationsbeschreibung wird erzeugt, indem zunächst der Typ des zu verwendenden Klassifikationsverfahrens willkürlich gesetzt wird. Anschließend werden die Hyperparameter aller Klassifikationsverfahren mit Zufallswerten aus deren jeweiligem Wertebereich belegt.

Erzeugung von Nachbarn: Für die Erzeugung eines Nachbarn wird eine zu modifizierende Konfigurationsbeschreibung C_1 in Richtung einer zweiten Konfigurationsbeschreibung C_2 bewegt. Dies geschieht wie folgt: Haben C_1 und C_2 verschiedene Klassifikator-Typen, so wird der Typ von C_2 in C_1 übernommen. Andernfalls wird willkürlich ein Hyperparameter des aktuell gewählten Verfahrens ausgewählt. Dieser Hyperparameter wird in C_1 auf den entsprechenden Wert aus C_2 gesetzt.

4.4.4 Zielfunktion der Optimierung

Ziel der Modellauswahl ist es, zu einem Fertigungsprozess das beste Analysemodell zu finden. Das beste Analysemodell ist dabei dasjenige, mit dem das generische Prozessüberwachungssystem mit bisher unbekannten Eingangsdaten die beste Vorhersagegüte erreicht. Somit ist die Verallgemeinerungsfähigkeit des resultierenden Prozessüberwachungssystems das wesentliche Gütekriterium eines Analysemodells.

Das hier beschriebene System verwendet zur Bewertung dieser Verallgemeinerungsfähigkeit ein Maß, das auf Basis eines Verifikationsdatensatzes $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$ berechnet wird. In dieses Maß fließt zunächst das empirische Risiko (siehe Glei-

chung 2.29) ein. Als Schadensfunktion verwendet das System hierbei die diskrete Schadensfunktion $L_d : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_0^+$, die gegeben ist durch

$$L_d(y_1, y_2) := \begin{cases} 0 & \text{falls } y_1 = y_2 \\ 1 & \text{falls } y_1 \neq y_2 \end{cases} \qquad \text{für alle } y_1, y_2 \in \mathcal{Y}. \tag{4.1}$$

Der mit L_d ermittelte Risikowert E entspricht der *Fehlerquote*, also dem relativen Anteil von Fehlprognosen an der Menge aller Prognosen. Die in der Praxis oft verwendete *Trefferquote* T lässt sich aus diesem Wert leicht ableiten. Es gilt

$$T = 1 - E. \tag{4.2}$$

Im Rahmen dieser Arbeit durchgeführte Untersuchungen zeigen, dass die alleinige Verwendung der Trefferquote insbesondere bei stark unbalancierten Verifikationsdaten zur Überanpassung führt. Unter unbalancierten Daten versteht man eine Menge von Datensätzen $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$, in denen die möglichen Prognoseergebnisse $c_1, \ldots, c_z \in \mathcal{Y}$ unterschiedlich oft vorkommen. Aus diesem Grund wertet das hier beschriebene System zusätzlich die *balancierte Trefferquote* T_B aus. Zur Berechnung der balancierten Trefferquote werden die Verifikationsdaten entsprechend ihres Prognoseergebnisses aufgeteilt. Enthalten die Verifikationsdaten also die Prognoseergebnisse $c_1, \ldots, c_z \in \mathcal{Y}$, so wird die Datensatzmenge $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$ in z paarweise disjunkte Teilmengen

$$\mathcal{D}_i := \{(\boldsymbol{x}, c) \in \mathcal{D} \mid c = c_i) \qquad \text{mit } i = 1, \ldots, z \tag{4.3}$$

aufgeteilt. Für jede dieser z Mengen wird die Trefferquote gesondert ermittelt. Die balancierte Trefferquote ist dann die mittlere Trefferquote über alle z Mengen [Guyon u. a. 2006a].

Die balancierte Trefferquote T_B fließt als Regularisierungsterm in das Gütemaß mit ein. Ist $\mathcal{M}$ die Menge aller möglichen Modellbeschreibungen, dann ergibt sich

die Zielfunktion $f : \mathcal{M} \to \mathbb{R}$ des hier vorgestellten Systems als gewichtete Summe der unbalancierten und der balancierten Trefferquote. Es gilt

$$f(x) := T + \gamma \cdot T_B \qquad \text{für alle } x \in \mathcal{M}. \tag{4.4}$$

Der frei wählbare Parameter γ gewichtet die beiden Trefferquoten. Über ihn lässt sich die Zielfunktion an die Beschaffenheit der verfügbaren Lerndaten anpassen.

Modelle, die auf Basis einer einfachen Validierung ausgewählt werden, haben meist hohe Varianzen. Deshalb wird das Gütemaß in einer k-fachen stratifizierten Kreuzvalidierung berechnet (siehe Abschnitt 2.4.4). Die Anzahl k_v der Validierungsdurchgänge ist dabei ebenfalls parametrierbar.

4.4.5 Parameter der Modellauswahl

Ein großer Vorteil der Artificial-Bee-Colony-Optimierung ist, dass das Verfahren nur sehr wenige Parameter hat. So ist lediglich zu spezifizieren, wie viele beschäftigte Arbeiterbienen ausgesandt werden und wie oft Futterquellen ohne Verbesserung besucht werden, bevor man sie aufgibt.

Untersuchungen am fertigen System zeigten, dass die Wahl der beiden oben genannten Parameter keinen nennenswerten Einfluss auf die Güte der Optimierungsergebnisse hat. Aus diesem Grund werden die Werte dieser Parameter für die Modellauswahl fest auf die von Karaboga vorgeschlagenen Werte gesetzt [Karaboga 2005]. Die Anzahl n der beschäftigten Arbeiterbienen hat somit den konstanten Wert 20, die Anzahl *maxNumTrials* der Besuche einer nicht verbesserten Futterquelle den konstanten Wert 10.

Somit verbleiben lediglich drei freie Parameter, die für die Einrichtung der Modellauswahl gesetzt werden müssen. Diese Parameter werden nachfolgend beschrieben.

Anzahl der Validierungsdurchgänge: Der Parameter k_v bestimmt die Anzahl der Validierungsdurchgänge für die intern durchgeführte Kreuzvalidierung. Der Standardwert dieses Parameters ist 10.

Gewichtung der Trefferquoten: Der Parameter γ_t regelt die Gewichtung der unbalancierten und der balancierten Trefferquote bei der Berechnung der Modellgüte. Der Wert dieses Parameters ist standardmäßig auf $\frac{1}{5}$ gesetzt.

Abbruchkriterium: Der Parameter n_max definiert das Abbruchkriterium. Die Artificial-Bee-Colony-Optimierung wird abgebrochen, wenn sich die Modellgüte n_max Iterationen lang nicht verbessert. Dieser Parameter hat im hier vorgestellten System den Standardwert 10.

Kapitel 5

Validierung der Leistungsfähigkeit des Verfahrens

Im Folgenden wird die Leistungsfähigkeit des hier vorgestellten Verfahrens zur automatisierten Erzeugung von Prozessüberwachungssystemen beleuchtet. Hierzu werden zwei Fertigungsverfahren herangezogen: das Ultraschallschweißen und das Spritzgießen von Kunststoffen. Zu beiden Fertigungsverfahren existieren bereits leistungsfähige Prozessüberwachungssysteme. Diese Überwachungssysteme wurden von Experten in jeweils mehrjähriger Forschungsarbeit erstellt. Für die Validierung werden mit dem neuen Verfahren weitere Prozessüberwachungssysteme automatisch generiert und mit den von Experten erstellten Systemen verglichen.

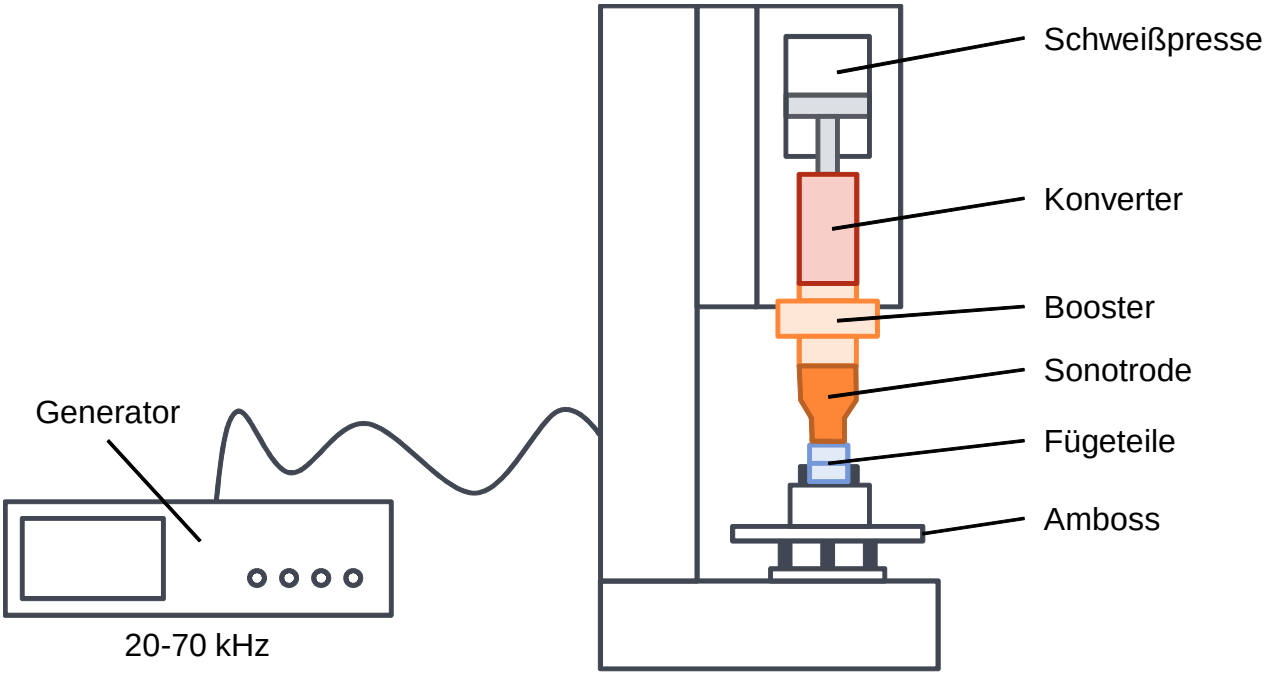


Abbildung 5.1: Komponenten einer Ultraschallschweißmaschine [Ehrenstein 2004]

5.1 Anwendung beim Ultraschallschweißen

5.1.1 Beschreibung des Fertigungsverfahrens

Beim Ultraschallschweißen werden thermoplastische Kunststoffe gefügt. Hierzu werden die zu verschweißenden Kunststoffteile gegeneinander gepresst und mittels Ultraschallenergie an den Fügeflächen aufgeschmolzen. Nach Abschalten des Ultraschalls wird der Anpressdruck zunächst aufrechterhalten. Der aufgeschmolzene Kunststoff erkaltet nun unter Druck und bildet dabei eine feste Schweißverbindung [Haberstroh u. a. 2004, Ehrenstein 2004].

Eine Ultraschallschweißmaschine besteht aus den in Abbildung 5.1 dargestellten Hauptkomponenten. Diese Komponenten haben im Einzelnen folgende Funk-

tionen: Der *Generator* erzeugt elektrische Schwingungen im Frequenzbereich von 20 kHz bis 70 kHz. Diese elektrischen Schwingungen werden im *Konverter* in mechanische Schwingungen mit gleicher Frequenz umgewandelt. Über das Transformationsstück (*Booster*) und die *Sonotrode* werden die mechanischen Schwingungen den Fügeteilen zugeleitet. Die *Schweißpresse* baut den Schweißdruck auf. Der *Amboss* ist das Aufnahmewerkzeug für die Fügeteile [Ehrenstein 2004].

Eine Ultraschallschweißung läuft wie folgt ab: Zunächst werden die Fügeteile in den Amboss eingelegt. Anschließend fährt die Schweißpresse die Sonotrode auf die Fügeteile. Nachdem die Sonotrode auf dem oberen Fügeteil aufliegt, wird der Generator eingeschaltet. Die vom Konverter erzeugte mechanische Schwingung wird über das Schwinggebilde aus Booster und Sonotrode in die Fügeteile geleitet. Durch die eingebrachte Energie schmelzen die Fügepartner an den Fügeflächen auf. Die entstehende Schmelze gibt dem von der Schweißpresse erzeugten Anpressdruck nach. Die dadurch entstehende messbare Verfahrbewegung des Schwinggebildes nennt man den *Schweißweg*. Nach Abschalten des Ultraschalls wird der Anpressdruck noch eine vorgegebene Zeit lang aufrechterhalten. In dieser *Haltezeit* erkaltet die Schmelze wieder und bildet eine feste Schweißnaht. Mit dem Ende der Haltezeit fährt die Schweißpresse das Schwinggebilde wieder zurück in die Ausgangsposition [Ehrenstein 2004, Neher 2012].

Der Schweißvorgang muss speziell an jeden Prozess angepasst werden. Dies geschieht durch die Wahl folgender Parameter: Amplitude der Ultraschallschwingungen an der Sonotrode, Schweißdauer, Haltedruck und Haltezeit. Die Wahl der Parameter und die daraus resultierenden Prozessbedingungen haben dabei großen Einfluss auf die Güte des Schweißvorgangs. Die wesentlichen Gütekriterien einer Schweißung sind die Dichtheit und Festigkeit der Schweißnaht sowie deren optisches Erscheinungsbild [Haberstroh u. a. 2004, Neher u. a. 2010].

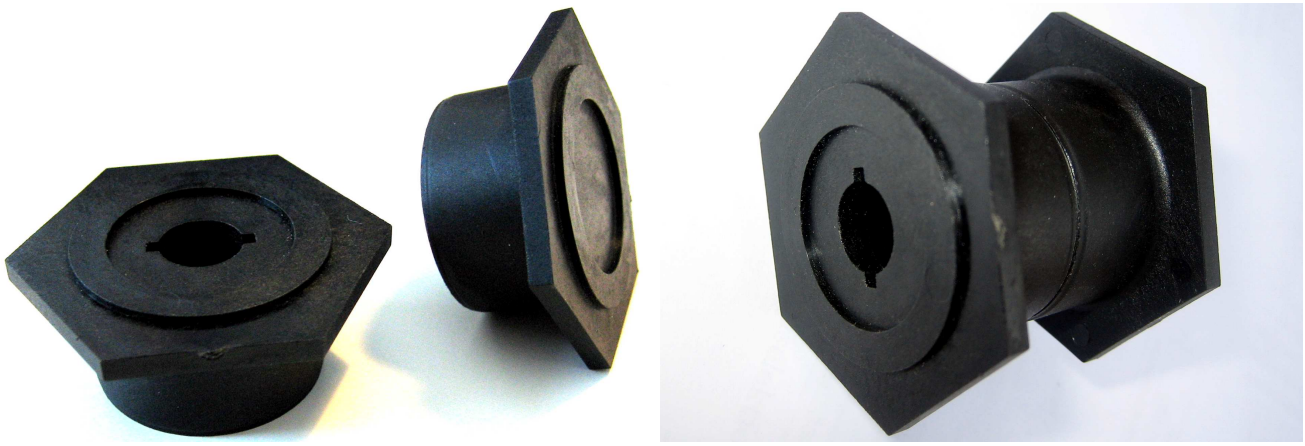
5.1.2 Vergleichssysteme

Das erste der hier vorgestellten Prozessüberwachungssysteme für das Ultraschallschweißen wurde in den beiden aufeinander aufbauenden Verbundforschungsprojekten QP-UVS und SC-QUPUS erarbeitet. Im Rahmen dieser Projekte wurden auf Basis aufwändiger Untersuchungen aussagekräftige Prozesskenngrößen ermittelt. Zudem wurden geeignete Verfahren für die Kenngrößenselektion und die Klassifikation ausgewählt. Der Gesamtaufwand für beide Projekte betrug 140 Personenmonate verteilt auf Projektlaufzeiten von insgesamt sechs Jahren [Schmidberger u. a. 2005a, Neher u. a. 2010]. Im Folgenden wird dieses Überwachungssystem als SC-QUPUS-*System* bezeichnet.

Die Kenngrößen werden im SC-QUPUS-System aus den Signalverläufen von Schweißweg, Amplitude an der Sonotrode, Leistungsaufnahme des Generators, Haltedruck, Fügekraft und Fügegeschwindigkeit gebildet. Zudem wird auch der Körperschall in den Schweißteilen erfasst und ausgewertet. In Versuchen konnte allerdings kein Zusammenhang zwischen dem Verlauf des Körperschalls und der Güte der Schweißnaht erkannt werden [Neher u. a. 2010].

Zur Kenngrößenselektion wird ein Filterverfahren eingesetzt. Die dabei verwendete Relevanzfunktion ähnelt der Relief-Funktion (siehe Abschnitt 2.2.2) sehr stark [Neher u. a. 2010, Neher 2012]. Zur Klassifikation auf Basis der selektierten Kenngrößen wird im SC-QUPUS-System die *Learning Vector Quantization (LVQ)* verwendet. Hierbei handelt es sich um einen Typen von neuronalen Netzen, der als Klassifikator verwendet werden kann [Kohonen 1990, Neher u. a. 2010].

Im Rahmen einer Dissertation erweiterte Neher das SC-QUPUS-System nochmals. Ein wesentlicher Beitrag war dabei die Implementierung eines Regressionsverfahrens zur Prognose der Prozessgüte. Dieses Regressionsverfahren mit dem Namen *General Regression Neuro-Fuzzy Network (GRNFN)* gehört zur Klasse der



(a) Einzelteile des Probekörpers (b) Verschweißter Probekörper

Abbildung 5.2: Der Probekörper aus [Neher u. a. 2010]

Neuro-Fuzzy-Verfahren [Neher 2012]. Das von NEHER erstellte Überwachungssystem wird nachfolgend NEHER-*System* genannt.

5.1.3 Verwendete Daten

Die oben beschriebenen Prozessüberwachungssysteme für das Ultraschallschweißen sollen nachfolgend mit automatisch generierten Überwachungssystemen verglichen werden. Für diesen Vergleich werden Datensätze aus zwei Versuchsreihen verwendet, anhand derer sowohl in [Neher u. a. 2010] als auch in [Neher 2012] die Leistungsfähigkeit der jeweiligen Systeme nachgewiesen wurde. Diese Versuchsreihen werden im Folgenden beschrieben.

Daten aus Schweißungen mit Probekörpern

In den Verbundforschungsprojekten QP-UVS und SC-QUPUS erfolgten die Untersuchungen zunächst auf Basis eigens entwickelter Probekörper. Diese Probekör-

per wurden so konstruiert, dass reproduzierbare Schweißvorgänge möglich sind. Gleichzeitig ermöglicht die Geometrie der Probekörper die Durchführung von Zug- und Berstdruckprüfungen in entsprechenden Prüfanlagen [Schmidberger u. a. 2005a, Neher u. a. 2010, Neher 2012]. Abbildung 5.2 zeigt die beiden Einzelteile des Probekörpers und einen geschweißten Probekörper.

Die Datensätze aus *Versuchsreihe A* wurden bei Schweißversuchen mit diesem Probekörper erfasst. Insgesamt wurden 47 Probekörper verschweißt. Dabei wurden folgende Größen als Zeitsignale erfasst: der Schweißweg, die Fügekraft, die Schwingungsamplitude der Sonotrode, die Leistungsaufnahme des Ultraschallgenerators, die Schweißkraft, der Körperschall an der Werkstückaufnahme und diverse Trigger. Die Signale wurden dabei während des gesamten Schweißzyklus mit 2 kHz abgetastet. Da die anregende Schweißfrequenz aber 20 kHz beträgt, wurden die Signale zusätzlich mit 45 kHz abgetastet, solange der Ultraschall aktiv war [Neher u. a. 2010, Neher 2012]. Die Abbildungen 5.3 und 5.4 zeigen beispielhaft die Signalverläufe während eines einzelnen Schweißzyklus.

Die in den Schweißversuchen erzeugten Probekörper wurden anschließend einer Berstdruckprüfung unterzogen. Der dabei gemessene Berstdruck wird dem entsprechenden Schweißzyklus als Qualitätsmerkmal zugeordnet. Die erfassten Druckwerte liegen hierbei im Bereich zwischen 6,5 bar und 12,8 bar [Neher u. a. 2010, Neher 2012]. Anhand der gemessenen Berstdrücke werden die geschweißten Probekörper in zwei Güteklassen eingeteilt: *gut* (Berstdruck > 8 bar) und *schlecht* (Berstdruck ≤ 8 bar).

In Summe beinhalten die hoch- und niederfrequenten Signale 380 000 Abtastwerte pro Schweißzyklus. Im Folgenden sei $\mathcal{X}_A \subseteq \mathbb{R}^{380\,000}$ die Menge der möglichen Eingabewerte, und $\mathcal{Y}_{A,\text{berst}} := \{0, 1\}$ sei die Menge der zugehörigen Ausgabewerte. Dabei steht der Wert 0 für die Güteklasse *schlecht* und der Wert 1 für

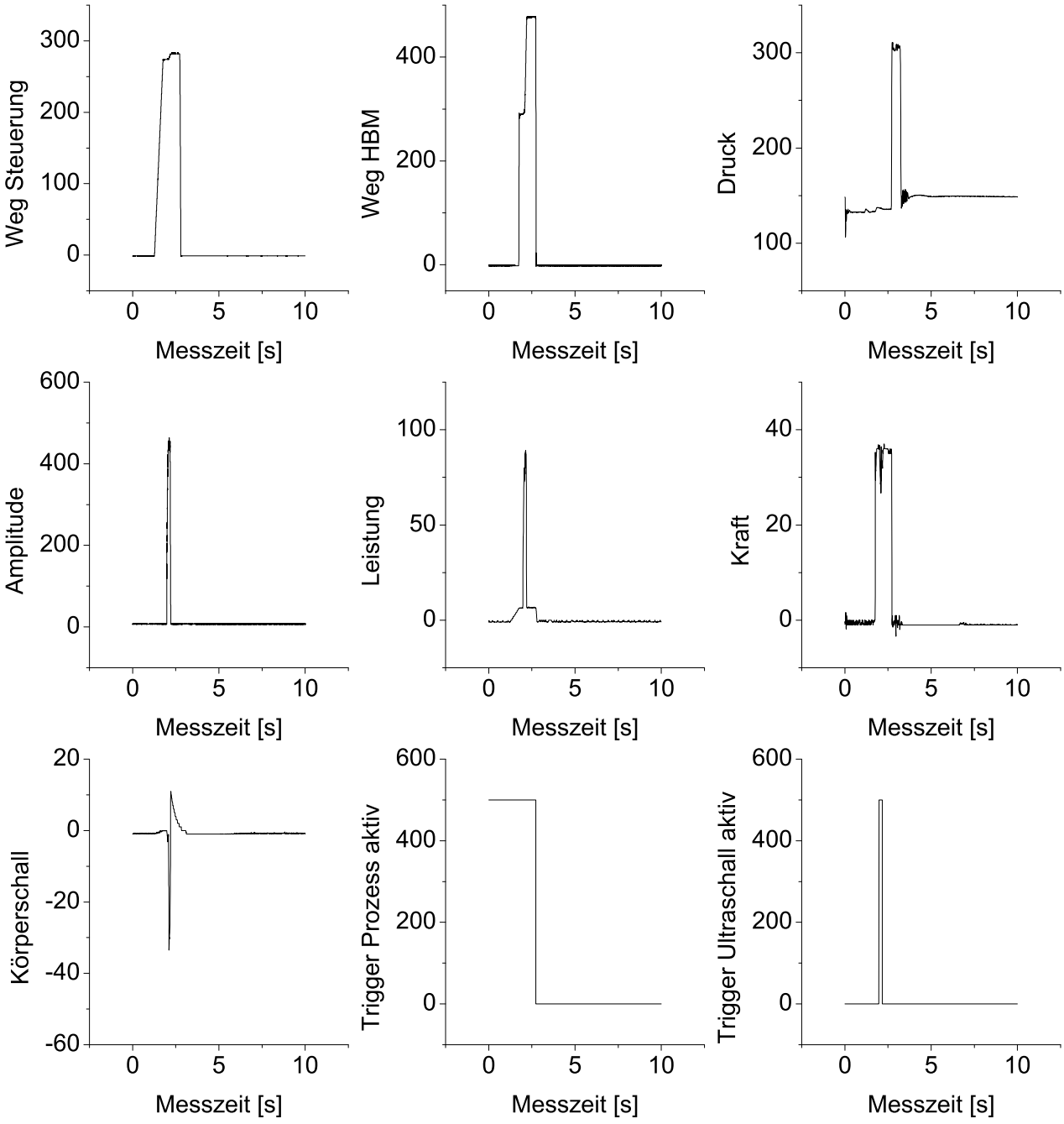


Abbildung 5.3: Verläufe der mit 2 kHz abgetasteten Signale beim Schweißen eines Probekörpers

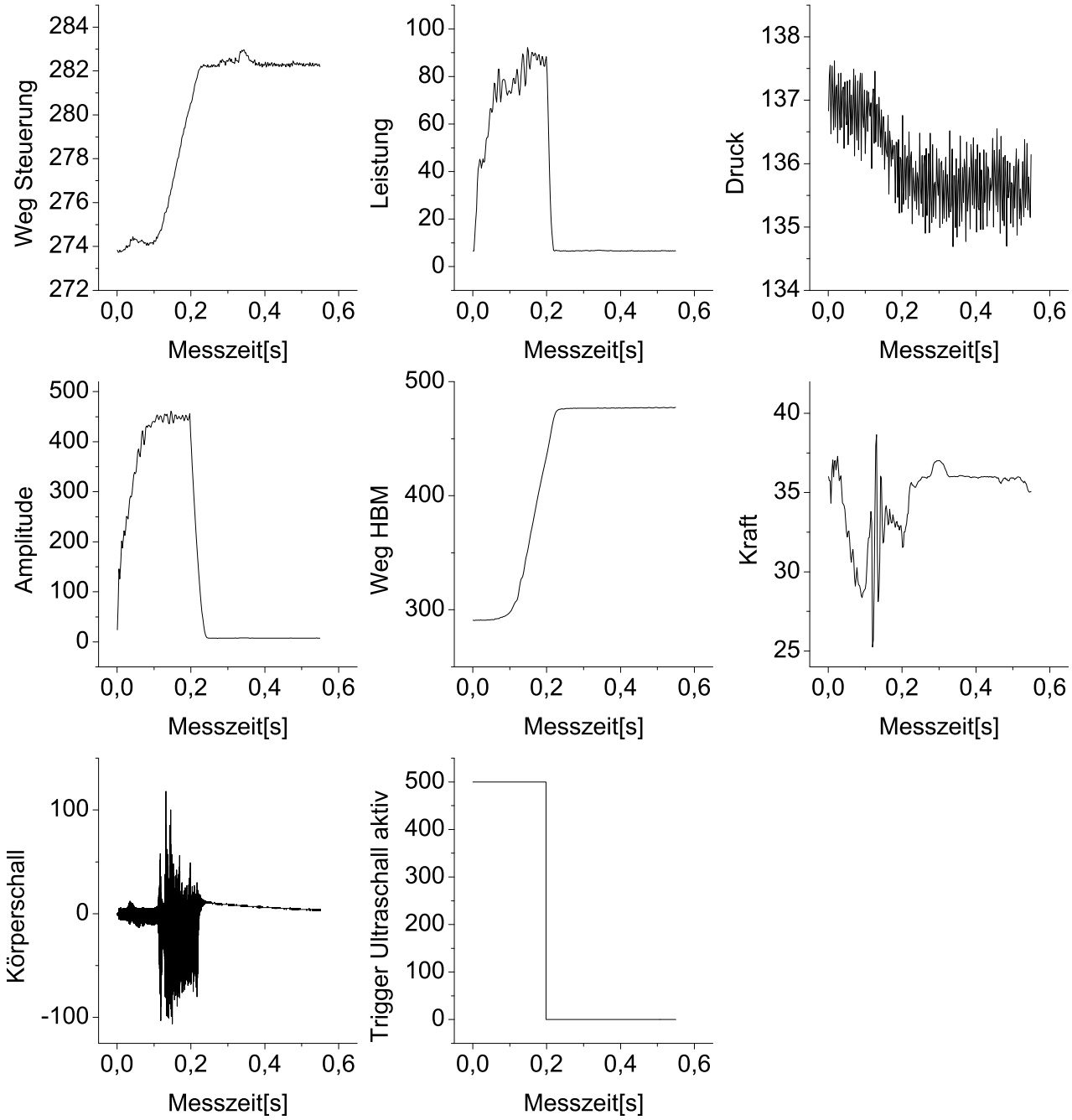


Abbildung 5.4: Verläufe der mit 45 kHz abgetasteten Signale beim Schweißen eines Probekörpers

die Güteklasse *gut*. Die Menge aller Datensätze aus Versuchsreihe A wird mit $\mathcal{D}_A \subseteq \mathcal{X}_A \times \mathcal{Y}_{A,\text{berst}}$ bezeichnet.

Daten aus Schweißungen mit Praxisbauteilen

Auch die Datensätze der *Versuchsreihe B* wurden im Rahmen des Forschungsprojektes SC-QUPUS erfasst. Hierbei wurde bei einem Industriepartner der Schweißprozess eines Wasserverteilers überwacht. Das Produkt besteht aus zwei Gehäuseteilen, in welche vor dem Schweißen eine weiche Filtermatte eingelegt wird. Für die Versuchsreihe wurden die Schweißparameter gezielt variiert. Die Versuchsreihe umfasst 179 Schweißzyklen. Für jeden Schweißzyklus wurden folgende Größen als Zeitsignale erfasst: der Schweißweg, die Schwingungsamplitude der Sonotrode, die Leistungsaufnahme des Ultraschallgenerators, die Schweißkraft, der Körperschall an der Werkstückaufnahme und diverse Trigger. Wie schon in Versuchsreihe A wurden die Signale während des gesamten Schweißzyklus mit 2 kHz abgetastet. Solange der Ultraschall aktiv war, wurden die Signale auch hier zusätzlich mit 45 kHz abgetastet [Neher u. a. 2010, Neher 2012]. Die Abbildungen 5.5 und 5.6 zeigen beispielhaft die Signalverläufe während eines einzelnen Schweißzyklus.

Im Vergleich zu den Schweißversuchen am Probekörper ist die Schweißzeit beim Wasserverteiler wesentlich länger. Entsprechend sind die erfassten Signale auch größer. So beinhalten die hoch- und niederfrequenten Signale in der Summe 416 000 Abtastwerte pro Schweißzyklus. Für die Menge $\mathcal{X}_B$ der möglichen Eingabedaten gilt somit $\mathcal{X}_B \subseteq \mathbb{R}^{416\,000}$.

Der im untersuchten Prozess verschweißte Wasserverteiler ist für den sichtbaren Einsatz in Wohnräumen konzipiert. Aus diesem Grund ist neben der Festigkeit der Schweißnaht auch der optische Eindruck des verschweißten Teils relevant. Die Nahtfestigkeit wurde wieder über eine Berstdruckprüfung bestimmt. Der Industriepartner hat hierzu eine entsprechende Prüfeinrichtung. Im Rahmen einer

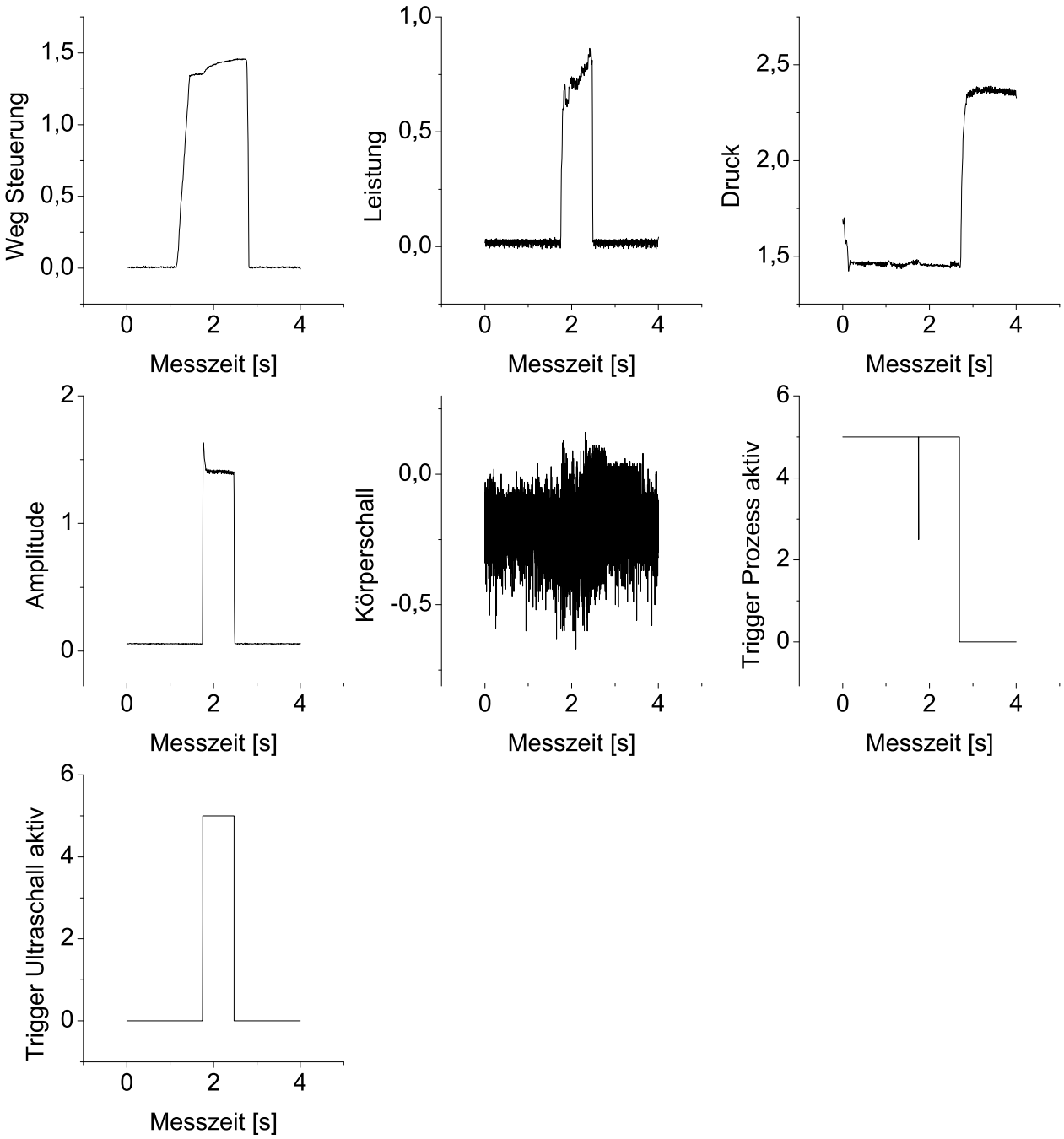


Abbildung 5.5: Verläufe der mit 2 kHz abgetasteten Signale beim Schweißen eines Wasserverteilers

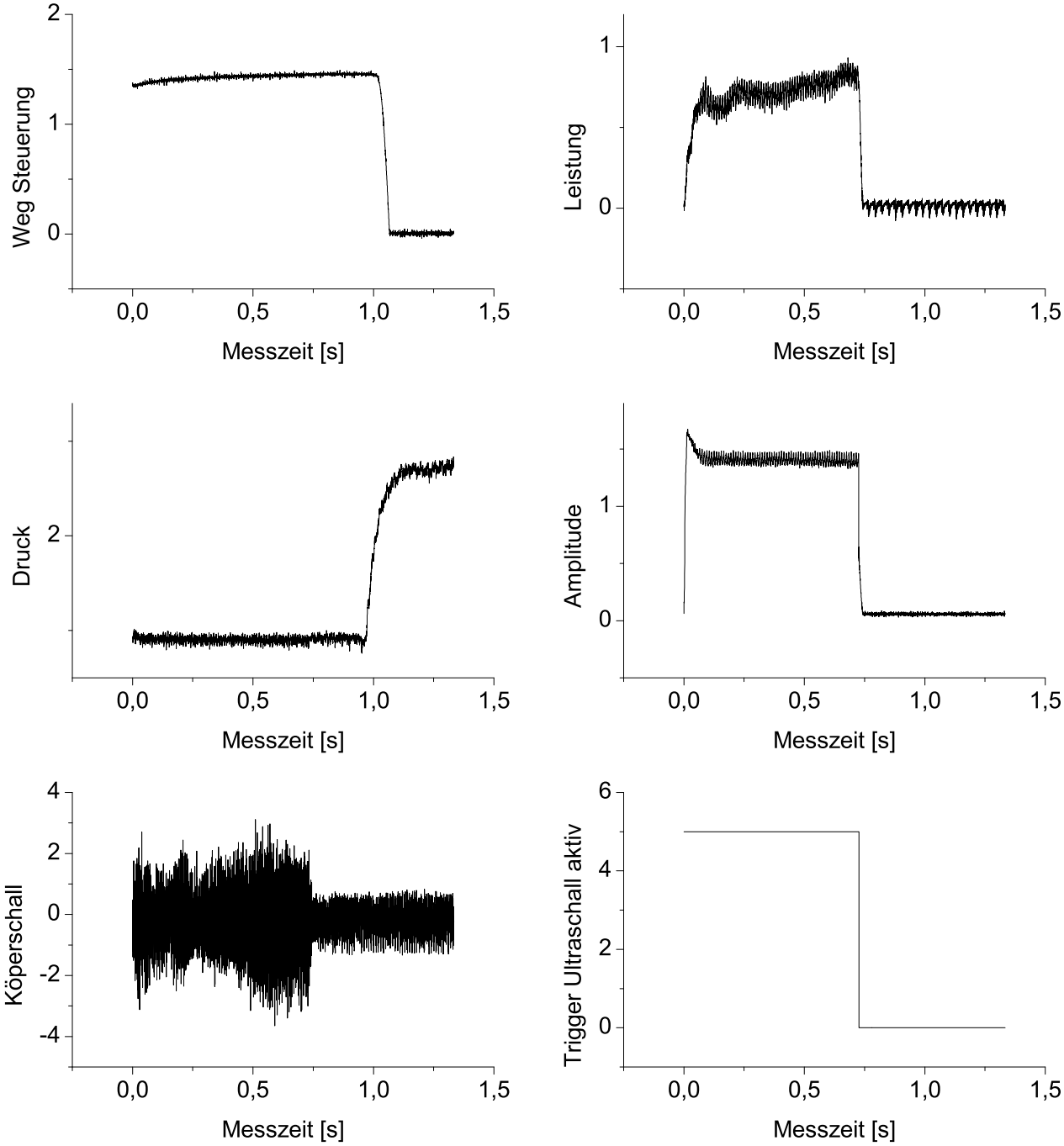


Abbildung 5.6: Verläufe der mit 45 kHz abgetasteten Signale beim Schweißen eines Wasserverteilers

visuellen Begutachtung wurde zudem das Spaltmaß zwischen den beiden Gehäuseteilen bewertet. Hierfür standen drei Kategorien zur Verfügung. Die Menge der möglichen Kategorien ist dabei $\mathcal{Y}_{B,\text{spalt}} := \{0, 1, 2\}$. Die gemessenen Berstdrücke liegen im Bereich zwischen 11,8 bar und 34,6 bar. Auch hier wurden die gefertigten Teile wieder in zwei Güteklassen eingeteilt: *gut* (Berstdruck $>$ 22 bar) und *schlecht* (Berstdruck $\leq$ 22 bar). Diese Güteklassen wurden mit numerischen Werten aus der Menge $\mathcal{Y}_{B,\text{berst}} := \{0, 1\}$ codiert. Dabei steht der Wert 0 wieder für die Güteklasse *schlecht* und der Wert 1 für die Güteklasse *gut*.

Kombiniert man die beiden Qualitätsmerkmale mit den Eingangsdaten, so erhält man zwei Mengen von Lerndatensätzen:

- die Datensatzmenge $\mathcal{D}_{B,1} \subseteq \mathcal{X}_B \times \mathcal{Y}_{B,\text{berst}}$ zur Prognose der Güteklasse,
- die Datensatzmenge $\mathcal{D}_{B,2} \subseteq \mathcal{X}_B \times \mathcal{Y}_{B,\text{spalt}}$ zur Prognose des Spaltmaßes.

5.1.4 Versuchsablauf

In den hier beschriebenen Versuchen sollen automatisch generierte Überwachungssysteme mit den beiden von Experten erstellten Systemen verglichen werden. Wie schon bei den Lernmodellen ist auch bei intelligenten Prozessüberwachungssystemen die Verallgemeinerungsfähigkeit eines Systems das wesentliche Maß für dessen Güte (siehe Abschnitt 2.4.3). Als Maßzahl für die Verallgemeinerungsfähigkeit wird nachfolgend die durchschnittliche Trefferquote $\bar{T}$ des entsprechenden Systems in einer 10-fachen stratifizierten Kreuzvalidierung bestimmt.

Für den Vergleich der verschiedenen Überwachungssysteme wurden für jedes System dessen durchschnittliche Trefferquoten bei der Prognose folgender Qualitätsmerkmale ermittelt:

- die Berstdruckklasse aus Datensatzmenge $\mathcal{D}_A \subseteq \mathcal{X}_A \times \mathcal{Y}_{A,\text{berst}}$,

- die Berstdruckklasse aus Datensatzmenge $\mathcal{D}_{B,1} \subseteq \mathcal{X}_B \times \mathcal{Y}_{B,\text{berst}}$,
- die Kategorie des Spaltmaßes aus Datensatzmenge $\mathcal{D}_{B,2} \subseteq \mathcal{X}_B \times \mathcal{Y}_{B,\text{spalt}}$.

Das NEHER-System verwendet mit GRNFN ein Regressionsverfahren für die Prognose. Das Ergebnis einer Prognose ist daher zunächst ein kontinuierlicher Wert $y \in \mathbb{R}$. Dieser Wert muss einer der diskreten Klassen aus der jeweiligen Zielgrößenmenge $\mathcal{Y}$ zugeordnet werden. Hierzu wird y auf den nächstliegenden ganzzahligen Wert aus $\mathcal{Y}$ gerundet.

Für das SC-QUPUS-System und das NEHER-System werden die durchschnittlichen Trefferquoten für jede der oben genannten Prognoseaufgaben wie folgt ermittelt: Die jeweils verwendete Datensatzmenge $\mathcal{D}_i \subseteq \mathcal{X}_i \times \mathcal{Y}_i$ wird in jeder Iteration der Kreuzvalidierung in eine Menge von Trainingsdaten $\mathcal{D}_{\text{train}} \subseteq \mathcal{D}_i$ und eine Menge von Testdaten $\mathcal{D}_{\text{test}} \subseteq \mathcal{D}_i$ aufgeteilt. Beide Systeme werden unter Verwendung der Lerndatensätze aus $\mathcal{D}_{\text{train}}$ trainiert. Hierzu werden den Systemen jeweils alle Trainingsdatensätze $(\boldsymbol{x}, y) \in \mathcal{D}_{\text{train}}$ übergeben. Jedes der Systeme führt auf den rohen Sensorsignalen $\boldsymbol{x}$ zunächst die von den Experten definierten Vorverarbeitungsschritte aus. Aus den vorverarbeiteten Signalen werden nach der fest vorgegebenen Berechnungsvorschrift des jeweiligen Systems Kenngrößen berechnet und anschließend selektiert. Mit den so ermittelten Kenngrößen und den zugehörigen Qualitätsmerkmalen y wird das jeweilige System dann trainiert. Anschließend werden den trainierten Systemen die Verifikationsdatensätze $(\boldsymbol{x}, y) \in \mathcal{D}_{\text{verif}}$ einzeln übergeben. Analog zu den Trainingsdaten werden auch hier die Eingangsdaten $\boldsymbol{x}$ zunächst vorverarbeitet. Aus den vorverarbeiteten Signalen werden die Kenngrößen berechnet und selektiert. Die erhaltenen Kenngrößen werden dann klassifiziert. Die Ergebnisse der Klassifikation werden mit dem jeweiligen Soll-Ergebnis y verglichen. Auf Basis dieser Vergleiche wird die durchschnittliche Trefferquote $\bar{T}$ über alle Iterationen der Kreuzvalidierung hinweg berechnet.

Parallel zu den oben beschriebenen Auswertungen wird auch die Leistungsfähigkeit des hier vorgestellten Verfahrens zur automatisierten Erzeugung von Prozessüberwachungssystemen evaluiert. Hierzu werden für jede der drei Prognoseaufgaben automatisch Prozessüberwachungssysteme erzeugt und deren durchschnittliche Trefferquoten bestimmt. Die Erzeugung und Bewertung der Prozessüberwachungssysteme erfolgt ebenfalls in einer 10-fachen stratifizierten Kreuzvalidierung. Der Ablauf ist dabei folgender: Zunächst wird auch hier die jeweils verwendete Datensatzmenge $\mathcal{D}_i \subseteq \mathcal{X}_i \times \mathcal{Y}_i$ in jeder Iteration der Kreuzvalidierung in eine Menge von Trainingsdaten $\mathcal{D}_{\text{train}} \subseteq \mathcal{D}_i$ und eine Menge von Testdaten $\mathcal{D}_{\text{test}} \subseteq \mathcal{D}_i$ aufgeteilt. Die Trainings- und Testdatenmengen sind dabei jeweils identisch mit denen, die für die Bewertung des SC-QUPUS- und des NEHER-Systems verwendet werden. Die Trainingsdaten $\mathcal{D}_{\text{train}}$ werden als Eingabedaten für die Modellauswahl verwendet. Mit diesen Trainingsdaten wird ein vollständiges Prozessüberwachungssystem generiert und trainiert. Dem neu erzeugten System werden anschließend die Verifikationsdaten $(\boldsymbol{x}, y) \in \mathcal{D}_{\text{verif}}$ einzeln übergeben. Das System erstellt dann für jedes Eingabe-Tupel $\boldsymbol{x}$ eine Prognose. Über alle Iterationen der Kreuzvalidierung hinweg werden wieder alle Prognosen mit dem jeweiligen Soll-Ergebnis y verglichen. Auf Basis dieses Vergleichs wird wiederum die durchschnittliche Trefferquote $\bar{T}$ bestimmt.

Das hier beschriebene System für die automatische Erzeugung wird für alle Versuchsreihen identisch parametriert. Hierbei werden für dessen Parameter k_{v}, γ_t und $n_{\max}$ jeweils die Standardwerte des Systems übernommen.

5.1.5 Vorhersageergebnisse

Das SC-QUPUS-System bildet für die Prognose der Berstdruckklasse für Datensatzmenge $\mathcal{D}_A$ insgesamt 227 Kenngrößen. Für die Prognosen anhand der Datensatzmengen $\mathcal{D}_{B,1}$ und $\mathcal{D}_{B,2}$ verwendet das SC-QUPUS-System jeweils 97 Kenngrö-

ßen. Eine Analyse der damit erzielten Prognoseergebnisse offenbarte, dass der im SC-QUPUS-System verwendete LVQ-Klassifikator das Potential dieser Kenngrößen nicht voll ausschöpft. Daher wurden im Rahmen der vorliegenden Arbeit auch Untersuchungen mit alternativen Klassifikationsverfahren durchgeführt. Diese Untersuchungen zeigen, dass andere Verfahren mit denselben Kenngrößen zum Teil wesentlich bessere Prognoseergebnisse erzielen. Die besten Prognoseergebnisse liefert dabei jeweils LogitBoost mit Entscheidungsbäumen als Basislerner. Bei den nachfolgenden Betrachtungen soll auch die eigentliche Güte der SC-QUPUS-Kenngrößen betrachtet werden. Aus diesem Grund wird für alle Prognoseaufgaben zusätzlich die durchschnittliche Trefferquote dargestellt, die LogitBoost mit Entscheidungsbäumen unter Verwendung der jeweiligen Kenngrößen aus dem SC-QUPUS-System erzielt.

Prognose der Berstdruckklasse für die Probekörper

Abbildung 5.7 stellt die durchschnittlichen Trefferquoten $\bar{T}$ der einzelnen Überwachungssysteme bei der Prognose der Berstdruckklasse für Datensatzmenge $\mathcal{D}_A$ dar. Das Prognoseergebnis, das mit den automatisch generierten Prozessüberwachungssystemen erzielt wurde, ist dabei farblich hervorgehoben.

Das SC-QUPUS-System erreicht bei dieser Prognoseaufgabe eine durchschnittliche Trefferquote von 93,62 %. Dies entspricht drei Fehlprognosen bei 47 Datensätzen. Werden dieselben Kenngrößen wie oben beschrieben mit LogitBoost mit Entscheidungsbäumen klassifiziert, so kann die Anzahl der Fehlprognosen auf zwei reduziert werden. Damit ergibt sich eine durchschnittliche Trefferquote von 95,74 %. Diese Trefferquote ist identisch mit der, die auch das NEHER-System erzielt. Die Überwachungssysteme, die mit dem hier vorgestellten Verfahren automatisch generiert wurden, erreichen im Durchschnitt eine Trefferquote von

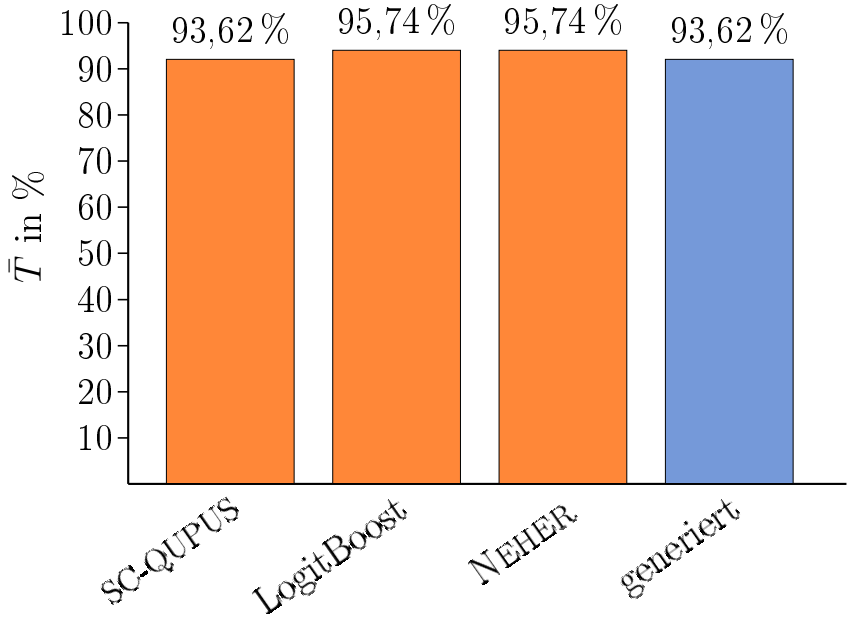


Abbildung 5.7: Durchschnittliche Trefferquoten $\bar{T}$ bei der Prognose der Berstdruckklasse für Datensatzmenge $\mathcal{D}_A$

93,62 %. Die erreichte Prognosegüte liegt also gleichauf mit der des SC-QUPUS-Systems, bleibt aber knapp hinter der Trefferquote des NEHER-Systems.

Für die automatische Erzeugung eines Überwachungssystems auf Basis der Datensatzmenge $\mathcal{D}_A$ benötigt die Modellauswahl des hier vorgestellten Verfahrens im Durchschnitt 18 Optimierungszyklen. Dabei dauert die Erzeugung solch eines Überwachungssystems auf einem aktuell handelsüblichen PC (Intel i7-3770, 16 GB Hauptspeicher) durchschnittlich rund 22 Stunden.

Prognose der Berstdruckklasse für die Praxisbauteile

Die durchschnittlichen Trefferquoten $\bar{T}$, die die einzelnen Überwachungssysteme bei der Prognose der Berstdruckklasse für Datensatzmenge $\mathcal{D}_{B,1}$ erreichen, sind

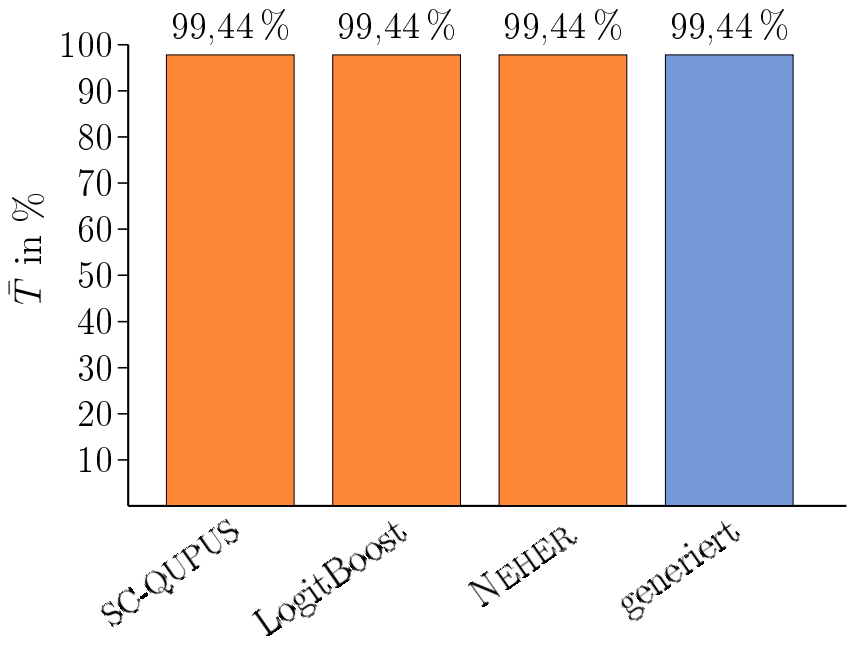


Abbildung 5.8: Durchschnittliche Trefferquoten $\bar{T}$ bei der Prognose der Berstdruckklasse für Datensatzmenge $\mathcal{D}_{B,1}$

in Abbildung 5.8 dargestellt. Bei dieser Aufgabe schnitten alle Verfahren gleich gut ab. Die sehr hohe Trefferquote von 99,44 % entspricht bei 179 Datensätzen einer einzigen Falschklassifikation. Für die automatische Erzeugung der Überwachungssysteme für diese Prognoseaufgabe benötigt das hier vorgestellte Verfahren im Mittel 15 Optimierungszyklen. Auf dem verwendeten PC (Intel i7-3770 mit 16 GB Hauptspeicher) dauert die Erzeugung solch eines Überwachungssystems durchschnittlich etwa 17 Stunden.

Prognose des Spaltmaßes für die Praxisbauteile

Bei der Prognose des Spaltmaßes für Datensatzmenge $\mathcal{D}_{B,2}$ werden deutlich niedrigere durchschnittliche Trefferquoten erreicht als bei den vorangegangenen Pro-

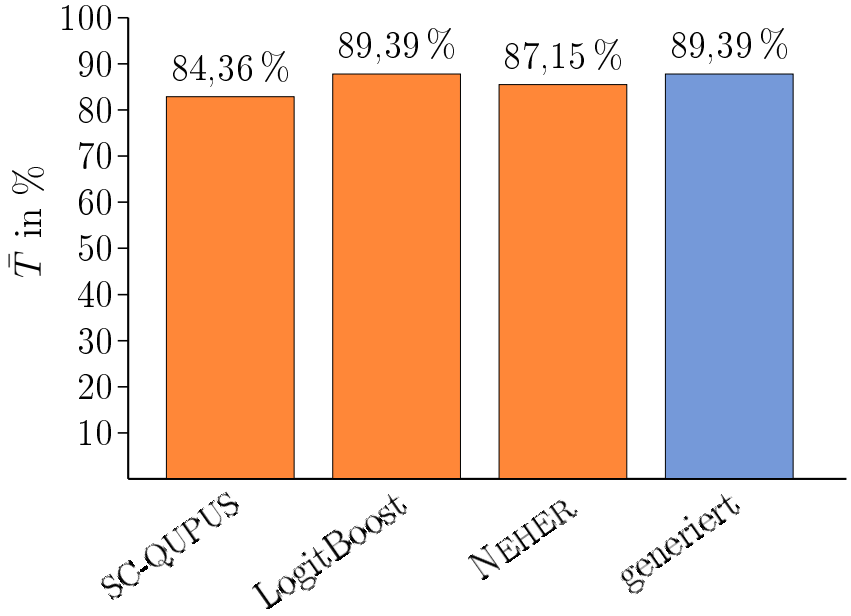


Abbildung 5.9: Durchschnittliche Trefferquoten $\bar{T}$ bei der Prognose des Spaltmaßes für Datensatzmenge $\mathcal{D}_{B,2}$

gnoseaufgaben. Dies war so zu erwarten, da das Spaltmaß nicht auf einer objektiv gemessenen Größe basiert. Vielmehr handelt es sich hierbei um eine subjektive Beurteilung durch einen menschlichen Gutachter. Solche subjektiven Größen sind oft mit einer Unschärfe behaftet. Dadurch sind sie meist schwieriger zu prognostizieren.

Abbildung 5.9 stellt die durchschnittlichen Trefferquoten der betrachteten Überwachungssysteme dar. Das SC-QUPUS-System erzielt mit $\bar{T} = 84,36\,\%$ die niedrigste Prognosegüte. Es zeigt sich aber, dass mit den vom SC-QUPUS-System verwendeten Kenngrößen eine wesentlich bessere Prognose möglich ist. So erzielt LogitBoost mit Entscheidungsbäumen mit denselben Kenngrößen eine durchschnittliche Trefferquote von 89,39 %. Damit übertrifft das Verfahren sogar das

NEHER-System, welches eine Trefferquote von 87,15 % erreicht. Die automatisch generierten Überwachungssysteme belegen mit einer durchschnittlichen Trefferquote von ebenfalls 89,39 % den Spitzenplatz.

Für die automatische Erzeugung eines Überwachungssystems auf Basis der vorgegebenen Daten benötigt das hier vorgestellte Verfahren im Durchschnitt 15 Iterationsschritte. Die Erzeugung solch eines Überwachungssystems auf dem Referenz-PC (Intel i7-3770, 16 GB Hauptspeicher) dauert im Mittel 29 Stunden.

5.1.6 Analyse der generierten Überwachungssysteme

Nachfolgend werden die automatisch erzeugten Prozessüberwachungssysteme für die einzelnen Aufgabestellungen genauer betrachtet. Es ist zu berücksichtigen, dass durch die 10-fache Kreuzvalidierung für jede Problemstellung zehn verschiedene Prozessüberwachungssysteme erzeugt werden. Da die Modellauswahl zufallsbasiert ist, unterscheiden sich die für dieselbe Problemstellung erzeugten Prozessüberwachungssysteme teilweise deutlich. Bei der nachfolgenden Analyse werden die relevanten Werte daher jeweils über alle generierten Überwachungssysteme der jeweiligen Problemstellung gemittelt.

Die zur Prognose der Berstdruckklasse für Datensatzmenge $\mathcal{D}_A$ erstellten Überwachungssysteme generieren durchschnittlich 2 985 Kenngrößen. Von diesen Kenngrößen werden im Mittel 2 734 selektiert. Die zehn Überwachungssysteme verwenden für die Kenngrößenselektion vorwiegend einen Ranker mit 1R-Relevanzfunktion (in vier Fällen). In acht der zehn Systeme kommt als Klassifikator LogitBoost mit Entscheidungsbäumen zum Einsatz. Die verbleibenden Systeme nutzen eine C-SVM.

Für die Prognose der Berstdruckklasse für Datensatzmenge $\mathcal{D}_{B,1}$ werden durchschnittlich 1 917 Kenngrößen generiert. Davon werden im Mittel 1 688 selektiert. Für die Kenngrößenselektion kommt vorrangig die Identitätsfunktion zum Ein-

satz. Sie wird in sieben der zehn Systeme verwendet. Die anderen Systeme nutzen einen Ranker. Neun der zehn Systeme verwenden für die Klassifikation LogitBoost mit Entscheidungsbäumen. Das verbliebene System nutzt den naiven Bayes-Klassifikator.

Um das Spaltmaß für Datensatzmenge $\mathcal{D}_{B,2}$ vorherzusagen, werden im Durchschnitt 2 538 Kenngrößen erzeugt. Hiervon werden im Mittel 2 230 selektiert. Als Verfahren für die Kenngrößenselektion wird vorrangig die Identitätsfunktion eingesetzt (in sechs der zehn Systeme). Alle zehn Systeme klassifizieren mit LogitBoost.

Über alle drei Prognoseaufgaben hinweg setzen die erzeugten Systeme für die Klassifikation also vorwiegend LogitBoost ein. Dies deckt sich auch mit den Erkenntnissen aus der Literatur. Dort gelten die Boosting-Verfahren als die besten Klassifikatoren für Eingangsdaten mit wenigen Tausend Kenngrößen [Guyon 2009].

Nachfolgend wird für jede Prognoseaufgabe untersucht, wie stark sich die einzelnen Rohsignale in den verwendeten Kenngrößen niederschlagen. Hierzu werden die Kenngrößen betrachtet, die letztendlich von der Kenngrößenselektion ausgewählt wurden. Aus der Quellcodedatei des zugrunde liegenden Kenngrößenerzeugers wird für jedes Signal die Gesamtanzahl der Messwerte ermittelt, die dieses zur Bildung der selektierten Kenngrößen beiträgt. Die so gewonnenen Häufigkeiten werden wieder über alle zehn Überwachungssysteme der jeweiligen Prognoseaufgabe gemittelt.

Das Diagramm in Abbildung 5.10 zeigt den Anteil, den die nieder- und hochfrequenten Signale zu den Rohdaten in Datensatzmenge $\mathcal{D}_A$ beitragen. Dem wird gegenübergestellt, welchen Anteil die nieder- und hochfrequenten Signale an den Messwerten haben, die zur Bildung der selektierten Kenngrößen für die Prognose der Berstdruckklasse verwendet werden. Analog dazu stellt Abbildung 5.11 den

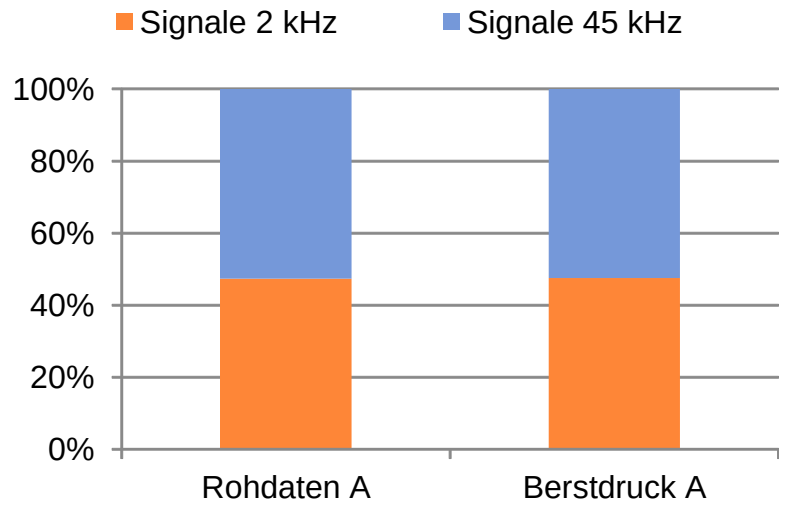


Abbildung 5.10: Anteil nieder- und hochfrequenter Abtastwerte in den Rohdaten und in den Kenngrößen für die Versuche mit dem Probekörper

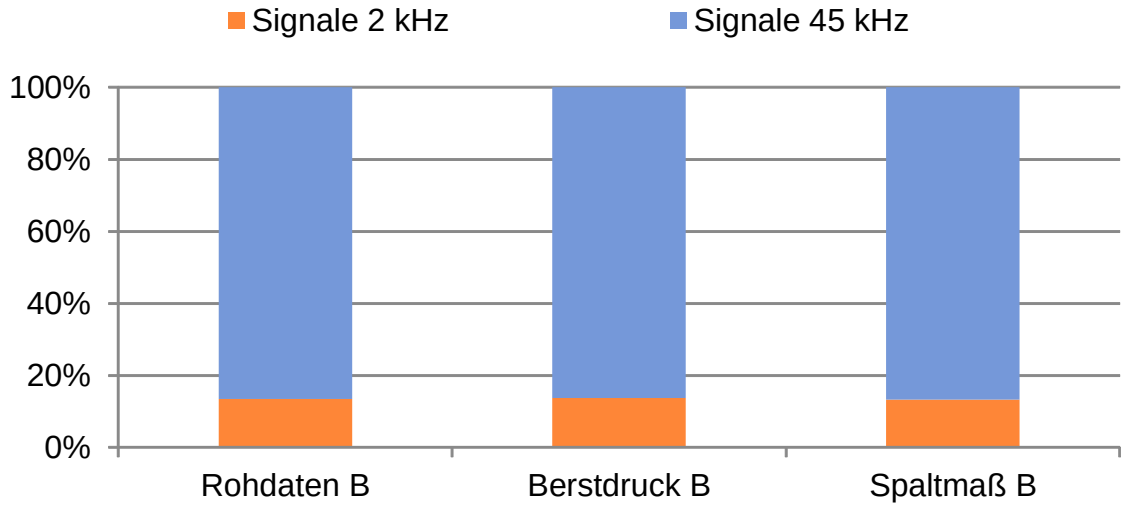


Abbildung 5.11: Anteil nieder- und hochfrequenter Abtastwerte in den Rohdaten und in den Kenngrößen für die Versuche mit dem Wasserverteiler

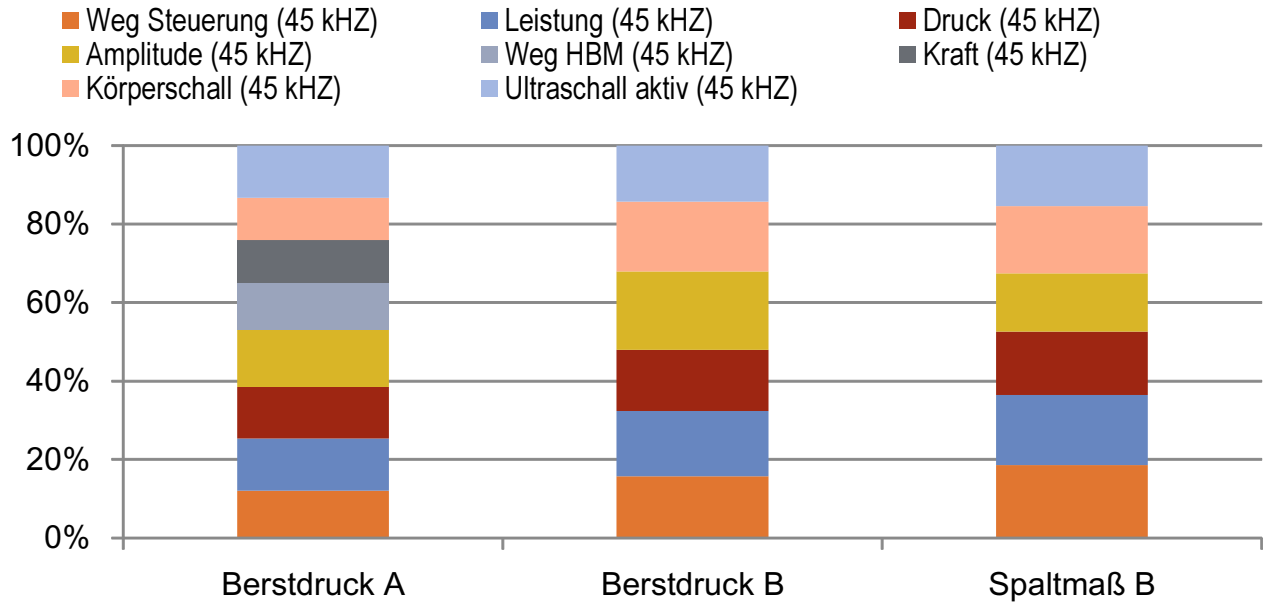


Abbildung 5.12: Durchschnittliche relative Häufigkeiten innerhalb der niederfrequenten Signale in den Kenngrößen der erzeugten Überwachungssysteme

Anteil der nieder- und hochfrequenten Signale an den Rohdaten in Datensatzmenge $\mathcal{D}_B$ dar. Zudem wird gezeigt, wie sich die nieder- und hochfrequenten Signale anteilig in den verwendeten Kenngrößen für die Prognose des Berstdruckes bzw. des Spaltmaßes niederschlagen. Es ist zu erkennen, dass für alle drei Prognoseaufgaben die Aufteilung für die Kenngrößen nahezu identisch mit der Aufteilung in den Rohdaten ist.

Die Abbildungen 5.12 und 5.13 gliedern für alle drei Prognoseaufgaben die Anteile der einzelnen Signale auf. Die nieder- und hochfrequenten Signale werden dabei getrennt betrachtet. Sowohl im hochfrequenten als auch im niederfrequenten Bereich zeigt sich dabei, dass die verwendeten Abtastwerte für alle drei Prognoseaufgaben nahezu gleichverteilt über alle Signale sind.

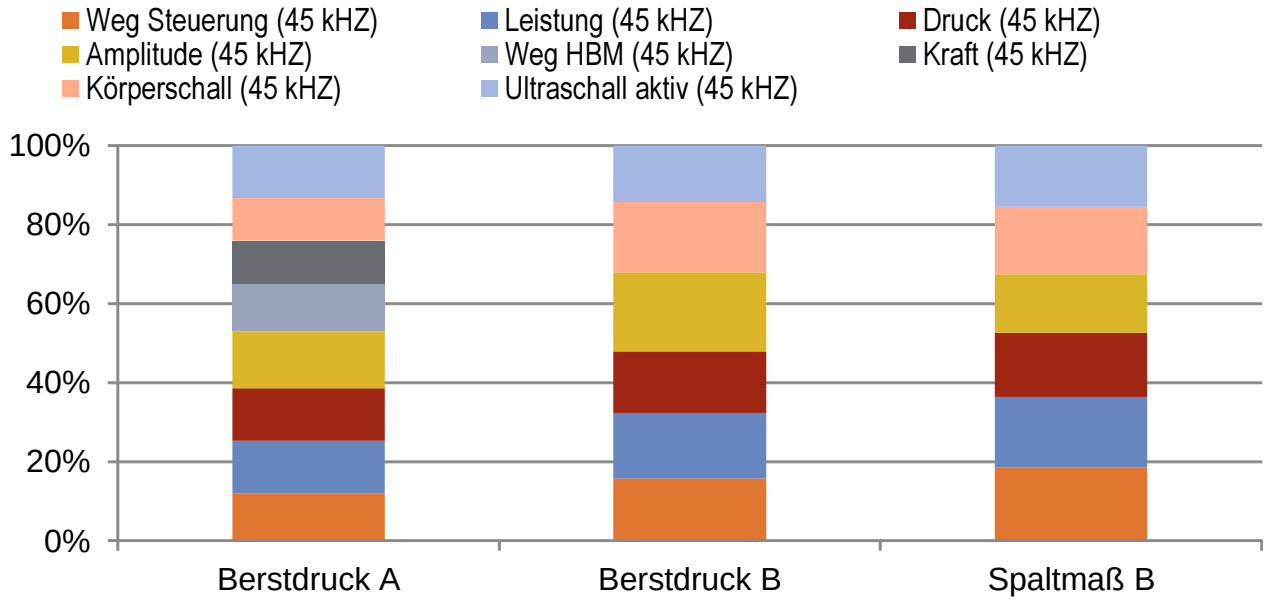


Abbildung 5.13: Durchschnittliche relative Häufigkeiten innerhalb der hochfrequenten Signale in den Kenngrößen der erzeugten Überwachungssysteme

Zusammenfassend lässt sich sagen, dass die Häufigkeit der Signale in den verwendeten Kenngrößen lediglich die Häufigkeit der Signale in den Rohdaten widerspiegelt. Somit können aus den Kenngrößen keine Rückschlüsse darauf gezogen werden, welche Signalquellen für die erzielten Prognoseergebnisse herausragend wichtig oder unwichtig sind.

5.2 Anwendung beim Spritzgießen

5.2.1 Beschreibung des Fertigungsverfahrens

Spritzgießen ist das wirtschaftlich bedeutendste Verfahren zur Herstellung von Formteilen aus Kunststoffen und Kautschuk. Beim Spritzgießen wird der Werk-

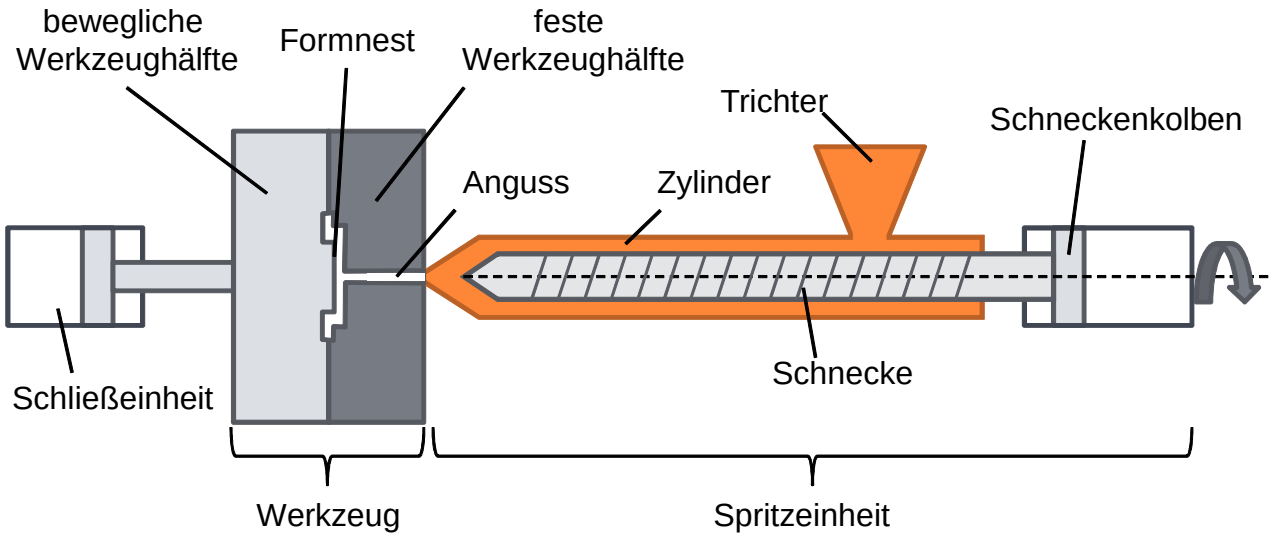


Abbildung 5.14: Komponenten einer Spritzgießmaschine [Johannaber u. a. 2004]

stoff aufgeschmolzen, in einen formgebenden Hohlraum eingespritzt und dort verdichtet. Nachdem der Werkstoff in der Form erhärtet ist, wird er als Formteil aus dem verwendeten Werkzeug ausgeworfen [Johannaber u. a. 2004].

Eine Spritzgießmaschine besteht im Wesentlichen aus den in Abbildung 5.14 dargestellten Komponenten. Die Spritzgießmaschine wird über den *Trichter* der *Spritzeinheit* befüllt. Das eingefüllte Werkstoffgranulat rieselt direkt in die Gänge der rotierenden *Schnecke*. Die Schnecke fördert das Werkstoffgranulat durch den von außen beheizten *Zylinder*. Dabei schmilzt das Granulat auf und erreicht die Schneckenspitze im vollständigen Schmelzezustand. Durch den Fördervorgang baut sich vor der Schnecke ein Druck auf, der diese nach hinten schiebt. Wenn die Schnecke einen bestimmten Weg zurückgeschoben wurde, wird die Förderung eingestellt. Die vor der Schneckenspitze gesammelte Schmelze reicht nun aus, um den Werkzeughohlraum zu füllen. Nachdem über eine Düse die Verbindung zum Werkzeughohlraum hergestellt wurde, wird der *Schneckenkolben* hydraulisch oder

mechanisch nach vorne bewegt. Der dadurch vor der Schneckenspitze entstehende Druck presst die Schmelze in das temperierte Werkzeug. Im Werkzeug erhärtet die Schmelze. Die *Schließeinheit* öffnet nun das Werkzeug und stößt das fertige Formteil aus dem Formnest [Johannaber u. a. 2004].

Ein Spritzgießvorgang kann in drei Phasen eingeteilt werden: In der *Einspritzphase* liegt die Schneckendüse am Werkzeug an. Die Schnecke ist im Vorwärtshub und presst die Schmelze in den Werkzeughohlraum. Die Einspritzphase dauert an, bis ein zeit- oder wegabhängiges Signal den Prozess in die *Nachdruckphase* schaltet. In dieser Phase drückt die Schnecke auf ein Restvolumenpolster vor ihrer Spitze. So presst sie weiter Schmelze in das Werkzeug, um den Volumenschwund der abkühlenden Schmelze im Formnest zu kompensieren. Den Nachdruck hält man solange aufrecht, bis die Schmelze im Anguss erhärtet ist. Ab diesem Zeitpunkt kann keine Schmelze mehr fließen. Im Anschluss an die Nachdruckphase kann die Schnecke mit dem Dosiervorgang für den nächsten Zyklus beginnen. Mit dem Ende der Nachdruckphase beginnt außerdem eine einstellbar lange Kühl- bzw. Aushärtezeit. Wenn diese Zeit verstrichen ist, beginnt die *Auswerfphase*. In dieser Phase hat die Schnecke bereits die Schmelze für den nächsten Einspritzvorgang dosiert. Die Schließeinheit öffnet nun das Werkzeug, stößt die Formteile aus und schließt das Werkzeug wieder. Es beginnt die Einspritzphase des nächsten Zyklus [Johannaber u. a. 2004].

Die Prozessparameter wie Drücke, Wege und Zeiten sind vielfältig einstellbar. Die jeweiligen Einstellungen haben dabei großen Einfluss auf den Prozessverlauf und auf die Güte des produzierten Formteils [Johannaber u. a. 2004, Ribeiro 2005, Schmidberger u. a. 2005b].

5.2.2 Vergleichssysteme

Auch für den Anwendungsbereich Spritzgießen sollen die automatisch erzeugten Prozessüberwachungssysteme mit zwei von Experten erstellten Überwachungssystemen verglichen werden.

Eines der beiden Vergleichssysteme ist das *Neuronale Prozessregelungssystem (NEPRES)*, das von Schmidberger und anderen bereits ab den 90er-Jahren entwickelt wurde [Schmidberger u. a. 1995]. Die Entwicklungsarbeit an NEPRES wurde über viele Jahre hinweg in mehreren Folgeprojekten fortgeführt. Mit Abschluss des Verbundforschungsprojektes NEPRES II erreichte das ursprüngliche NEPRES-System seinen heutigen Stand. Kennzeichnend für NEPRES ist die Qualitätsüberwachung auf Basis von *Multilayer Perceptrons*. Die Multilayer Perceptrons sind ein weit verbreiteter Typ neuronaler Netze. NEPRES unterstützt insbesondere die Überwachung von Spritzgießprozessen. So bietet das System eine Vielzahl von Funktionen, die die Kenngrößenbildung speziell für das Spritzgießen unterstützen [Schmidberger u. a. 2005b].

Das zweite Vergleichssystem wurde ausgehend vom oben beschriebenen Stand des NEPRES-Systems im Verbundforschungsprojekt QSLR und dessen Nachfolgeprojekt *Adaptive Klassifikation* entwickelt. Dieses System – im folgenden QSLR-System genannt – wurde speziell auf die Anforderungen von Produktionsprozessen mit kleinen und mittleren Losgrößen zugeschnitten. Es liefert aber auch für Prozesse mit großen Losgrößen sehr gute Ergebnisse. Kern des QSLR-Systems ist ein modifiziertes LVQ-Klassifikationsverfahren [Kaupp u. a. 2010a]. Untersuchungen zeigen, dass sich das QSLR-System sehr gut für die Überwachung von Spritzgießprozessen eignet [Kaupp u. a. 2011, Kaupp u. a. 2012a]. Der Gesamtaufwand für die Erstellung des QSLR-Systems betrug mehr als sechs Personenjahre, verteilt auf eine Gesamtlaufzeit von 57 Monaten [Schmidberger u. a. 2008, Kaupp u. a. 2012b].

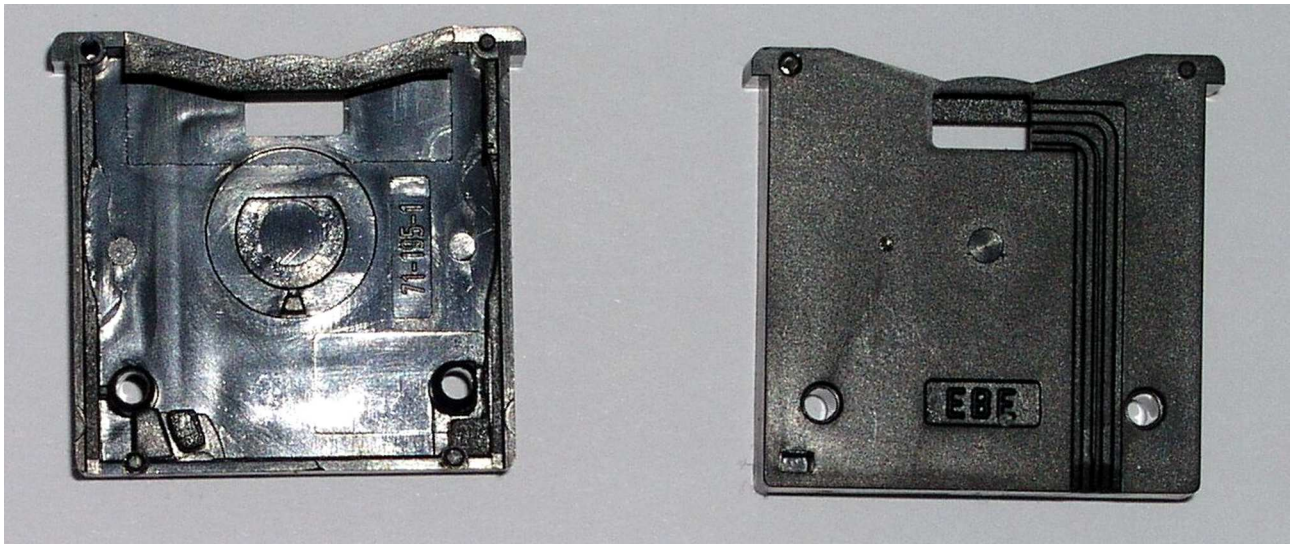


Abbildung 5.15: Präzisionsschaltergehäuse aus Versuchsreihe C (Quelle: Fraunhofer IPA)

5.2.3 Verwendete Daten

Für den Vergleich der Prozessüberwachungssysteme wird ein Datensatz verwendet, der bei Versuchen im Rahmen des Projektes NEPRES II aufgezeichnet wurde. Anhand dieses Datensatzes wird in [Kaupp u. a. 2010b] auch die Leistungsfähigkeit des QSLR-Systems dargestellt.

Die Datensätze der hier beschriebenen *Versuchsreihe C* wurden bei Spritzgießversuchen am Fraunhofer-Institut für Produktionstechnik und Automatisierung (IPA) erfasst. Dabei wurden auf der institutseigenen Spritzgießmaschine die in Abbildung 5.15 dargestellten Präzisionsschaltergehäuse gefertigt. Die Versuchsreihe umfasste 470 Produktionszyklen. Die aufgezeichneten Datensätze beinhalten für jeden Zyklus die Zeitsignale für folgende Größen: den Hydraulikdruck der Spritzgießmaschine, den Druck und die Drehzahl des Hydraulikmotors der Spritzeinheit, den Schneckenweg, den Druck und die Temperatur im Formnest sowie den Zu-

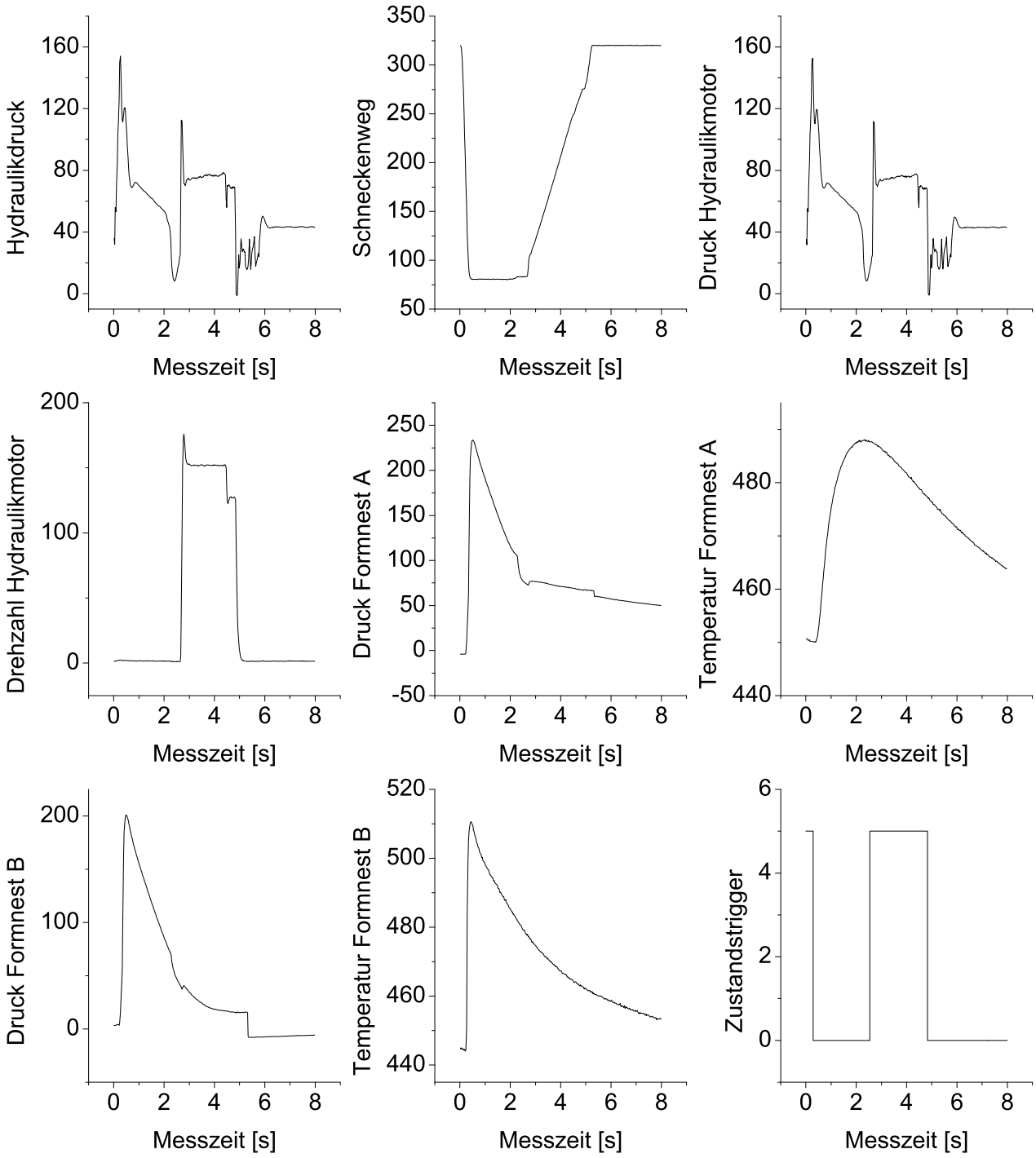


Abbildung 5.16: Signalverläufe eines Fertigungszyklus aus Versuchsreihe C

standstrigger der Spritzeinheit. Zur Erfassung des Drucks und der Temperatur im Formnest wurden jeweils zwei verschiedene Typen von Sensoren eingesetzt. Daher sind für diese Größen je zwei Signalkurven vorhanden [Schmidberger u. a. 2005b]. Alle erfassten Signale wurden mit 2 kHz abgetastet. Abbildung 5.16 zeigt beispielhaft die Signalverläufe eines Fertigungszyklus aus Versuchsreihe C.

Die in Versuchsreihe C gefertigten Präzisionsschaltergehäuse wurden nach Abschluss der Versuche einzeln gewogen. Jedem Produktionszyklus i wurde dabei das entsprechende Teilegewicht m_i als Qualitätsmerkmal zugeordnet. Die erfassten Gewichte liegen im Bereich zwischen 2,0954 g und 2,1330 g. Die obere Toleranzgrenze für das Teilegewicht liegt bei 2,1140 g. Die untere Toleranzgrenze liegt bei 2,1030 g. Entsprechend wurden die Gewichte in drei Klassen eingeteilt: *zu leicht* ($m_i < 2{,}1030\,\mathrm{g}$), *gut* ($2{,}1030\,\mathrm{g} \leq m_i \leq 2{,}1140\,\mathrm{g}$) und *zu schwer* ($m_i > 2{,}1140\,\mathrm{g}$).

Insgesamt umfassen die neun Signale 288 000 Abtastwerte. Im Folgenden sei $\mathcal{X}_C \subseteq \mathbb{R}^{288\,000}$ die Menge der möglichen Eingabewerte, und $\mathcal{Y}_C := \{0, 1, 2\}$ sei die Menge der zugehörigen Ausgabewerte. Dabei steht der Wert 0 für die Güteklasse *zu leicht*, der Wert 1 steht für die Güteklasse *gut* und der Wert 2 steht für die Güteklasse *zu schwer*. Die Menge aller Datensätze aus Versuchsreihe C wird mit $\mathcal{D}_C \subseteq \mathcal{X}_C \times \mathcal{Y}_C$ bezeichnet.

5.2.4 Versuchsablauf

Auch für das Spritzgießen sollen automatisch generierte Prozessüberwachungssysteme mit den von Experten erstellten Überwachungssystemen verglichen werden. Hierzu wird mit allen Systemen die Gewichtsklasse für die oben beschriebene Datensatzmenge $\mathcal{D}_C \subseteq \mathcal{X}_C \times \mathcal{Y}_C$ prognostiziert. Dabei wird wieder die durchschnittliche Trefferquote $\bar{T}$ der Prognosen in einer 10-fachen stratifizierten Kreuzvalidierung bestimmt. Die Versuche werden analog zu den Versuchen mit den Daten aus

den Ultraschallschweißprozessen durchgeführt. Der entsprechende Versuchsablauf wird in Abschnitt 5.1.4 beschrieben.

Das NEPRES-System verwendet mit den Multilayer Perceptrons ein Regressionsverfahren für die Prognose. Das Ergebnis einer Prognose ist daher zunächst ein kontinuierlicher Wert $y \in \mathbb{R}$. Dieser Wert muss einer der diskreten Klassen aus der Zielgrößenmenge $\mathcal{Y}_C$ zugeordnet werden. Hierzu wird y auf den nächstliegenden ganzzahligen Wert aus $\mathcal{Y}_C$ gerundet.

Das System zur automatischen Erzeugung der Prozessüberwachungssysteme ist so parametriert wie auch schon bei den Versuchen zum Ultraschallschweißen. Auch hier wurden für die Parameter k_v, γ_t und n_max jeweils die Standardwerte des Systems übernommen.

5.2.5 Vorhersageergebnisse

Abbildung 5.17 zeigt die Trefferquoten, die die einzelnen Prozessüberwachungssysteme bei der Prognose der Gewichtsklasse für Datensatzmenge $\mathcal{D}_C$ erzielen. Es ist zu erkennen, dass die automatisch generierten Prozessüberwachungssysteme auch bei der Überwachung von Spritzgießprozessen sehr gute Resultate liefern. Mit einer durchschnittlichen Trefferquote von 97,02 % übertreffen die generierten Überwachungssysteme die beiden von Experten erstellten Referenzsysteme deutlich.

Für die automatische Erzeugung eines Überwachungssystems auf Basis der vorgegebenen Daten benötigt das hier vorgestellte Verfahren im Durchschnitt 22 Optimierungszyklen. Die Erzeugung solch eines Überwachungssystems dauert auf dem Referenz-PC (Intel i7-3770, 16 GB Hauptspeicher) im Mittel 47 Stunden.

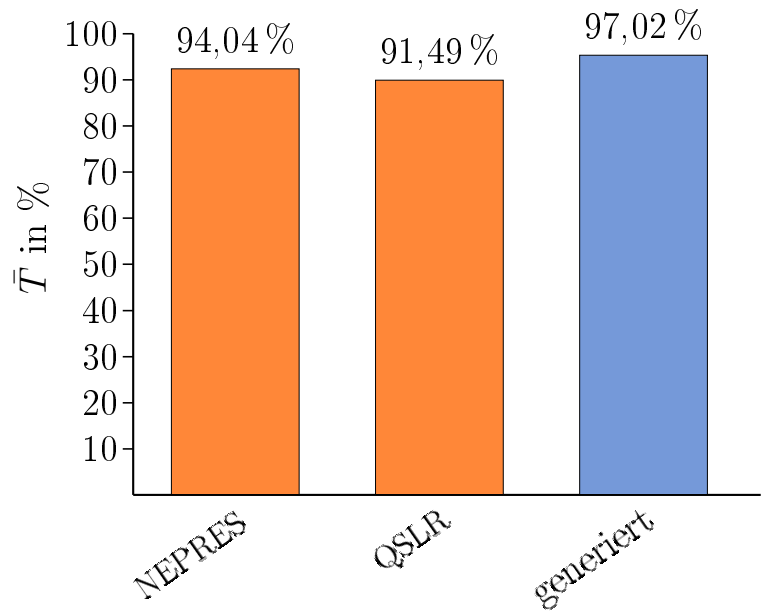


Abbildung 5.17: Durchschnittliche Trefferquoten $\bar{T}$ bei der Prognose der Gewichtsklasse für Datensatzmenge $\mathcal{D}_C$

5.2.6 Analyse der generierten Überwachungssysteme

Analog zur Bewertung der Überwachungssysteme für das Ultraschallschweißen in Abschnitt 5.1.6 sollen auch die automatisch erzeugten Prozessüberwachungssysteme für die Spritzgießanwendung analysiert werden. Auch hier wurden im Rahmen einer 10-fachen Kreuzvalidierung insgesamt zehn – potentiell stark unterschiedliche – Überwachungssysteme erzeugt.

Die erzeugten Systeme für die Prognose der Gewichtsklasse für Datensatzmenge $\mathcal{D}_C$ generieren durchschnittlich 2 053 Kenngrößen. Von diesen Kenngrößen werden im Mittel 1 974 selektiert. Bei der Kenngrößenselektion kommt in neun der zehn Systeme die Identitätsfunktion zum Einsatz. Als Klassifikationsverfahren wird in allen zehn Fällen LogitBoost ausgewählt. Eine Analyse der Zwischen-

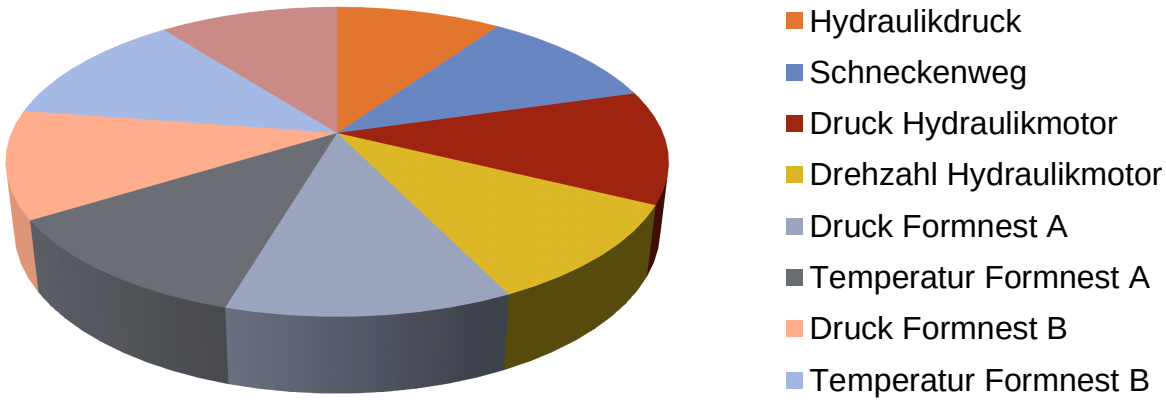


Abbildung 5.18: Durchschnittliche relative Häufigkeit der Signale in den Kenngrößen der erzeugten Überwachungssysteme

schritte zeigt, dass auch die anderen Klassifikationsverfahren bei der Auswahl berücksichtigt werden. Insbesondere die Support Vector Machines liefern hierbei ebenfalls sehr gute Prognoseergebnisse. Allerdings schneidet LogitBoost stets ein bis zwei Prozentpunkte besser ab.

Auch hier wird betrachtet, wie viele Messwerte die einzelnen Rohsignale zu den selektierten Kenngrößen beitragen. Dafür werden die Messwert- und Bereichsvariablen der verwendeten Kenngrößenerzeuger ausgewertet. So lässt sich für jedes Signal die Anzahl der Messwerte ermitteln, die es zur Gesamtmenge der verwendeten Messwerte beiträgt. Die Mittelwerte der so bestimmten Häufigkeiten für die einzelnen Signale sind in Abbildung 5.18 dargestellt. Es zeigt sich, dass die verwendeten Messwerte gleichmäßig über alle Signale gestreut sind. Folglich können aus den ermittelten relativen Häufigkeiten keine Rückschlüsse auf eine besondere Signifikanz einzelner Signale gezogen werden.

5.3 Fazit

In den hier beschriebenen Untersuchungen wurde die Leistungsfähigkeit des neu entwickelten Verfahrens zur automatisierten Erzeugung von Prozessüberwachungssystemen bewertet. Hierzu wurden für die Fertigungsverfahren Kunststoffspritzguss und Ultraschallschweißen jeweils zwei von Experten erstellte Überwachungssysteme als Vergleichssysteme gewählt. Für vier Prognoseaufgaben wurden mit dem hier beschriebenen System automatisch Prozessüberwachungssysteme erzeugt. Die Prognosegüte der automatisch erzeugten Systeme wurde den Prognosegüten der jeweiligen Vergleichssysteme gegenübergestellt. Hierbei zeigte sich die hohe Qualität der automatisch erzeugten Überwachungssysteme. In drei von vier der Prognoseaufgaben erzielten die generierten Prozessüberwachungssysteme die höchste Prognosegüte. Bei der Prognose der Berstdruckklasse für die Probekörper lagen die automatisch generierten Systeme auf Rang zwei.

Das gute Abschneiden der automatisch erzeugten Überwachungssysteme ist aus zwei Gründen besonders bemerkenswert: Zum einen sind die Vergleichssysteme allesamt hochentwickelte Verfahren, die von Experten in jeweils mehrjähriger Arbeit realisiert wurden. Und zum anderen sind insbesondere die bearbeiteten Prognoseaufgaben aus dem Umfeld des Ultraschallschweißens sehr schwierig. Für diese Aufgaben stehen vergleichsweise wenige Lerndatensätze zur Verfügung. Gleichzeitig ist die Datenmenge der Eingangsdatensätze mit jeweils etwa 400 000 Abtastwerten sehr groß. Für solche Datenmengen ist die Kenngrößenerzeugung sehr aufwändig.

Die Tatsache, dass die erzeugten Prozessüberwachungssysteme für die beiden stark unterschiedlichen Fertigungsverfahren Kunststoffspritzguss und Ultraschallschweißen sehr gute Prognoseergebnisse liefern, zeigt, dass der gewählte Ansatz der dynamischen Programmierung der Kenngrößenerzeuger sehr flexibel und leistungsfähig ist. Diese Flexibilität geht allerdings zulasten des Prozessverständnis-

ses. Während die mit der Erstellung eines herkömmlichen Prozessüberwachungssystems verbundenen Untersuchungen oft wertvolle Erkenntnisse über die Zusammenhänge im überwachten Fertigungsprozess geben, ist auf Basis der automatisch erzeugten Kenngrößen keine solche Aussage möglich.

Zusammenfassend lässt sich sagen, dass das in dieser Arbeit beschriebene Verfahren sehr gute Prozessüberwachungssysteme erzeugt. Für die Erzeugung dieser Systeme ist dabei kein Expertenwissen nötig. Das hier beschriebene System benötigt für die automatische Erzeugung eines Überwachungssystems im Durchschnitt ein bis zwei Tage. Dies bedeutet eine signifikante Verbesserung gegenüber der manuellen Erzeugung, die oft viele Monate in Anspruch nimmt.

Kapitel 6

Zusammenfassung und Ausblick

6.1 Zusammenfassung

Fertigungsstätten können in Hochlohnländern auf Dauer nur dann bestehen, wenn sie ihre Produktionsprozesse weitgehend automatisieren. Voraussetzung für eine Automatisierung ist aber die Existenz robuster und zuverlässiger Prozessüberwachungssysteme. Nur so ist es möglich, ungünstige Prozesszustände schnell zu erkennen und zu beheben.

Prozessüberwachungssysteme erfassen Sensordaten aus dem zu überwachenden Prozess. Aus den Sensordaten werden aussagekräftige Kenngrößen extrahiert. Aus den so gewonnenen Kenngrößen wird – entweder nach starren Regeln (Schwellwerte, Hüllkurven, ...) oder mittels künstlicher Intelligenz – der aktuelle Prozesszustand abgeleitet. Die intelligenten Prozessüberwachungssysteme gelten dabei als die leistungsfähigere Variante.

Bisher ist die Erstellung intelligenter Prozessüberwachungssysteme für ein Fertigungsverfahren sehr zeitaufwändig und erfordert ein hohes Maß an Expertenwissen. Dies ist ein großes Hemmnis für den flächendeckenden Einsatz solcher Systeme.

In dieser Arbeit wird erstmalig ein Verfahren für die automatische Erzeugung intelligenter Prozessüberwachungssysteme für beliebige zyklische Fertigungsprozesse vorgestellt. Die Fertigungsprozesse werden hierbei durch eine Menge von Lerndaten beschrieben. Jeder Datensatz dieser Lerndatenmenge enthält zum einen Messwerte, die während eines Fertigungszyklus aufgezeichnet wurden, und zum anderen das gewünschte Prognoseergebnis für diesen Fertigungszyklus. Ausgehend von diesen Lerndaten generiert das hier beschriebene Verfahren ein Prozessüberwachungssystem, das später auch solchen Messwerten eine Prognose zuweisen kann, die nicht in den Lerndaten enthalten waren.

Für die Umsetzung des in dieser Arbeit beschriebenen Verfahrens wurde ein generisches Prozessüberwachungssystem erarbeitet und implementiert. Dieses generische Prozessüberwachungssystem bietet die Infrastruktur für die Datenerfassung und bildet die Datenflüsse ab, die von den später zu erzeugenden Überwachungssystemen benötigt werden. Das generische Prozessüberwachungssystem enthält aber zunächst keinerlei Logik für die Verarbeitung und Bewertung der erfassten Daten. Diese Verarbeitungs- und Bewertungslogik wird von außen in Form eines Analysemodells vorgegeben. Solch ein Analysemodell ist eine Verarbeitungskette, die aus aufeinander abgestimmten Verfahren für die Signalverarbeitung, die Kenngrößenbildung, die Kenngrößenselektion und die Klassifikation besteht. Durch Setzen eines geeigneten Analysemodells lässt sich das generische Prozessüberwachungssystem an jeden Fertigungsprozess anpassen.

Mit der oben beschriebenen Konzeption ist das Erzeugen eines Prozessüberwachungssystems für einen Fertigungsprozess ein Optimierungsproblem: Man sucht

dasjenige Analysemodell, das das generische Prozessüberwachungssystem am besten an den gegebenen Fertigungsprozess anpasst. Für die Lösung dieses Optimierungsproblems wurde ein heuristisches Optimierungsverfahren mit dem Namen Artificial-Bee-Colony-Optimierung gewählt. Im Rahmen der hier beschriebenen Arbeit wurde dieses Optimierungsverfahren entscheidend erweitert, sodass es auf die gegebene Problemstellung angewandt werden konnte.

Bei den in den Analysemodellen enthaltenen Verfahren für die Kenngrößenselektion und die Klassifikation wurde jeweils auf bereits existierende Algorithmen zurückgegriffen. So stellt das System sowohl für die Kenngrößenselektion als auch für die Klassifikation eine Menge ausgewählter Algorithmen bereit. Im Rahmen der Optimierung werden dann die jeweils passenden Verfahren ausgewählt und entsprechend parametriert. Für die Signalverarbeitung und die Kenngrößenbildung musste dagegen ein noch flexiblerer Ansatz erarbeitet werden. Nur so ist es möglich, aus stark unterschiedlichen Signalen verschiedenster Fertigungsprozesse aussagekräftige Kenngrößen abzuleiten. Aus diesem Grund wurde im Rahmen der hier vorgestellten Arbeit eigens eine Programmiersprache für die Verarbeitung von Messwerten entwickelt. Im Zuge des oben beschriebenen Optimierungsverfahrens werden in dieser Programmiersprache dann automatisch Module formuliert, die sowohl die Signalverarbeitung als auch die Kenngrößenbildung realisieren. Die so beschriebenen Module werden kompiliert, in das System eingebunden und bewertet.

Die Modellauswahl liefert somit ein Analysemodell, das aus einem dynamisch programmierten Modul für die Signalverarbeitung und die Kenngrößenbildung sowie aus den jeweils ausgewählten und parametrierten Verfahren für die Kenngrößenselektion und die Klassifikation besteht. Mit diesem Modell kann das generische Überwachungssystem passend konfiguriert werden.

Für die Bewertung der Leistungsfähigkeit des Verfahrens wurden für verschiedene Fertigungsprozesse automatisch Prozessüberwachungssysteme generiert. Die Prognosegüte der so generierten Überwachungssysteme wurde mit der Prognosegüte mehrerer Prozessüberwachungssysteme verglichen, die von Experten erstellt wurden. In dieser Gegenüberstellung erzielten die automatisch generierten Systeme in drei von vier Fällen das beste Prognoseergebnis.

Für die Erzeugung eines Prozessüberwachungssystems benötigt das hier vorgestellte Verfahren auf einem handelsüblichen PC im Durchschnitt ein bis zwei Tage. Dies ist eine signifikante Verbesserung im Vergleich zu den monatelangen Untersuchungen, die mit der bisher üblichen Erzeugung durch Experten einhergehen.

Durch seine Leistungsfähigkeit und Geschwindigkeit liefert das hier vorgestellte Verfahren einen wesentlichen Beitrag zu einem Paradigmenwechsel vom experten-getriebenen hin zum automatischen Design von Prozessüberwachungssystemen.

6.2 Ausblick

Das in dieser Arbeit vorgestellte Verfahren wird bislang eingesetzt, um Prozessüberwachungssysteme zu erzeugen, die einzelne Fertigungsprozesse isoliert betrachten. In weitergehenden Forschungsarbeiten soll dieses Verfahren nun auch angewandt werden, um Überwachungssysteme für ganze Prozessketten zu generieren. Zur Überwachung solcher Prozessketten müssen prozessschrittübergreifende Kenngrößen gebildet und ausgewertet werden. Dies kann wegen der Menge der zu berücksichtigenden Messwerte und der Komplexität der Zusammenhänge auch von Experten bisher nicht ausreichend bewältigt werden.

Der wesentliche Nachteil des hier vorgestellten Verfahrens ist die schwierige Interpretierbarkeit der erzeugten Überwachungssysteme. In manuell erzeugten Prozessüberwachungssystemen geben die verwendeten Kenngrößen meist wertvolle Hinweise auf Zusammenhänge innerhalb des Prozesses. Eine Analyse dieser Kenngrößen kann so das Prozessverständnis verbessern. Aus den automatisch erzeugten Kenngrößen können solche Erkenntnisse im Allgemeinen nicht gewonnen werden. In der Grundlagenforschung werden derzeit Verfahren erarbeitet, die Kausalzusammenhänge zwischen Zufallsvariablen erkennen sollen [Janzing u. a. 2010, Peters u. a. 2011]. Für die Zukunft kann es eine lohnende Aufgabe sein, zu untersuchen, inwieweit mit solchen Verfahren auch aus automatisch erzeugten Kenngrößen Prozesswissen abgeleitet werden kann.

Auch auf den Gebieten der Modellauswahl und der Optimierungsverfahren gibt es aktuell vielversprechende Forschungsarbeiten [Özöğür-Akyüz u. a. 2011]. Der Einsatz leistungsfähigerer und schnellerer Verfahren für die Modellauswahl ist ein Ansatzpunkt für zukünftige Weiterentwicklungen des hier vorgestellten Systems. So ließe sich die automatische Erzeugung noch weiter beschleunigen.

Kapitel 7

Summary

Manufacturing sites in developed countries can only exist in the long run if their production processes are automated. Prerequisite for this automation is the existence of robust and reliable monitoring systems to detect and fix unfavorable process states.

Monitoring systems capture sensor data from the observed process. From this data, pertinent features are extracted. The current process state is deduced from these features; either based on fixed rules (thresholds, envelopes ...) or by the means of artificial intelligence. Intelligent process monitoring systems are considered to be the more powerful type.

Currently the creation of intelligent process monitoring systems for a given manufacturing method is time-consuming and requires a high degree of expert knowledge. This is a major barrier for a comprehensive application of such systems.

This thesis presents a method for the automatic creation of intelligent process monitoring systems for arbitrary cyclic production processes. These production processes can be described by a set of training data. Each data record in this training data set contains both: data that has been measured in a single cycle of the production process, and the desired prediction result for this production cycle. Based on this training data, the new method generates a process monitoring system that is able to assign predictions even to such data records that have not been part of the training data set.

For the realization of the proposed system, a generic process monitoring system was designed and implemented. This system provides the infrastructure for data acquisition and the data streams required for the generated monitoring systems. In the first stage, the generic monitoring system does not hold any program logic for processing and evaluating the acquired data. This processing and evaluation logic is provided by an external analysis model. Such a model is a processing chain integrating methods for signal preprocessing, feature generation, feature selection and classification. By setting the analysis model, the generic process monitoring system can be adapted to any manufacturing process.

With the concept described above, the creation of a process monitoring system for a manufacturing process can be reduced to an optimization problem. The goal is to find the analysis model that adapts best the generic monitoring system to the given manufacturing process. To solve this optimization problem, a heuristic optimization algorithm named Artificial Bee Colony Optimization is applied. For the method proposed in this thesis, the original Artificial Bee Colony Optimization was adapted to handle non-real valued problem spaces.

The methods which the analysis models use for feature selection and classification are selected from a pool of already existing algorithms, therefore offering a set of pre-selected methods for either task. Within the optimization, the best

matching methods are selected and their parameters are set. For the realization of the signal processing and the feature generation, a more flexible approach had to be taken in order to produce meaningful features for a broad range of diverse manufacturing processes. Thus, a new programming language was developed to explicitly fit the needs of measured data processing. In the course of the optimization, the system dynamically programs a software module that realizes both signal processing and feature generation.

Overall, the model selection creates an analysis module that contains a dynamically programmed module for signal processing and feature generation, as well as selected and configured methods for feature selection and classification. This analysis model can be used to adapt the generic process monitoring system to the given manufacturing process.

To evaluate the performance of the proposed method, process monitoring systems were created for different manufacturing processes. The prediction quality of the thus generated monitoring systems was compared with the prediction quality of monitoring systems that were manually created by experts. In this comparison, the automatically created systems outperformed the expert-created systems in three out of four cases.

The creation of a process monitoring system with the proposed method takes an average of one to two days using a standard PC. This is a significant improvement compared to the creation by experts which usually takes several months.

Due to its performance and speed the proposed method is a major contributor to a paradigm shift from expert-driven to automated design of process monitoring systems.

Literaturverzeichnis

[Abellan-Nebot u. a. 2010] Abellan-Nebot, Jose Vicente und Fernando Romero Subirón, 2010. A review of machining monitoring systems based on artificial intelligence process models. *The International Journal of Advanced Manufacturing Technology* **47**(1–4), S. 237–257. DOI 10.1007/s00170-009-2191-8.

[Akay u. a. 2012] Akay, Bahriye und Dervis Karaboga, 2012. A modified Artificial Bee Colony algorithm for real-parameter optimization. *Information Sciences* **192**, S. 120–142. DOI 10.1016/j.ins.2010.07.015.

[Alatas 2010] Alatas, Bilal, 2010. Chaotic bee colony algorithms for global numerical optimization. *Expert Systems with Applications* **37**(8), S. 5682–5687. DOI 10.1016/j.eswa.2010.02.042.

[Arlot u. a. 2010] Arlot, Sylvain und Alain Celisse, 2010. A survey of cross-validation procedures for model selection. *Statistics Surveys* **4**, S. 40–79. DOI: 10.1214/09-SS054.

[Azadeh u. a. 2013] Azadeh, A. u. a., 2013. A flexible algorithm for fault diagnosis in a centrifugal pump with corrupted data and noise based on ANN and support vector machine with hyper-parameters optimization. *Applied Soft Computing* **13**(3), S. 1478–1485. DOI 10.1016/j.asoc.2012.06.020.

[Banharnsakuna u. a. 2011] Banharnsakuna, Anan und Achalakula, Tiranee und Booncharoen Sirinaovakul, 2011. The best-so-far selection in Artificial Bee Colony algorithm. *Applied Soft Computing* **11**, S. 2888–2901. DOI 10.1016/j.asoc.2010.11.025.

[Bartlett u. a. 2002] Bartlett, Peter L. und Boucheron, Stéphane und Gábor Lugosi, 2002. Model Selection and Error Estimation. *Machine Learning* **48**(1–3), S. 85–113. DOI 10.1023/A:1013999503812.

[Bengio u. a. 2004] Bengio, Yoshua und Yves Grandvalet, 2004. No Unbiased Estimator of the Variance of K-Fold Cross-Validation. *Journal of Machine Learning Research* **5**, S. 1089–1105. Verfügbar: http://jmlr.org/papers/v5/grandvalet04a.html. Zugriff: 03.11.2011.

[Bennett u. a. 2006] Bennett, K.P. u. a., 2006. Model Selection via Bilevel Optimization. In: *IEEE International Joint Conference on Neural Network*, 16.–21. Juli 2006, Vancouver. Piscataway NJ: IEEE, S. 1922–1929. DOI 10.1109/IJCNN.2006.246935.

[Bhanu u. a. 2005] Bhanu, Bir, Lin, Yingqiang und Krzysztof Krawiec, 2005. *Evolutionary synthesis of pattern recognition systems*. New York: Springer. ISBN 978-0-387-24452-5.

[Boullé 2006] Boullé, Marc, 2006. MODL: A Bayes optimal discretization method for continuous attributes. *Machine Learning* **65**(1), S. 131–165. DOI 10.1007/s10994-006-8364-x.

[Boullé 2009] Boullé, Marc, 2009. A Parameter-Free Classification Method for Large Scale Learning. *Journal of Machine Learning Research* **10**, S. 1367–1385. Verfügbar: http://jmlr.org/papers/volume10/boulle09a/boulle09a.pdf. Zugriff: 24.11.2011.

[Bousquet u. a. 2002] Bousquet, Olivier und André Elisseeff, 2002. Stability and Generalization. *Journal of Machine Learning Research* **2**, S. 499–526. Verfügbar: http://jmlr.org/papers/volume2/bousquet02a/bousquet02a.pdf. Zugriff: 16.05.2011.

[Bousquet u. a. 2004] Bousquet, Olivier, Boucheron, Stéphane und Gábor Lugosi, 2004. Introduction to Statistical Learning Theory. In: Bousquet, Olivier, Hrsg., Luxburg, Ulrike von, Hrsg. und Gunnar Rätsch, Hrsg. *Advanced lectures on machine learning*. Berlin: Springer, S. 169–207. ISBN 978-3-540-23122-6.

[Bühlmann u. a. 2007] Bühlmann, Peter und Torsten Hothorn, 2007. Boosting Algorithms: Regularization, Prediction and Model Fitting. *Statistical Science* **22**(4). S. 477–505. DOI 10.1214/07-STS242.

[Byrne u. a. 1995] Byrne, G. u. a., 1995. Tool Condition Monitoring (TCM) — The Status of Research and Industrial Application. *CIRP Annals – Manufacturing Technology* **44**(2), S. 541–567. DOI 10.1016/S0007-8506(07)60503-4.

[Caruana u. a. 2006] Caruana, Rich und Alexandru Niculescu-Mizil, 2006. An empirical comparison of supervised learning algorithms. In: Cohen, William W., Hrsg., Moore, Andrew W., Hrsg. und William Cohen, Hrsg. *Twenty-Third International Conference on Machine Learning*, 25.–29. Juni 2006, Pittsburgh PA. New York: ACM, S. 161–168. DOI 10.1145/1143844.1143865.

[Cawley u. a. 2007a] Cawley, Gavin C., Janacek, Gareth J. und Nicola L. C. Talbot, 2007. Generalised Kernel Machines. In: *International Joint Conference on Neural Networks*, 12.–17. August 2007, Orlando FL. New York: IEEE, S. 1720–1725. DOI 10.1109/IJCNN.2007.4371217.

[Cawley u. a. 2007b] Cawley, Gavin C. und Nicola L. C. Talbot, 2007. Preventing Over-Fitting during Model Selection via Bayesian Regularisation of the Hyper-Parameters. *Journal of Machine Learning Research* **8**. Verfügbar: http://jmlr.org/papers/volume8/cawley07a/cawley07a.pdf. Zugriff: 16.05.2011.

[Cawley u. a. 2010] Cawley, Gavin C. und Nicola L. C. Talbot, 2010. On Overfitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research* **11**, S. 2079–2107. Verfügbar: http://jmlr.org/papers/volume11/cawley10a/cawley10a.pdf. Zugriff: 18.04.2011.

[Chen u. a. 2005] Chen, Pai-Hsuen, Lin, Chih-Jen und Bernhard Schölkopf, 2005. A tutorial on ν-support vector machines. *Applied Stochastic Models in Business and Industry* **21**(2), S. 111–136. DOI 10.1002/asmb.537.

[Chen u. a. 2007] Chen, Xiaozhi und Beizhi Li, 2007. Acoustic emission method for tool condition monitoring based on wavelet analysis. *The International*

Journal of Advanced Manufacturing Technology **33**(9–10), S. 968–976. DOI 10.1007/s00170-006-0523-5.

[Chi Zhou u. a. 2003] Chi Zhou u. a., 2003. Evolving accurate and compact classification rules with gene expression programming. *IEEE Transactions on Evolutionary Computation* **7**(6), S. 519–531. DOI 10.1109/TEVC.2003.819261.

[Cho u. a. 2005] Cho, Sohyung u. a., 2005. Tool breakage detection using support vector machine learning in a milling process. *International Journal of Machine Tools and Manufacture* **45**(3), S. 241–249. DOI 10.1016/j.ijmachtools.2004.08.016.

[Cui u. a. 2005] Cui, Suomin und Daniel S. Weile, 2005. Application of a parallel particle swarm optimization scheme to the design of electromagnetic absorbers. *IEEE Transactions on Antennas and Propagation* **53**(11), S. 3616–3624. DOI 10.1109/TAP.2005.858866.

[Darvin 1872] Darvin, Charles, 1872. *The origin of species by means of natural selection, or the preservation of favoured races in the struggle for life.* 6. Aufl. London: Murray.

[Demetgul 2012] Demetgul, M., 2012. Fault diagnosis on production systems with support vector machine and decision trees algorithms. *The International Journal of Advanced Manufacturing Technology.* DOI 10.1007/s00170-012-4639-5.

[Dettling u. a. 2003] Dettling, Marcel und Peter Bühlmann, 2003. Boosting for tumor classification with gene expression data. *Bioinformatics* **19**(9), S. 1061–1069. DOI 10.1093/bioinformatics/btf867.

[Doymaz u. a. 2001] Doymaz, Fuat u. a., 2001. A robust strategy for real-time process monitoring. *Journal of Process Control* **11**(4), S. 343–359. DOI 10.1016/S0959-1524(00)00004-4.

[Duch 2006] Duch, Włodzisław, 2006. Filter Methods. In: Guyon, Isabelle u. a., Hrsg. *Feature extraction*. Berlin: Springer, S. 89–117. ISBN 978-3-540-35487-1.

[Ehrenstein 2004] Ehrenstein, Gottfried W., Hrsg., 2004. *Handbuch Kunststoff-Verbindungstechnik*. München, Wien: Hanser. ISBN 978-3446226685.

[Elangovan u. a. 2010] Elangovan, M., Ramachandran, K. I. und V. Sugumaran, 2010. Studies on Bayes classifier for condition monitoring of single point carbide tipped tool based on statistical and histogram features. *Expert Systems with Applications* **37**(3), S. 2059–2065. DOI 10.1016/j.eswa.2009.06.103.

[Elbeltagi u. a. 2005] Elbeltagi, Emad, Hegazy, Tarek und Donald Grierson, 2005. Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics* **19**(1), S. 43–53. DOI 10.1016/j.aei.2005.01.004.

[Elbestawi u. a. 2006] Elbestawi, Mo A. und Mihaela Dumitrescu, 2006. Tool Condition Monitoring in Machining – Neural Networks. In: Shen, Weiming, Hrsg. *Seventh International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services*, 4.–6. September 2006, Niagara Falls, Kanada. New York, Berlin: Springer, S. 5–16. ISBN 978-0-387-36594-7.

[Ertekin u. a. 2003] Ertekin, Yalcin M., Kwon, Yongjin und Tzu-Liang Tseng, 2003. Identification of common sensory features for the control of CNC

milling operations under varying cutting conditions. *International Journal of Machine Tools and Manufacture* **43**(9), S. 897–904. DOI 10.1016/S0890-6955(03)00087-7.

[Escalante u. a. 2009] Escalante, Hugo Jair, Montes, Manuel und Luis Enrique Sucar, 2009. Particle Swarm Model Selection. *Journal of Machine Learning Research* **10**, S. 405–440. Verfügbar: http://jmlr.org/papers/volume10/escalante09a/escalante09a.pdf. Zugriff: 19.12.2011.

[Escalante u. a. 2010] Escalante, Hugo Jair, Montes, Manuel und Luis Enrique Sucar, 2010. Ensemble particle swarm model selection. In: *International Joint Conference on Neural Network*, 18.–23. Juli 2010, Barcelona. Piscataway NJ: IEEE, S. 1–8. DOI 10.1109/IJCNN.2010.5596915.

[Espejo u. a. 2010] Espejo, Pedro G. Ventura, Sebastián und Francisco Herrera, 2010. A Survey on the Application of Genetic Programming to Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **40**(2), S. 121–144. DOI 10.1109/TSMCC.2009.2033566.

[Estébanez u. a. 2005] Estébanez, César u. a., 2005. A First Attempt at Constructing Genetic Programming Expressions for EEG Classification. In: Duch, Włodzisław u. a., Hrsg. *15th International Conference on Artificial Neural Networks*, 11.–15. September 2005, Warschau. Berlin u. a.: Springer, S. 665–670. DOI 10.1007/11550822_103.

[Estébanez u. a. 2007] Estébanez, César, Aler, Ricardo und José M. Valls, 2007. A Method Based on Genetic Programming for Improving the Quality of Data-

sets in Classification Problems. *International Journal of Computer Science & Applications* **4**(1), S. 69–80. ISSN 2324-7037.

[Ferreira u. a. 2012] Ferreira, Artur J. und Mário A. T. Figueiredo, 2012. Efficient feature selection filters for high-dimensional data. *Pattern Recognition Letters* **33**(13), S. 1794–1804. DOI 10.1016/j.patrec.2012.05.019.

[Friedman u. a. 2000] Friedman, Jerome, Hastie, Trevor und Robert Tibshirani, 2000. Additive logistic regression: a statistical view of boosting. *Annals of Statistics* **28**(2), S. 337–407. ISSN 0090-5364.

[Gamma 1995] Gamma, Erich, 1995. *Design patterns: Elements of reusable object-oriented software*. Reading MA: Addison-Wesley. ISBN 978-0201633610.

[Geman u. a. 1992] Geman, Stuart, Bienenstock, Elie und René Doursat, 1992. Neural Networks and the Bias/Variance Dilemma. *Neural Computation* **4**(1), S. 1–58. DOI 10.1162/neco.1992.4.1.1.

[Ghosh u. a. 2007] Ghosh, N. u. a., 2007. Estimation of tool wear during CNC milling using neural network-based sensor fusion. *Mechanical Systems and Signal Processing* **21**(1), S. 466–479. DOI 10.1016/j.ymssp.2005.10.010.

[Gorissen u. a. 2009] Gorissen, Dirk, Dhaene, Tom und Filip De Turck, 2009. Evolutionary Model Type Selection for Global Surrogate Modeling. *Journal of Machine Learning Research* **10**, S. 2039–2078. Verfügbar: http://jmlr.org/papers/volume10/gorissen09a/gorissen09a.pdf. Zugriff: 04.05.2011.

[Guo u. a. 2005a] Guo, Hong, Jack, Lindsay B. und Asoke K. Nandi, 2005. Feature Generation Using Genetic Programming With Application to Fault

Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **35**(1), S. 89–99. DOI 10.1109/TSMCB.2004.841426.

[Guo u. a. 2005b] Guo, Hong und Asoke K. Nandi, 2005. Breast Cancer Diagnosis Using Genetic Programming Generated Feature. In: Barros, Allan, Hrsg. *IEEE Workshop on Machine Learning for Signal Processing* , 28.–30. September 2010, Mystic CT. Piscataway NJ: IEEE, S. 215–220. DOI 10.1109/MLSP.2005.1532902.

[Guyon 2008] Guyon, Isabelle, 2008. Practical Feature Selection: from Correlation to Causality. In: Fogelman-Soulié, Françoise, Hrsg. *Mining massive data sets for security*. Amsterdam: IOS Press, S. 27–43. DOI 10.3233/978-1-58603-898-4-27.

[Guyon 2009] Guyon, Isabelle, 2009. A practical guide to model selection. In: Marie, J., Hrsg. *Machine Learning Summer School*. Berlin, Heidelberg: Springer. Verfügbar: http://eprints.pascal-network.org/archive/00005768/01/guyon-mlss.pdf. Zugriff: 15.07.2011.

[Guyon u. a. 2003] Guyon, Isabelle und André Elisseeff, 2003. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research* **3**, S. 1157–1182. Verfügbar: http://machinelearning.wustl.edu/mlpapers/paper_files/GuyonE03.pdf. Zugriff: 28.05.2013.

[Guyon u. a. 2006a] Guyon, Isabelle u. a., 2006. Performance Prediction Challenge. In: *IEEE International Joint Conference on Neural Networks*, 16.–21. Juli 2006, Vancouver. Piscataway NJ: IEEE, S. 1649–1656. DOI 10.1109/IJCNN.2006.246632

[Guyon u. a. 2006b] Guyon, Isabelle und André Elisseeff, 2006. An Introduction to Feature Extraction. In: Guyon, Isabelle u. a., Hrsg. *Feature extraction.* Berlin: Springer, S. 1–25. ISBN 978-3-540-35487-1.

[Guyon u. a. 2008a] Guyon, Isabelle u. a., 2008. Analysis of the IJCNN 2007 agnostic learning vs. prior knowledge challenge. *Neural Networks* **21**(2–3), S. 544–550. ISSN 0893-6080.

[Guyon u. a. 2008b] Guyon, Isabelle u. a., 2008. Design and Analysis of the Causation and Prediction Challenge. In: Guyon, Isabelle u. a., Hrsg. *Causation and Prediction Challenge at WCCI 2008*, 1.–6. Juni 2008, Hong Kong. S. 1–33. Verfügbar: http://jmlr.org/proceedings/papers/v3/guyon08a/guyon08a.pdf. Zugriff: 29.04.2011.

[Guyon u. a. 2010] Guyon, Isabelle u. a., 2010. Model Selection: Beyond the Bayesian/Frequentist Divide. *Journal of Machine Learning Research* **11**, S. 61–87. Verfügbar: http://jmlr.org/papers/volume11/guyon10a/guyon10a.pdf. Zugriff: 31.10.2011.

[Haberstroh u. a. 2004] Haberstroh, Edmund und Kai Kuhlmann, 2004. Prozessführung beim Ultraschallschweißen. *Kunststoffe* **84**(7), S. 35–38. ISSN 0023-5563.

[Hall 1999] Hall, Mark A., 1999. *Correlation-based Feature Subset Selection for Machine Learning.* Hamilton, New Zealand, University of Waikato, Diss. Verfügbar: http://www.cs.waikato.ac.nz/ mhall/thesis.pdf. Zugriff: 20.03.2013.

[Hand u. a. 2001] Hand, David J. und Keming Yu, 2001. Idiot's Bayes—Not So Stupid After All?. *International Statistical Review* **69**(3), S. 385–398. DOI 10.1111/j.1751-5823.2001.tb00465.x.

[Hastie u. a. 2009] Hastie, Trevor, Tibshirani, Robert und Jerome Friedman, 2009. *The elements of statistical learning: Data mining, inference, and prediction*. 2nd edition. New York: Springer. ISBN 978-0-387-84858-7.

[Hofmann u. a. 2008] Hofmann, Thomas, Schölkopf, Bernhard und Alexander J. Smola, 2008. Kernel methods in machine learning. *The Annals of Statistics* **36**(3), S. 1171–1220. DOI 10.1214/009053607000000677.

[Holte 1993] Holte, Robert C., 1993. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning* **11**, S. 63–91. DOI 10.1023/A:1022631118932.

[Hsu u. a. 2010] Hsu, Chih-Wei, Chang, Chih-Chung und Chih-Jen Lin, 2010. *A Practical Guide to Support Vector Classification*. Department of Computer Science, National Taiwan University, Taipei. Forschungsbericht. Verfügbar: http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf. Zugriff: 03.04.2013.

[Iswandy u. a. 2009] Iswandy, Kuncup und Andreas König, 2009. Methodology, Algorithms, and Emerging Tool for Automated Design of Intelligent Integrated Multi-Sensor Systems. *Algorithms* **4**(2), S. 1368–1409. DOI 10.3390/a2041368.

[Jankowski u. a. 2006] Jankowski, Norbert und Krzysztof Grabczewski, 2006. Learning Machines. In: Guyon, Isabelle u. a., Hrsg. *Feature extraction*. Berlin: Springer, S. 29–64. ISBN 978-3-540-35487-1.

[Janzing u. a. 2010] Janzing, Dominik und Bernhard Schölkopf, 2010. Causal Inference Using the Algorithmic Markov Condition. *IEEE Transactions on Information Theory* **56**(10), S. 5168–5194. DOI 10.1109/TIT.2010.2060095.

[Jardine u. a. 2006] Jardine, Andrew K.S, Lin, Daming und Dragan Banjevic, 2006. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing* **20**(7), S. 1483–1510. DOI 10.1016/j.ymssp.2005.09.012.

[Jelodar u. a. 2006] Jelodar, M. S., Fakhraie, S. M. und M. N. Ahmadabadi, 2006. Two-Stage Morphological Filter Design Using Genetic Algorithm. In: *IEEE International Conference on Engineering of Intelligent Systems*, 22.–23. April 2006, Islamabad, Pakistan. Piscataway NJ: IEEE, S. 1–5. DOI 10.1109/ICEIS.2006.1703152.

[Jemielniak 2000] Jemielniak, Krzysztof, 2000. Some aspects of AE application in tool condition monitoring. *Ultrasonics* **38**(1–8), S. 604–608. DOI 10.1016/S0041-624X(99)00195-X.

[Jemielniak u. a. 2012] Jemielniak, Krzysztof u. a., 2012. Tool condition monitoring based on numerous signal features. *The International Journal of Advanced Manufacturing Technology* **59**(1–4), S. 73–81. DOI 10.1007/s00170-011-3504-2.

[Johannaber u. a. 2004] Johannaber, Friedrich und Walter Michaeli, 2004. *Handbuch Spritzgiessen*. 2. Aufl. München: Hanser. ISBN 978-3446229662.

[Kameyama 2009] Kameyama, Keisuke, 2009. Particle Swarm Optimization – A Survey. *IEICE Transactions on Information and Systems* **92**(7), S. 1354–1361. ISSN 1745-1361.

[Karaboga 2005] Karaboga, Dervis, 2005. *An idea based on honey bee swarm for numerical optimization: Technical Report- TR06*. Erciyes University, Kayseri, Türkei. Verfügbar: http://mf.erciyes.edu.tr/abc/pub/tr06_2005.pdf. Zugriff: 10.02.2011.

[Karaboga u. a. 2007] Karaboga, Dervis und Bahriye Basturk, 2007. A powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. *Journal of Global Optimization* **39**(3), S. 459–471. DOI 10.1007/s10898-007-9149-x.

[Karaboga u. a. 2009] Karaboga, Dervis und Bahriye Akay, 2009. A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation* **214**, S. 108–132. DOI 10.1016/j.amc.2009.03.090.

[Kaupp u. a. 2010a] Kaupp, Markus und Joachim Neher, 2010. A new Approach for Quality Monitoring in Small Batch Processes. In: Lien, Terje K., Hrsg. *3rd CIRP Conference on Assembly Technologies and Systems*, 1.–3. Juni 2010, Trondheim. Trondheim: Tapir Academic Press, S. 205–208.

[Kaupp u. a. 2010b] Kaupp, Markus und Joachim Neher, 2010. Active Learning-Based Adaptive Control in Low-Runner Processes. In: Teti, Roberto, Hrsg. *7th CIRP International Conference on Intelligent Computation in Manufacturing Engineering*, 23.–25. Juni 2010, Capri, S. 1–4.

[Kaupp u. a. 2011] Kaupp, Markus und Joachim Neher, 2011. Active Learning Based Process Monitoring – An Application Report. In: Spath, Dieter, Hrsg. und Rolf Ilg, Hrsg. *21th International Conference on Production Research*, 31. Juli–4. August 2011, Stuttgart. Stuttgart: Fraunhofer Verlag, S. 1–5. ISBN 978-3-8396-0293-5.

[Kaupp u. a. 2012a] Kaupp, Markus und Joachim Neher, 2012. Überwachung von Prozessen mit kleiner Losgröße: Active Learning in der Prozessüberwachung. *atp edition* **54**(34–39), S. 34–39. ISSN 0178-2320.

[Kaupp u. a. 2012b] Kaupp, Markus und Joachim Neher, 2012. *Schlussbericht zum Forschungsvorhaben „Adaptive Klassifikation zur Inline-Qualitätssicherung zyklischer Fertigungsprozesse (Adaptive Klassifikation)“*. Fraunhofer IPA, Forschungsgemeinschaft Qualität. Forschungsbericht.

[Kearns u. a. 1997] Kearns, Michael u. a., 1997. An Experimental and Theoretical Comparison of Model Selection Methods. *Machine Learning* **27**, S. 7–50. DOI 10.1023/A:1007344726582.

[Kennedy u. a. 1995] Kennedy, J. und R. Eberhart, 1995. Particle swarm optimization. In: *IEEE International Conference on Neural Networks*, 27. November–1. Dezember 1995, Perth. Piscataway NJ: IEEE, S. 1942–1948. DOI 10.1109/ICNN.1995.488968.

[Khoshgoftaar u. a. 2010] Khoshgoftaar, Taghi M., Hulse, Jason van und Amri Napolitano, 2010. Comparing Boosting and Bagging Techniques With Noisy and Imbalanced Data. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans* **41**(3), S. 552 - 568. DOI: 10.1109/TSMCA.2010.2084081.

[Kim u. a. 2011] Kim, Hack-Eun u. a., 2011. Bearing fault prognosis based on health state probability estimation. *Expert Systems with Applications* **39**(5), S. 5200–5213. DOI 10.1016/j.eswa.2011.11.019.

[Kitazawa u. a. 2011] Kitazawa, Guillermo, Siller, Hector R. und Jose Vicente Abellan-Nebot, 2011. Comparison of Analytical and Artificial Intelligent

Models for Quality Assurance in Micro-milling Operations. In: Batyrshin, Ildar, Hrsg. und Grigori Sidorov, Hrsg. *10th Mexican International Conference on Artificial Intelligence*, 26. November–4. Dezember 2011, Puebla, Mexico. Los Alamitos CA: IEEE, S. 88–93. DOI 10.1109/MICAI.2011.34.

[Kohavi u. a. 1997] Kohavi, R. und George H. John, 1997. Wrappers for feature subset selection. *Artificial Intelligence* **97**(1–2), S. 273–324. DOI 10.1016/S0004-3702(97)00043-X.

[Kohonen 1990] Kohonen, Teuvo, 1990. The self-organizing map. *Proceedings of the IEEE* **78**(9), S. 1464–1480. DOI 10.1109/5.58325.

[König 2007] König, Erwin, 2007. Dynamische Temperaturmessung als Ausschussprophylaxe. *Kunststoffe* **97**(6), S. 56–60. ISSN 0023-5563.

[Kononenko 1994] Kononenko, Igor, 1994. Estimating attributes: Analysis and extensions of RELIEF. In: Bergadano, Francesco, Hrsg. und Luc de Raedt, Hrsg. *European Conference on Machine Learning*, , 6.–8. April 1994, Catania, Italien. Berlin, New York: Springer, S. 171–182. DOI 10.1007/3-540-57868-4_57

[Kovač u. a. 2011] Kovač, Pavel u. a., 2011. A review of machining monitoring systems. *Journal of Production Engineering* **14**(1), S. 1–16. ISSN 1821-4932.

[Lei Guo u. a. 2009] Lei Guo, Jin Chen und Xinglin Li, 2009. Rolling Bearing Fault Classification Based on Envelope Spectrum and Support Vector Machine. *Journal of Vibration and Control***15**(9), S. 1349–1363. DOI 10.1177/1077546308095224.

[Li 2010] Li, Ping, 2010. Robust LogitBoost and Adaptive Base Class (ABC) LogitBoost. In: Grunwald, Peter, Hrsg., Spirtes, Peter, Hrsg. und Jeff Bilmes, Hrsg. *Uncertainty in artificial intelligence*. Corvallis OR: AUAI Press, S. 302–311. Verfügbar: http://event.cwi.nl/uai2010/papers/UAI2010_0282.pdf. Zugriff: 04.04.2013.

[Liang u. a. 2004] Liang, Steven Y., Hecker, Rogelio L. und Robert G. Landers, 2004. Machining Process Monitoring and Control: The State-of-the-Art. *Journal of Manufacturing Science and Engineering* **126**(2), S. 297–310. DOI 10.1115/1.1707035.

[Lin u. a. 2005] Lin, Y. und B. Bhanu, 2005. Evolutionary Feature Synthesis for Object Recognition. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)* **35**(2), S. 156–171. DOI 10.1109/TSMCC.2004.841912.

[Lin u. a. 2008] Lin, Jung-Yi u. a., 2008. Classifier design with feature selection and feature extraction using layered genetic programming. *Expert Systems with Applications* **34**(2), S. 1384–1393. DOI 10.1016/j.eswa.2007.01.006.

[Liu u. a. 1996] Liu, Huan und Rudy Setiono, 1996. A probabilistic approach to feature selection – A filter solution. In: Saitta, Lorenza, Hrsg. *Thirteenth International Conference on Machine Learning*, 3.–6. Juli 1996, Bari. San Francisco CA: Kaufmann, S. 319–327. Verfügbar: http://www.public.asu.edu/ huanliu/papers/ml96.ps. Zugriff: 20.03.2013.

[Liu u. a. 2005] Liu, Huan und Lei Yu, 2005. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering* **17**(4), S. 491–502. DOI 10.1109/TKDE.2005.66.

[Lutz 2006] Lutz, Roman Werner, 2006. LogitBoost with Trees Applied to the WCCI 2006 Performance Prediction Challenge Datasets. In: *IEEE International Joint Conference on Neural Network*, 16.–21. Juli 2006, Vancouver. Piscataway NJ: IEEE, S. 1657–1660. DOI 10.1109/IJCNN.2006.246633.

[Mandrikova u. a. 2011] Mandrikova, O. V. u. a., 2011. New wavelet-based approach intended for the analysis of subtle features of complex natural signals. *Pattern Recognition and Image Analysis* **21**(2), S. 300–303. DOI 10.1134/S1054661811020726.

[Matić u. a. 2012] Matić, Dragan u. a., 2012. Support vector machine classifier for diagnosis in electrical machines: Application to broken bar. *Expert Systems with Applications* **39**(10), S. 8681–8689. DOI 10.1016/j.eswa.2012.01.214.

[Michalewicz u. a. 2004] Michalewicz, Zbigniew und David B. Fogel, 2004. *How to solve it: Modern heuristics*. 2nd edition. Berlin, New York: Springer. ISBN 978-3540224945.

[Molinaro 2005] Molinaro, A. M., 2005. Prediction error estimation: a comparison of resampling methods. *Bioinformatics* **21**(15), S. 3301–3307. DOI 10.1093/bioinformatics/bti499.

[Muharram u. a. 2005] Muharram, Mohammed und George D. Smith, 2005. Evolutionary constructive induction. *IEEE Transactions on Knowledge and Data Engineering* **17**(11), S. 1518–1528. DOI 10.1109/TKDE.2005.182.

[Muni u. a. 2006] Muni, Durga Prasad, Pal, Nikhil R. und Jyotirmoy Das, 2006. Genetic programming for simultaneous feature selection and classifier design. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* **36**(1), S. 106–117. DOI 10.1109/TSMCB.2005.854499.

[Muralidharan u. a. 2012] Muralidharan, V. und V. Sugumaran, 2012. A comparative study of Naïve Bayes classifier and Bayes net classifier for fault diagnosis of monoblock centrifugal pump using wavelet analysis. *Applied Soft Computing* **12**(8), S. 2023–2029. DOI 0.1016/j.asoc.2012.03.021.

[Narasimhan 2009] Narasimhan, Harikrishna, 2009. Parallel Artificial Bee Colony (PABC) Algorithm. In: Abraham, Ajith, Hrsg. *World Congress on Nature & Biologically Inspired Computing*, 21.–22. Dezember 2009, Bhubaneswar, India. Piscataway N.J: IEEE, S. 306–311. DOI 10.1109/NABIC.2009.5393726.

[Neher 2012] Neher, Joachim, 2012. *Neuro-Fuzzy-Modellierung zur umfassenden Prozessüberwachung am Beispiel des Ultraschallschweißens von Kunststoffteilen*. Stuttgart: Fraunhofer Verlag. ISBN 978-3-8396-0424-3. Stuttgart, Univ., Diss., 2011. Verfügbar: http://publica.fraunhofer.de/eprints/urn:nbn:de:bsz:93-opus-75874.pdf. Zugriff: 25.07.2014

[Neher u. a. 2010] Neher, Joachim und Günther Fischer, 2010. *Abschlussbericht zum Verbundforschungsprojekt „SC-QUPUS: Signalbasiertes Clustering zur qualitätsorientierten Prozessüberwachung beim Ultraschallschweißen thermoplastischer Kunststoffprodukte“*. Fraunhofer IPA, Hochschule Esslingen. Forschungsbericht.

[Özöğür-Akyüz u. a. 2011] Özöğür-Akyüz, Süreyya, Ünay, Devrim und Alexander J. Smola, 2011. Guest editorial: model selection and optimization in machine learning. *Machine Learning* **85**(1–2), S. 1–2. DOI 10.1007/s10994-011-5261-8.

[Parpinelli u. a. 2011] Parpinelli, Rafael Stubs, Benitez, César Manuel Vargas und Heitor Silvério Lopes, 2011. Parallel Approaches for the Artificial Bee Colony Algorithm. In: Panigrahi, Bijaya Ketan, Hrsg. Shi, Yuhui, Hrsg. und

Meng-Hiot Lim, Hrsg. *Handbook of swarm intelligence*. Berlin, Heidelberg: Springer, S. 329–345. DOI 10.1007/978-3-642-17390-5_14.

[Pedersen u. a. 2010] Pedersen, Magnus Erik Hvass und Andrew John Chipperfield, 2010. Simplifying Particle Swarm Optimization. *Applied Soft Computing* **10**(2), S. 618–628. DOI 10.1016/j.asoc.2009.08.029.

[Peters u. a. 2011] Peters, Jonas, Janzing, Dominik und Bernhard Schölkopf, 2011. Causal Inference on Discrete Data Using Additive Noise Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **33**(12), S. 2436–2450. DOI 10.1109/TPAMI.2011.71.

[Poli 2008] Poli, Riccardo, 2008. Analysis of the Publications on the Applications of Particle Swarm Optimisation. *Journal of Artificial Evolution and Applications* **2008**, S. 1–10. DOI 10.1155/2008/685175.

[Poli u. a. 2008] Poli, Riccardo u. a., 2008. *A field guide to genetic programming*. [S.I.]: Lulu Press. ISBN 978-1-4092-0073-4.

[Pöyhönen u. a. 2004] Pöyhönen, Sanna, Jover, Pedro und Heikki Hyötyniemi, 2004. Signal processing of vibrations for condition monitoring of an induction motor. In: *First International Symposium on Control, Communications and Signal Processing*, Communications and Signal Processing, 21.–24. März 2004. Hammamet, Tunesien. Piscataway NJ: IEEE, S. 499–502. DOI 10.1109/ISCCSP.2004.1296338.

[Press 2007] Press, William H., 2007. *Numerical recipes: The art of scientific computing*. 3rd edition. Cambridge UK, New York: Cambridge University Press. ISBN 978-0521880688

[Qu u. a. 2011] Qu, Jian und Ming J. Zuo, 2011. An LSSVR-based algorithm for online system condition prognostics. *Expert Systems with Applications* **39**(5), S. 6089–6102. DOI DOI: 10.1016/j.eswa.2011.12.002.

[Quinlan 1986] Quinlan, J.R., 1986. Induction of Decision Trees. *Machine Learning* **1**, S. 81–106. DOI 10.1007/BF00116251.

[Reif u. a. 2014] Reif, Matthias u. a., 2014. Automatic classifier selection for non-experts. *Pattern Analysis and Applications* **17**(1), S. 83–96. DOI 10.1007/s10044-012-0280-z.

[Ribeiro 2005] Ribeiro, B., 2005. Support Vector Machines for Quality Monitoring in a Plastic Injection Molding Process. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **35**(3), S. 401–410. DOI 10.1109/TSMCC.2004.843228.

[Rizki u. a. 2002] Rizki, Mateen M., Zmuda, Michael A. und Louis A. Tamburino, 2002. Evolving pattern recognition systems. *IEEE Transactions on Evolutionary Computation* **6**(6), S. 594–609. DOI 10.1109/TEVC.2002.806167.

[Sakthivel u. a. 2012] Sakthivel, N.R., Nair, Binoy. B. und V. Sugumaran, 2012. Soft computing approach to fault diagnosis of centrifugal pump. *Applied Soft Computing* **12**(5), S. 1574–1581. DOI 10.1016/j.asoc.2011.12.009.

[Salibian-Barrera u. a. 2008] Salibian-Barrera, Matias und Stefan van Aelst, 2008. Robust model selection using fast and robust bootstrap. *Computational Statistics & Data Analysis* **52**(12), S. 5121–5135. DOI 10.1016/j.csda.2008.05.007.

[Schapire u. a. 1999] Schapire, Robert E. und Yoram Singer, 1999. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning* **37**(3), S. 297–336. DOI 10.1023/A:1007614523901.

[Schlittgen u. a. 2001] Schlittgen, Rainer und Bernd Streitberg, 2001. *Zeitreihenanalyse*. 9., unwesentlich veränderte Aufl. München, Wien: Oldenbourg. ISBN 978-3486257250.

[Schmidberger u. a. 1995] Schmidberger, Ernst u. a., 1995. Neuronale Netze beim Spritzgießen: Ein neuer Weg zur Qualitätssicherung.*Kunststoffe* **85**(5), S. 620–623. ISSN 0023-5563.

[Schmidberger u. a. 2005a] Schmidberger, Ernst und Günther Fischer, 2005. *Abschlussbericht zum Verbundforschungsprojekt „Qualitätsprognose mittels Prozessinformation beim Ultraschall- und Vibrationsschweißen (QP-UVS)“*. Fraunhofer IPA, Hochschule Esslingen. Forschungsbericht.

[Schmidberger u. a. 2005b] Schmidberger, Ernst und Joachim Neher, 2005. *Leitfaden für die qualitätsorientierte Prozessüberwachung und -regelung zyklischer Produktionsprozesse (NEPRES II)*. Frankfurt am Main: FQS – Forschungsgemeinschaft Qualität e.V.

[Schmidberger u. a. 2008] Schmidberger, Ernst und Joachim Neher, 2008. *Leitfaden für die Qualitätssicherung und -überwachung von LowRunner-Prozessen (QSLR)*. Frankfurt am Main: FQS – Forschungsgemeinschaft Qualität e.V.

[Schölkopf u. a. 2002] Schölkopf, Bernhard und Alexander J. Smola, 2002. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. Cambridge MA: MIT Press. ISBN 978-0262194754.

[Seiffert u. a. 2008] Seiffert, Chris u. a., 2008. Resampling or Reweighting: A Comparison of Boosting Implementations. In: *20th IEEE International Conference on Tools with Artificial Intelligence*, 3.–5. November 2008, Dayton OH. Los Alamitos CA: IEEE, S. 445–451. DOI 978-0-7695-3440-4.

[Sharif u. a. 2010] Sharif, Muhammad u. a., 2010. Adaptive Filter and Morphological Operators Using Binary PSO. In: Zhu, Rongbo, Hrsg. *International Conference on Information Computing and Applications*, 15.–18. Oktober 2010, Tangshan, China. Berlin: Springer, S. 525–532. DOI 10.1007/978-3-642-16167-4_67.

[Song u. a. 2012] Song, Qinbao, Wang, Guangtao und Chao Wang, 2012. Automatic recommendation of classification algorithms based on data set characteristics. *Pattern Recognition* **45**(7), S. 2672–2689. DOI 10.1016/j.patcog.2011.12.025.

[Subotic u. a. 2010] Subotic, Milos, Tuba, Milan und Nadezda Stanarevic, 2010. Parallelization of the Artificial Bee Colony (ABC) Algorithm. In: Munteanu, Viorel u. a., Hrsg. *Recent Advances in Neural Networks, Fuzzy Systems & Evolutionary Computing*. [S. l.]: WSEAS, S. 191–196. ISBN 978-960-474-195-3. Verfügbar: http://www.wseas.us/e-library/conferences/2010/Iasi/NNECFS/NNECFS-27.pdf. Zugriff: 11.02.2011.

[Teodorović 2009] Teodorović, Dušan, 2009. Bee Colony Optimization (BCO). In: Chee, Peng Lim, Hrsg., Jain, L. C., Hrsg. und Satchidananda Dehuri, Hrsg. *Innovations in swarm intelligence*. Berlin: Springer, S. 39–60. DOI 10.1007/978-3-642-04225-6_3.

[Teti u. a. 2010] Teti, R. u. a., 2010. Advanced monitoring of machining operations. *CIRP Annals – Manufacturing Technology* **59**(2), S. 717–739. DOI 10.1016/j.cirp.2010.05.010.

[Varewyck u. a. 2011] Varewyck, Matthias und Jean-Pierre Martens, 2011. A Practical Approach to Model Selection for Support Vector Machines With a Gaussian Kernel. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **41**(2), S. 330–340. DOI 10.1109/TSMCB.2010.2053026.

[Verl u. a. 2009] Verl, Alexander u. a., 2009. Sensorless automated condition monitoring for the control of the predictive maintenance of machine tools. *CIRP Annals – Manufacturing Technology* **58**(1), S. 375–378. DOI 10.1016/j.cirp.2009.03.039.

[Wei Chu 2006] Wei Chu, 2006. Model Selection: An Empirical Study on Two Kernel Classifiers. In: *IEEE International Joint Conference on Neural Network*, 16.–21. Juli 2006, Vancouver. Piscataway NJ, IEEE, S. 1673–1679. DOI 10.1109/IJCNN.2006.246636.

[Westkämper u. a. 2009] Westkämper, Engelbert, Hrsg. und Erich Zahn, Hrsg., 2009. *Wandlungsfähige Produktionsunternehmen: Das Stuttgarter Unternehmensmodell*. Berlin: Springer. ISBN 978-3-540-68890-7.

[Wiendahl u. a. 2004] Wiendahl, Hans-Peter, Hrsg., Gerst, Detlef, Hrsg. und Lars Keunecke, Hrsg., 2004. *Variantenbeherrschung in der Montage: Konzept und Praxis der flexiblen Produktionsendstufe*. Berlin u. a.: Springer. ISBN 978-3-642-18947-0

[Wurm 2006] Wurm, Hermine, 2006. 100 % Qualitätskontrolle. *Plastverarbeiter* **13**(8), S. 82. ISSN 0032-1338.

[Yu 2011] Yu, Jianbo, 2011. A hybrid feature selection scheme and self-organizing map model for machine health assessment. *Applied Soft Computing* **11**(5), S. 4041–4054. DOI 10.1016/j.asoc.2011.03.026.

[Yu u. a. 2000] Yu, Nong, Li, Yushu und Qifu Xu, 2000. Genetic training algorithm for morphological filters. In: *5th International Conference on Signal Processing, 16th World Computer Congress*, 21.–25 August 2000, Peking. Piscataway NJ: IEEE, S. 476–479. DOI 10.1109/ICOSP.2000.894535.

[Zhu u. a. 2009] Zhu, Kunpeng, Wong, Yoke San und Geok Soon Hong, 2009. Wavelet analysis of sensor signals for tool condition monitoring: A review and some new results. *International Journal of Machine Tools and Manufacture* **49**(7–8), S. 537–553. DOI 10.1016/j.ijmachtools.2009.02.003,

Eine Voraussetzung für die Automatisierung von Produktionsprozessen ist die Existenz zuverlässiger Prozessüberwachungssysteme. Solche Systeme erfassen Sensordaten aus dem zu überwachenden Prozess und leiten daraus den aktuellen Prozesszustand ab. In dieser Arbeit wird ein Verfahren für die automatische Erzeugung von Prozessüberwachungssystemen für beliebige zyklische Fertigungsprozesse vorgestellt. Die erzeugten Systeme verwenden dabei Methoden aus dem Gebiet des maschinellen Lernens.
Für die Umsetzung der automatischen Erzeugung wurde ein generisches Prozessüberwachungssystem implementiert, das mithilfe eines Optimierungsverfahrens an den zu überwachenden Fertigungsprozess angepasst werden kann. Für die Optimierung wird dabei ein eigens erweiterter Artificial-Bee-Colony-Algorithmus eingesetzt.

ISBN 978-3-8396-0780-0

9 783839 607800

FRAUNHOFER VERLAG