Modellierung dynamischer Prozesse mit radialen Basisfunktionen

Dissertation

zur Erlangung des Doktorgrades

der Mathematisch-Naturwissenschaftlichen Fakultäten

der Georg-August-Universität zu Göttingen

vorgelegt von

Jörg Dittmar

aus Kassel

Göttingen 2010

D7

Referent:	Prof. Dr. Ulrich Parlitz
Korreferent:	Prof. Dr. Werner Lauterborn
Tag der mündlichen Prüfung:	20.08.2010

Inhaltsverzeichnis

Kapitel 1
Einleitung

Die Wissenschaft beschäftigt sich mit der Untersuchung von Systemen, um deren zugrundeliegenden Gesetzmäßigkeiten zu bestimmen. Dabei kann es sich um ganz unterschiedliche Systeme handeln, die z.B. den Natur-, Sozial- oder Finanzwissenschaften zuzuordnen sind. Eine Methode besteht darin, aus den entsprechenden Grundprinzipien – in der Physik bspw. die Newton'schen Gesetze – ein mathematisches Modell zu konstruieren, das für gewöhnlich aus einem Satz von Differentialgleichungen besteht, und diese zu lösen. Experimentelle Daten dienen dann dazu, die Gültigkeit des Modells zu verifizieren und ggf. Werte für freie Parameter (Konstanten) des Differentialgleichungssystems zu bestimmen.

Oft ist diese Methode jedoch nicht anwendbar, weil für die untersuchten Systeme z.B. die zugrundeliegenden Gesetze noch nicht bekannt oder die Zusammenhänge zu kompliziert sind und sich nicht auf „Laborbedingungen" reduzieren (d.h. genügend genau durch ein handhabbares, aus Grundprinzipien abgeleitetes mathematisches Modell beschreiben) lassen. Aufgrund des technischen Fortschritts mit immer leistungsfähigeren Computern und Messtechniken ist es allerdings oft möglich, große Mengen an Messdaten von solchen Systemen zu gewinnen. Anschließend kann versucht werden, Zusammenhänge zwischen diesen Daten aufzudecken und mathematisch zu beschreiben. Bei dieser Art von Modellierung handelt es sich um eine *datengesteuerte* Modellierung, der kein spezifisches mathematisches Modell des Systems zugrundeliegt, sondern bei der Zusammenhänge innerhalb der Daten ausschließlich aus den Daten selbst gewonnen werden sollen. Man spricht dann auch von *Black-Box*-Modellierung.

Der Black-Box-Ansatz wird auch in der vorliegenden Arbeit verfolgt. Auch wenn dabei zur Demonstration der Methoden oft numerisch erzeugte Daten von dynamischen Systemen verwendet werden, für deren Generierung mittels eines numerischen DGL-Lösers natürlich die entsprechenden Differentialgleichungen bekannt

sein müssen, so wird die Kenntnis der Systemgleichungen lediglich zur *Erzeugung* dieser Daten verwendet. In die anschließende *Modellierung* mittels der hier verwendeten Black-Box-Techniken gehen hingegen nur die Daten selbst ein, jedoch keinerlei weitergehenden Informationen.

Das Modellierungsproblem besteht nun allgemein darin, deterministische Zusammenhänge zwischen gemessenen Daten aufzudecken und zu approximieren. In der statistischen Lerntheorie wird dies als das *Lernen* des Zusammenhangs bezeichnet. Man unterscheidet dabei zwischen nicht überwachtem Lernen (*unsupervised learning*) und überwachtem oder angeleitetem Lernen (*supervised learning*). Bei Ersterem werden Zusammenhänge innerhalb der Daten gesucht, wozu z.B. die Schätzung der den Daten zugrundeliegenden Wahrscheinlichkeitsdichteverteilung gehört oder die Identifikation von Clustern innerhalb der Daten. Als weiteres Anwendungsgebiet sei hier noch die (nichtlineare) Dimensionsreduktion genannt, bei der die Korrelationen zwischen den Daten dazu genutzt werden, um eine niedrigdimensionalere Repräsentation der Daten zu gewinnen bzw. um relevante Merkmale (*features*) aus den Daten zu generieren [1, 2].

Beim Supervised Learning hingegen werden die Daten in unabhängige Variablen x und davon abhängige Variablen y unterteilt und ein deterministischer Zusamenhang in Form einer (unbekannten) Abbildung $f : x \mapsto y$ angenommen. Die Aufgabe der Modellierung besteht dann darin, eine Approximation dieser Abbildung zu finden. Dazu präsentiert man dem Modellierungsalgorithmus die gemessenen Daten in Form von Trainingsexemplaren $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ mit Eingaben $\boldsymbol{x}_i$ und zugehörigen Ausgaben y_i. Anhand dieser „Lernbeispiele“ soll der Modellierungsalgorithmus dann den funktionellen Zusammenhang lernen und anschließend auch für neue Eingaben $\boldsymbol{x}^{(\text{new})}$, die nicht Eingang in das Training gefunden haben, eine möglichst gute Approximation der richtigen Ausgabe $y^{(\text{new})}$ liefern. Je nachdem, ob es sich bei der Ausgabevariable y um eine kontinuierliche oder eine kategorische Variable handelt, unterscheidet man beim Supervised Learning noch zwischen *Regression* und *Klassifikation*.

In dieser Arbeit kommt ausschließlich die Methode des Supervised Learning mit der Einschränkung auf Regressionsprobleme zum Einsatz. Die Eingabedaten $\boldsymbol{x}$ sind dabei i. Allg. vektorwertig, während es sich bei den Ausgaben y um reellwertige Größen handelt. Der Fall vektorwertiger Ausgaben kann auf den Fall skalarer Ausgaben zurückgeführt werden, indem für jede Komponente der Ausgabe ein skalarwertiges Modell konstruiert wird.

Die vorliegende Arbeit gliedert sich wie folgt: Im Kapitel 2 wird zunächst ein

Überblick über dynamische Systeme, einige ihrer Eigenschaften und die Messung und Rekonstruktion ihrer Dynamik aus Zeitreihen gegeben.

Im breiter angelegten Kapitel 3 wird zunächst dargestellt, wie sich das Problem der Zeitreihenvorhersage in den Kontext des Supervised Learning einordnet. Anschließend werden die Grundlagen der statistischen Regressionstheorie ausführlich behandelt und wichtige Aspekte wie die Fähigkeit eines Modells zur Generalisierung, Bias und Varianz des Modells und deren Abhängigkeiten untereinander erläutert. Die Fähigkeit eines Modells zur Generalisierung bezeichnet dessen Vermögen, auch für andere Daten desselben Systems als die zur Konstruktion verwendeten Lernbeispiele die zugehörigen Ausgaben ähnlich genau approximieren zu können wie die zum Training verwendeten Daten. Damit im Zusammenhang steht die Problematik des Over- und Underfitting. Beim Auftreten von Overfitting wurde das Modell zu genau an die Trainingsdaten angepasst, so dass das Modell nicht nur die den Daten zugrundeliegende Dynamik abbildet, sondern auch Eigenschaften der konkreten Realisierung der Trainingsdaten wie ein überlagertes Rauschsignal. Die Anwendung des Modells zur Vorhersage neuer, unabhängiger Daten, die wiederum eine eigene konkrete Realisierung der Ausgabe des Systems darstellen, führt dann im statistischen Mittel zu schlechteren Vorhersagen. Beim Underfitting tritt das Gegenteil ein: Das Modell ist nicht flexibel (d.h. komplex) genug, um die Dynamik des betrachteten Systems zu erfassen, und liefert im statistischen Mittel ebenfalls schlechte Vorhersagen. Diese Problematik wird ausführlich behandelt und Möglichkeiten zur Erkennung und Vermeidung von Over- und Underfitting werden diskutiert. Wichtige Konzepte in diesem Zusammenhang sind die Modellvalidierung, Regularisierung und Termselektion, auf die ausführlich eingegangen wird. Verschiedene Termselektionsalgorithmen werden verglichen und in ihren Auswirkungen einer nichtlinearen Optimierung der Modellparameter gegenübergestellt.

Das Kapitel 4 behandelt die Modellierung von Parameterabhängigkeiten dynamischer Systeme aus Zeitreihen. Zuerst werden die gängigen Ansätze zur Bewältigung dieser Aufgabe vorgestellt. Anschließend wird auf die hier besonders ausgeprägte Problematik eingegangen, die die Konstruktion von Modellen mit guter Übereinstimmung der Vorhersage der Langzeitdynamik mit den gemessenen Daten bei der freien Iteration des Modells mit sich bringt. Schließlich werden Methoden vorgestellt, die einen direkten quantitativen Vergleich der rekonstruierten Attraktoren der Dynamik erlauben und damit als Maß für die Übereinstimmung der Langzeitdynamik des frei iterierten Systems mit der wahren Dynamik ver-

wendet werden können. Dieses Maß kann dann zur automatisierten Konstruktion eines Ensembles von Modellen verwendet werden, das eine genauere Approximation der Parameterabhängigkeit und damit der Langzeitdynamik erlaubt. Die Anwendung der vorgestellten Methoden wird an einigen Beispielen numerisch generierter Zeitreihen chaotischer dynamischer Systeme demonstriert.

Kapitel 5 fasst die Ergebnisse zusammen, unterzieht die erzielten Ergebnisse einer kritischen Diskussion und gibt einen kleinen Ausblick auf daraus resultierende Fragestellungen.

Kapitel 2
Grundlagen

Bevor auf das eigentliche Modellierungsproblem eingegangen wird, sollen hier zunächst einige Grundlagen dynamischer Systeme erläutert werden, von denen einige später als zu modellierende Objekte dienen werden.

2.1 Dynamische Systeme

Dynamische Systeme sind ganz allgemein Systeme, die sich nach bestimmten Regeln zeitlich ändern. Der Zustand eines solchen Systems lässt sich formal als Vektor $\boldsymbol{x}$ in einem Vektorraum V beschreiben, der *Zustandsraum* genannt wird. Ist die zeitliche Entwicklung eines dynamischen Systems ausgehend von einem Anfangszustand eindeutig bestimmt, heißt das System *deterministisch*. Im kontinuierlichen Fall wird die Zeitentwicklung eines solchen Systems meist durch einen Satz gewöhnlicher Differentialgleichungen beschrieben:

$$\dot{\boldsymbol{x}} = \boldsymbol{F}(\boldsymbol{x})\,. \tag{2.1}$$

Diese Darstellung als autonomes Differentialgleichungssystem erster Ordnung ist insofern allgemeingültig, als sich auch jedes nichtautonome System höherer Ordnung durch die Einführung zusätzlicher Variablen in die Form (2.1) bringen lässt. Die zeitliche Entwicklung eines Zustandes $\boldsymbol{x}(t) \in V$ nennt man *Trajektorie* oder Bahnkurve oder auch *Orbit* von $\boldsymbol{x}$. Die Gesamtheit aller Trajektorien wird auch als Phasenbild des Systems bezeichnet.

Im zeitdiskreten Fall wird ein dynamisches System durch einen Satz von Differenzengleichungen beschrieben:

$$\boldsymbol{x}_{n+1} = \boldsymbol{F}(\boldsymbol{x}_n)\,. \tag{2.2}$$

Durch beide Fälle wird eine i. Allg. stetig differenzierbare Abbildung $\boldsymbol{\phi}$ definiert:

$$\begin{aligned} \boldsymbol{\phi} : \mathbb{R}^D \times \mathbb{K} &\to \mathbb{R}^D \\ (\boldsymbol{x}, t) &\mapsto \boldsymbol{\phi}(\boldsymbol{x}, t)\,. \end{aligned} \tag{2.3}$$

Diese besitzt die Eigenschaften

$$\begin{aligned} \boldsymbol{\phi}(\boldsymbol{x},0) &= \boldsymbol{x} \\ \boldsymbol{\phi}(\boldsymbol{\phi}(\boldsymbol{x}, u), t) &= \boldsymbol{\phi}(\boldsymbol{x}, u+t) \qquad \forall u, t \in \mathbb{K}\ \forall \boldsymbol{x} \in \mathbb{R}^D \end{aligned} \tag{2.4}$$

und beschreibt die zeitliche Entwicklung eines Zustandes $\boldsymbol{x}$. Für zeitkontinuierliche Systeme ist $\mathbb{K} = \mathbb{R}$ und der Fluss wird durch (2.1) erzeugt. Für diskrete Systeme hingegen ist $\mathbb{K} = \mathbb{Z}$, wobei der Fluss in diesem Fall durch die Abbildung (2.2) selbst gegeben ist. Da die Trajektorie eines Zustandes durch einen Zustandsvektor bereits eindeutig definiert ist, können sich Trajektorien im Zustandsraum nicht schneiden. Anders ausgedrückt bedeutet dies: Zwei Trajetorien, die einen Punkt gemeinsam haben, sind identisch.

Im Gegensatz zu konservativen Systemen, bei denen ein Volumen des Zustandsraums unter der zeitlichen Entwicklung konstant bleibt, schrumpft ein solches Volumen bei dissipativen Systemen, was gleichbedeutend mit $\operatorname{div} \boldsymbol{F} < 0$ ist. Typisch für dissipative Systeme ist, dass ein solches Volumen im asymptotischen Verhalten auf eine kompakte Untermenge $A \subset V$ zustrebt, die aufgrund ihres anziehenden Verhaltens als *Attraktor* bezeichnet wird. Ein Attraktor A hat die Eigenschaften [3]

- Attraktivität: Es gibt eine offene Umgebung U von A ($A \subset U$), so dass $\boldsymbol{\phi}(U, t) \subset U$ für $t > 0$ und die sich unter der Wirkung von $\boldsymbol{\phi}$ auf A zusammenzieht, d.h.
$$A = \bigcap_{t>0} \boldsymbol{\phi}(U, t)\,. \tag{2.5}$$
- Invarianz: Aus $\boldsymbol{x} \in A$ folgt auch $\boldsymbol{\phi}(\boldsymbol{x}, t) \in A\ \forall t$, d.h. der Attraktor A ist invariant unter der Wirkung des Flusses.
- Nichtzerlegbarkeit: Mit wachsendem t und für fast alle[1] $\boldsymbol{x}_0$ gilt: $\boldsymbol{\phi}(\boldsymbol{x}_0, t) \in U_{\boldsymbol{a}}$ für beliebige Umgebungen $U_{\boldsymbol{a}}$ aller Attraktorpunkte $\boldsymbol{a} \in A$.

Die letzte Eigenschaft bedeutet, dass der Attraktor A nicht in zwei abgeschlossene, nichtüberlappende, invariante Mengen zerlegt werden kann. Die Menge aller

[1] d.h. alle bis auf eine Menge mit Lebesgue-Maß Null

Punkte des Zustandsraums, für die A anziehend wirkt, heißt *Bassin* oder Einzugsgebiet des Attraktors. Im Zustandsraum eines (dissipativen) dynamischen Systems können mehrere Attraktoren koexistieren. Weiterhin können sie durch Variation der Systemparameter entstehen oder vernichtet werden oder unattraktiv (repulsiv) werden, was auf die Theorie von Fixpunkten und Bifurkationen führt [4, 5].

2.2 Messung und Rekonstruktion des Attraktors

Für die Modellierung eines dynamischen Systems spielen die Messung von Zuständen und die Rekonstruktion von Attraktoren eine wichtige Rolle. Formal lässt sich eine Messung durch eine Abbildung $\boldsymbol{h} : V \to B \subseteq \mathbb{R}^b,\ \boldsymbol{x}_t \mapsto \boldsymbol{s}_t = \boldsymbol{h}(\boldsymbol{x}_t)$ vom Zustandsraum V in einen *Beobachtungsraum* B beschreiben. Die gemessenen Werte $\{\boldsymbol{s}_t | t = 1, \ldots, N\}$ bilden eine *Zeitreihe*, die man für $b = 1$ als *skalare* und für $b > 1$ als *multivariate* Zeitreihe bezeichnet. Die Messung erfolgt dabei gewöhnlich in festen Zeitabständen $\Delta t = 1/f_s$ mit der *Abtastfrequenz* f_s.

Zwar wird die zeitliche Entwicklung eines kontinuierlichen Systems durch (2.1) vollständig beschrieben und kann durch die Bestimmung eines Anfangszustands zu einem bestimmten Zeitpunkt – also durch Festlegung eines Punktes im Zustandsraum – prinzipiell für alle Zeiten vorhergesagt werden, jedoch ist das die Dynamik beschreibende Differentialgleichungssystem (2.1) oft unbekannt bzw. so kompliziert, dass es sich nicht auf „Laborbedingungen“ reduzieren lässt, oder der Zustandsraum ist nicht vollständig für Messungen zugänglich. In diesem Fall ist die Dimension b des Beobachtungsraums B kleiner als die Dimension des Zustandsraums, d.h. es tritt bei der Messung ein Informationsverlust auf, der (auch bei hinreichend großer Abtastfrequenz) zu einem Verlust der Eindeutigkeit der Trajektorien im Beobachtungsraum führen kann. Oft ist sogar $b = 1$, es liegt also nur eine skalare Zeitreihe vor. Man könnte in diesem Fall z.B. versuchen, mit Hilfe aufeinanderfolgender Samples (Abtastwerte) der Zeitreihe durch Bildung von Differenzenquotienten die Ableitungen zu approximieren. Dieses Vorgehen ist jedoch ungenau und anfällig gegen Rauschen. Vielversprechender ist die Anwendung der sog. *Delay-Rekonstruktion*. Diese stellt eine Anwendung des Einbettungstheorems von Takens [6] bzw. dessen Verallgemeinerung durch Sauer et al. [7] dar, das Aussagen darüber liefert, unter welchen Voraussetzungen der aus

einer skalaren Zeitreihe rekonstruierte Attraktor diffeomorph[2] (d.h. topologisch äquivalent) zum Original-Attraktor im Zustandsraum ist. Unter einem solchen Diffeomorphismus bleiben die geometrischen Invarianten der Dynamik wie die Attraktordimension und die positiven Lyapunov-Exponenten erhalten, was eine Grundvoraussetzung für eine erfolgversprechende Modellierung der Dynamik im rekonstruierten Zustandsraum ist. Eine Abbildung aus dem Beobachtungs- in den Rekonstruktionsraum mit diesen Eigenschaften wird *Einbettung* genannt.

Dazu sei das dynamische System durch den Fluss $\boldsymbol{\phi}$ auf der offenen Menge $M \subset \mathbb{R}^D$ beschrieben und die Dynamik verlaufe auf einem Attraktor $A \subset M$ mit der Kapazitätsdimension d. Es sei $\tau > 0$ und $h: \ M \to \mathbb{R}$ eine stetig differenzierbare Funktion. Dann ist die Delaykoordinaten-Abbildung $F(h, \boldsymbol{\phi}, \tau): \ M \to \mathbb{R}^n$ definiert durch

$$F(h, \boldsymbol{\phi}, \tau)(\boldsymbol{x}) = (h(\boldsymbol{x}), h(\boldsymbol{\phi}_{-\tau}(\boldsymbol{x})), \ldots, h(\boldsymbol{\phi}_{-(n-1)\tau}(\boldsymbol{x}))) \ . \tag{2.6}$$

Dabei ist τ die *Delay*-Zeit, die natürlich ein Vielfaches von Δt ist. Die entscheidende Aussage des Einbettungstheorems ist nun folgende: Unter den Voraussetzungen, dass A nur endlich viele Gleichgewichtspunkte und keine periodischen Orbits der Periode τ und 2τ und höchstens endlich viele periodische Orbits der Perioden 3τ,$4\tau, \ldots, n\tau$ in $\boldsymbol{\phi}$ enthält und dass die Linearisierung des Flusses entlang dieser periodischen Orbits verschiedene Eigenwerte ergibt, ist die Delaykoordinaten-Abbildung (2.6) für fast jede stetig differenzierbare Funktion $h: \ M \to \mathbb{R}$ eine Einbettung, falls $n > 2d$ gewählt wird[3].

In der Praxis gewinnt man durch Messungen einer skalaren Größe $s_t = h(\boldsymbol{x}_t)$ am System eine skalare Zeitreihe $\{s_t | t = 1, \ldots, N\}$ und konstruiert hieraus die Delayvektoren

$$\boldsymbol{x}_t = (s_t, s_{t-\tau}, \ldots, s_{t-(D-1)\tau}) \in \mathbb{R}^D \tag{2.7}$$

mit der Einbettungsdimension D und dem Delay $\tau = k \cdot \Delta t$ für ein $k \in \mathbb{N}$.

Streng genommen gilt das Einbettungstheorem nur für rauschfreie Zeitreihen. Diese Voraussetzung ist in der Praxis für experimentell gewonnene Daten allerdings nicht zu erfüllen. Weiterhin mag zwar für fast jede Zeitverzögerung $\tau > 0$ eine topologische Äquivalenz zwischen originalem und rekonstruiertem Attraktor existieren, jedoch ist dies im mathematischen Sinne zu sehen. In der Praxis sind

[2] Ein Diffeomorphismus ist eine bijektive, stetig differenzierbare Abbildung, bei der auch die Umkehrabbildung stetig differenzierbar ist.

[3] „fast jede“ bedeutet hier „mit Wahrscheinlichkeit Eins“.

für eine sehr kleine Zeitverzögerung[4] die Samples s_t und $s_{t-\tau}$ nahezu identisch, so dass die Delayvektoren praktisch auf der Raumdiagonalen liegen, was Probleme aufgrund der endlichen Rechengenauigkeit der Computer nach sich zieht und eine Modellierung oder Berechnung der Invarianten im Rekonstruktionsraum unmöglich macht. Eine deutlich zu groß gewählte Delay-Zeit hingegen führt bei chaotischer Dynamik aufgrund der sensitiven Abhängigkeit von unvermeidlichen Messungenauigkeiten dazu, dass die beiden Samples ihre statistische Korreliertheit verlieren und die Delayvektoren scheinbar zufällig im Rekonstruktionsraum verteilt sind. Zur Wahl der Einbettungsdimension sagt das Theorem aus, dass man auf jeden Fall eine Einbettung erhält, wenn $D > 2d$ gewählt wird. Die Entfaltung eines Attraktors funktioniert allerdings oft auch schon bei deutlich niedrigeren Werten für D.

In der Praxis hängt der Erfolg einer Modellierung oder der Berechnung von Invarianten also durchaus von der Wahl von geeigneten Werten für den Delay und die Einbettungsdimension ab. Methoden zur Bestimmung geeigneter Einbettungsparameter finden sich z.B. in [8–10].

[4] sehr klein im Vergleich zur Zeitskala, auf der sich die Systemdynamik abspielt

Kapitel 3
Modellierung

3.1 Modellierung von Zeitreihen

Die Modellierung von Zeitreihen hat zum Ziel, aus aktuellen und vergangenen Werten der Zeitreihe zukünftige Werte vorherzusagen. Eine solche Zeitreihe wird als einem dynamischen System entstammend angenommen. Wie schon in Abschnitt 2.2 erläutert wurde, steht der Zustandsraum des dynamischen Systems häufig nicht (vollständig) für Messungen zur Verfügung, sondern man hat oft nur eine skalare Zeitreihe $\{s_1, \ldots, s_n\}$, aus der man durch Delay-Einbettung rekonstruierte Zustandsraumvektoren erzeugt. Im Kontext des Supervised Learning gestaltet sich das Ganze dann so, dass man aus der Zeitreihe einen Trainingsdatensatz durch Definition von

$$\begin{aligned} \boldsymbol{x}_t &= (s_t, s_{t-\tau}, \ldots, s_{t-(D-1)\tau}) \\ y_t &= s_{t+r} \end{aligned} \tag{3.1}$$

mit Eingabedaten $\boldsymbol{x}_t \in \mathbb{R}^D$ und zugehörigen Ausgaben $y_t \in \mathbb{R}$ konstruiert. Dabei ist D die Dimension des rekonstruierten Zustandsraums, τ der Delay (in Vielfachen der Abtastperiode Δt) und $r \in \mathbb{N}$ die Vorhersageschrittweite (ebenfalls in Vielfachen von Δt). Die Modellierung besteht damit aus der Approximation des Flusses $\boldsymbol{\phi}(\boldsymbol{x}_t, r\Delta t)$ im rekonstruierten Zustandsraum. Der funktionelle Zusammenhang zwischen den $\boldsymbol{x}_t$ und y_t hängt dabei i. Allg. natürlich auch von der Länge r der direkten Vorhersageschrittweite ab und wird bei nichtlinearen Systemen mit wachsendem r immer komplizierter. Das bedeutet, dass ein solches Modell immer nur diese feste Anzahl r von Schritten direkt in die Zukunft vorhersagen kann, für andere Schrittweiten i. Allg. aber keine sinnvollen Ergebnisse liefert. Ist man aber am zukünftigen Verlauf der Zeitreihe über mehr als einen zukünftigen

Schritt interessiert, so kann man dieses Problem mittels der iterierten Vorhersage lösen, bei der Ausgaben des Modells in zukünftige Eingaben mit einfließen. Das Modell $g(\boldsymbol{x})$ wird für eine feste direkte Vorhersageschrittweite trainiert (oft wird dabei $r = 1$ gewählt) und anschließend frei iteriert. Beginnt man mit dem zeitlich neuesten Delayvektor $\boldsymbol{x}_t$ und bezeichnet die Vorhersage des unbekannten nächsten Samples s_{t+1} mit $\hat{s}_{t+1} = g(\boldsymbol{x}_t)$, so kann man aus dieser eine Approximation $\hat{\boldsymbol{x}}_{t+1}$ des nächsten rekonstruierten Zustands $\boldsymbol{x}_{t+1}$ konstruieren mit

$$\hat{\boldsymbol{x}}_{t+1} = (\hat{s}_{t+1}, s_{t+1-\tau}, \ldots, s_{t+1-(D-1)\tau}) , \tag{3.2}$$

dessen Modellausgabe wiederum zur Konstruktion von $\hat{\boldsymbol{x}}_{t+2}$ verwendet werden kann usw. Allerdings akkumulieren sich die Einschritt-Vorhersagefehler bei der iterierten Vorhersage, so dass schon der nächste Eingabevektor $\hat{\boldsymbol{x}}_{t+1}$ i. Allg. keinen gültigen Systemzustand mehr darstellt. Bei chaotischen dynamischen Systemen kommt aufgrund von positiven Lyapunov-Exponenten noch die exponentielle Verstärkung von kleinsten Abweichungen hinzu, so dass der Prädiktionshorizont auch bei einem perfekten Modell prinzipiell begrenzt ist.

3.2 Statistische Aspekte der Modellierung

Das Modellierungsproblem besteht in der Aufgabe, aus einem Satz von Trainingsdaten $\mathcal{D} = \{(\boldsymbol{x}_t, y_t) | t = 1, \ldots, N\}$, denen ein funktioneller Zusammenhang $f : \mathbb{R}^D \to \mathbb{R},\ \boldsymbol{x}_t \mapsto y_t$ unterstellt wird, eine Approximation $g : \mathbb{R}^D \to \mathbb{R},\ \boldsymbol{x}_t \mapsto \hat{y}_t$ für die unbekannte Abbildung f zu konstruieren. Die Annahme eines rein deterministischen Zusammenhangs zwischen den Ein- und Ausgaben ist jedoch unrealistisch, u.a. wegen unvermeidbarem Messrauschen und wegen evtl. existierender Einflüsse aus weiteren, nicht messbaren Variablen auf die Ausgaben y. Stattdessen lassen sich die Paare $(\boldsymbol{x}_t, y_t)$ als unabhängige Realisationen von Zufallsvariablen X bzw. Y auffassen, die über eine (unbekannte) bedingte Wahrscheinlichkeitsverteilung $P(\mathrm{Y}|\mathrm{X})$ miteinander verknüpft sind[1]. Vor der Konstruktion eines Modells $g(\boldsymbol{x})$, das die Ausgabe $\hat{y} = g(\boldsymbol{x})$ für eine Eingabe $\boldsymbol{x}$ liefert, muss man sich zuerst für einen (geeigneten) Modellierungsansatz entscheiden, der die prinzipielle Architektur festlegt, innerhalb der das Modell dann durch einen Satz von Parametern spezifiziert wird. Bei einem lokalen Modell sind dies z.B. die Anzahl

[1] Diese allgemeine Formulierung schließt den rein deterministischen Fall mit ein, in dem die Ausgabe y zu einer Eingabe $\boldsymbol{x}$ exakt bestimmt ist.

der nächsten Nachbarn und der Grad der lokal angefitteten Polynome, bei einem global polynomialen Modell z.B. der maximale Grad des Polynoms. Der Lernvorgang besteht dann in der Anpassung dieser Parameter. Dazu wird ein Maß für die Größe der Abweichung zwischen der Modellausgabe $\hat{y}$ und dem gemessenen Wert y benötigt, das in Form einer *Loss*-Funktion $L(y, \hat{y})$ definiert wird. Die weitaus am häufigsten verwendete Loss-Funktion ist das Quadrat der Abweichung:

$$L(y, \hat{y}) = (y - \hat{y})^2 \,. \tag{3.3}$$

Die zugehörige Schätzmethode für die Parameter trägt den Namen *Methode der kleinsten Quadrate*, im Englischen mit *least squares* bezeichnet, bei der die Modellparameter so gewählt werden, dass sie den mittleren quadratischen Fehler (*mean of squared errors*, MSE)

$$\mathrm{MSE} = \frac{1}{N} \sum_{t=1}^{N} (y_t - g(\boldsymbol{x}_t))^2 \tag{3.4}$$

minimieren, der dem Mittelwert des Loss über alle Trainingsdaten entspricht.

Die bedingte Wahrscheinlichkeitsverteilung $P(\mathrm{Y}|\mathrm{X})$ lässt sich überall dort, wo $P(\mathrm{X}) > 0$ gilt, schreiben als

$$P(\mathrm{Y}|\mathrm{X}) = P(\mathrm{X}, \mathrm{Y})/P(\mathrm{X}) \,. \tag{3.5}$$

Sie kann auf komplizierte Art und Weise von X abhängen. Eine gebräuchliche und oft sinnvolle Approximation des realen Zusammenhangs ist das additive Modell

$$\mathrm{Y}_t = f(\mathrm{X}_t) + \epsilon_t \,, \tag{3.6}$$

bei dem $P(\mathrm{Y}|\mathrm{X})$ rein deterministisch von X abhängt und sämtliche zusätzliche Effekte, die Y beeinflussen wie Messrauschen und nicht messbare Variablen, in einer additiven Zufallsvariablen ϵ zusammengefasst sind, die unabhängig von X ist. So kann jedes Trainingsdatenpaar $(\boldsymbol{x}_t, y_t)$ als Realisation der entsprechenden Zufallsvariablen X_t bzw. Y_t aufgefasst werden, die über (3.6) miteinander verknüpft sind, wobei angenommen wird, dass die Rauschvariablen mittelwertfrei sowie unkorreliert sind und gleiche Varianz σ_ϵ^2 haben: $\mathrm{E}[\epsilon_t] = 0$, $\mathrm{Cov}\big[\epsilon_s, \epsilon_t\big] = \delta_{st}\sigma_\epsilon^2 \;\forall s, t = 1, \ldots, N$. Einen solchen stochastischen Prozess nennt

man auch *Weißes Rauschen* und schreibt

$$\{\epsilon_t\} \sim \mathrm{WN}(0, \sigma_\epsilon^2)\,, \quad t = 1, \ldots, N\,. \tag{3.7}$$

Für ein (beliebiges) Modell $g(\boldsymbol{x})$ ist der Erwartungswert des quadratischen Fehlers

$$\begin{aligned}
\mathrm{E}[(y - g(\boldsymbol{x}))^2|\boldsymbol{x}] &= \mathrm{E}\big[\big((y - \mathrm{E}[y|\boldsymbol{x}]) + (\mathrm{E}[y|\boldsymbol{x}] - g(\boldsymbol{x}))\big)^2 \;\big|\; \boldsymbol{x}\big] \\
&= \mathrm{E}\big[\big(y - \mathrm{E}[y|\boldsymbol{x}]\big)^2 \;\big|\; \boldsymbol{x}\big] + \big(\mathrm{E}[y|\boldsymbol{x}] - g(\boldsymbol{x})\big)^2 \\
&\quad + 2 \cdot \mathrm{E}\big[y - \mathrm{E}[y|\boldsymbol{x}] \;\big|\; \boldsymbol{x}\big] \cdot \big(\mathrm{E}[y|\boldsymbol{x}] - g(\boldsymbol{x})\big) \\
&= \mathrm{E}\big[\big(y - \mathrm{E}[y|\boldsymbol{x}]\big)^2 \;\big|\; \boldsymbol{x}\big] + \big(\mathrm{E}[y|\boldsymbol{x}] - g(\boldsymbol{x})\big)^2 \\
&\quad + 2(\mathrm{E}[y|\boldsymbol{x}] - \mathrm{E}[y|\boldsymbol{x}]) \cdot \big(\mathrm{E}[y|\boldsymbol{x}] - g(\boldsymbol{x})\big) \\
&= \mathrm{E}\big[\big(y - \mathrm{E}[y|\boldsymbol{x}]\big)^2 \;\big|\; \boldsymbol{x}\big] + \big(\mathrm{E}[y|\boldsymbol{x}] - g(\boldsymbol{x})\big)^2 \\
&\geq \mathrm{E}\big[\big(y - \mathrm{E}[y|\boldsymbol{x}]\big)^2 \;\big|\; \boldsymbol{x}\big] = \mathrm{E}\big[\big(y - f(\boldsymbol{x})\big)^2 \;\big|\; \boldsymbol{x}\big]\,,
\end{aligned} \tag{3.8}$$

da $\mathrm{E}[\mathrm{E}[y|\boldsymbol{x}]|\boldsymbol{x}] = \mathrm{E}[y|\boldsymbol{x}]$ und $\mathrm{E}[g(\boldsymbol{x})|\boldsymbol{x}] = g(\boldsymbol{x})$. Die bestmögliche Voraussage für eine Eingabe $\boldsymbol{x}$ im Sinne einer Minimierung des quadratischen Fehlers ist also der bedingte Erwartungswert $\mathrm{E}[\mathrm{Y}|\mathrm{X} = \boldsymbol{x}]$, der auch als *Regressionsfunktion* bezeichnet wird und für das Modell mit additivem Fehler (3.3) gerade mit dem deterministischen Anteil der Abbildung übereinstimmt: $f(\boldsymbol{x}) = \mathrm{E}[\mathrm{Y}|\mathrm{X} = \boldsymbol{x}]$. Das Ziel der Modellierung besteht also in einer möglichst guten Schätzung der Regression anhand der Trainingsdatenmenge $\mathcal{D}$. Wie sich noch zeigen wird, besteht das Problem dabei in der Tatsache, dass jede Trainingsmenge prinzipiell nur *endlich* viele Realisationen von $P(\mathrm{Y}|\mathrm{X})$ enthält, was unweigerlich zu Fehlern bei der Schätzung von $\mathrm{E}[\mathrm{Y}|\mathrm{X} = \boldsymbol{x}]$ führt.

3.2.1 Die Wahl der Modellarchitektur

Die möglichen Ansätze zur Konstruktion eines Modells lassen sich einteilen in parametrische, semiparametrische und nichtparametrische Methoden. In diesen Bezeichnungen drückt sich der Grad der (angenommenen) Kenntnis der Form der bedingten Wahrscheinlichkeitsverteilung aus und damit der Grad der Einschränkung auf einen bestimmten Modelltypus. Bei der parametrischen Regression nimmt man eine meist bis auf relativ wenige Parameter bekannte bestimmte Form der bedingten Wahrscheinlichkeitsverteilung an, z.B. das lineare Modell $g(\boldsymbol{x}) = a_0 + \boldsymbol{a}_1^{\mathrm{T}}\boldsymbol{x}$ oder das polynomiale Modell $g(\boldsymbol{x}) = p_M(\boldsymbol{x})$ mit vorgegebenem

Grad M. Das Training des Modells entspricht dann der Schätzung der unbekannten Parameter. Der Vorteil des parametrischen Ansatzes liegt darin, dass relativ wenige Trainingsdaten zur Bestimmung der Parameter nötig sind, falls der gewählte Ansatz richtig ist. Der Nachteil ist, dass man die Wahrscheinlichkeitsverteilung der Daten im Prinzip kennen muss, denn bei einem falsch gewählten Ansatz ist keine erfolgversprechende Modellierung möglich. Nichtparametrische Methoden setzen keine bestimmte funktionale Form voraus, sondern konstruieren die Abbildung „aus den Daten“ . Das bedeutet, dass sowohl die letztlich gewählte Architektur als auch die innerhalb dieser Architektur auftretenden freien Parameter anhand der Trainingsdaten gewählt werden. Beispiele hierfür sind z.B. nächste-Nachbar-basierte lokale Modelle [11,12], mehrschichtige Neuronale Netze (Multi-Layer Perceptrons, [13]) oder auf Kernfunktionen basierende Methoden. Nichtparametrische Methoden benötigen i. Allg. sehr viel mehr Trainingsdaten zur Schätzung ihrer Parameter, da aus den Daten letztlich sowohl die funktionale Form als auch die zugehörigen Parameter geschätzt werden müssen.

Der in dieser Arbeit gewählte Ansatz entspricht der semiparametrischen Regression und verwendet eine Entwicklung des Modells in (nichtlineare) *Basisfunktionen*

$$g(\boldsymbol{x}) = \sum_{k=1}^{M} w_k g_k(\boldsymbol{x})\,. \tag{3.9}$$

Diese Architektur ist eine nichtlineare Verallgemeinerung des linearen Modells. Festgelegt ist hier zwar der Typ der verwendeten Basisfunktionen $g_k(\boldsymbol{x})$, allerdings ist ihre Anzahl M variabel, was diesen Methoden ihre große Flexibilität verleiht. Speziell bei der nichtlinearen Zeitreihenmodellierung, bei der die durch Delay-Rekonstruktion erzeugten Trainingsdaten nach (3.1) zeitlich geordnet sind und voneinander abhängen, werden oft NARMAX-Modelle verwendet[2], die sich in der Form (3.9) darstellen lassen und eine nichtlineare Erweiterung der weit verbreiteten ARMAX-Modelle darstellen [14,15]. In der Literatur ist die Verwendung zahlreicher Basisfunktionentypen dokumentiert, darunter global-polynomiale Basisfunktionen [14–23], Spline-Funktionen [24], rationale Basisfunktionen [25, 26], radiale Basisfunktionen [27–32], Wavelets [33,34] oder sigmoidale Funktionen der Form $g_k(\boldsymbol{x}) = \sigma(\boldsymbol{\alpha}_k^{\mathrm{T}}\boldsymbol{x} + b_k)$ mit $\sigma(x) = 1/(1 + e^{-x})$ (Single-Hidden-Layer Feed-Forward Neural Network oder Two-Layer Perceptron, [24]).

Die Verwendung nichtlinearer Funktionen in (3.9) kann aufgefasst werden als

[2] NARMAX steht für **N**onlinear **A**uto**R**egressive **M**oving **A**verage with e**X**ogenous inputs.

Transformation der Eingaben in einen *Merkmalsraum* (engl. *feature space*), in dem das Regressionsproblem mit linearen Methoden gelöst wird. Der Vorteil dieses Ansatzes ist, dass sich für fest definierte Basisfunktionen das Problem auf die Schätzung der linear in die Modellausgabe eingehenden Parameter w_k beschränkt, so dass der zu minimierende MSE eine konvexe Funktion dieser Parameter mit einem eindeutigen, globalen Minimum ist, das mit Standardverfahren der linearen Algebra bestimmt werden kann. Weiterhin lässt sich über die Anzahl M der Basisfunktionen direkt die Komplexität bzw. Flexibilität des Modells steuern.

In dieser Arbeit werden vorwiegend Gauß'sche radiale Basisfunktionen (RBF) der Gestalt

$$g_k(\boldsymbol{x}) = \exp\left(-\sum_{d=1}^{D} \frac{(x_d - c_{kd})^2}{r_{kd}^2}\right) , \tag{3.10}$$

verwendet, die durch einen Zentrenvektor $\boldsymbol{c}_k = (c_{k1}, \ldots, c_{kD})^{\mathrm{T}}$ und eine Breitenskalierung $\boldsymbol{r}_k = (r_{k1}, \ldots, r_{kD})^{\mathrm{T}}$ mit $r_{kd} > 0 \ \forall d$ parametrisiert werden[3]. Diese Art von Basisfunktionen hat einige Vorteile gegenüber den oft verwendeten polynomialen Funktionen. Letztere neigen bei freier Iteration des Modells, wie sie z.B. bei der Zeitreihenvorhersage angewendet wird, sehr leicht zu divergentem Verhalten. Der Grund dafür ist, dass bei nichtlinearen Systemen der Grad des Polynoms hinreichend hoch sein muss, um auch kompliziertere Zusammenhänge zwischen Ein- und Ausgaben abbilden zu können. Ein solches Polynom höheren Grades neigt dann aber an Orten im (rekonstruierten) Zustandsraum, in deren Nähe keine oder nur wenige Trainingsdaten liegen, zu wilden Oszillationen, die bei freier Iteration zur Divergenz führen können [15]. Bei den RBFs in (3.10) können solche Divergenzen nicht auftreten, da sie mit zunehmender Entfernung von ihrem Zentrum sehr stark abfallen und gegen Null konvergieren. Die Zentren selbst werden typischerweise den Trainingsdaten entnommen [35, 36] und folgen so der Verteilung der Eingabedaten. Die Breitenskalierungen werden so gewählt, dass eine gewisse Überlappung der RBFs im Eingaberaum gegeben ist, damit dort keine „Löcher" entstehen, also Orte, die durch keine der RBFs abgedeckt werden. Kecman [36] schlägt als Faustregel $r_{kd} \approx \Delta c_d$ vor, wobei Δc_d der Mittelwert der Abstände zwischen den d-ten Zentrenkoordinaten ist. Die einfachste Möglichkeit zur Modellkonstruktion besteht darin, aus den Trainingsdaten zufällig so viele als Zentren auszuwählen und den r_{kd} Werte in der Größenordnung von Δc_d zuzuweisen, bis die gewünschte Modellkomplexität erreicht ist. Fortschrittlichere

[3] Falls die r_{kd} nicht für alle $d = 1, \ldots, D$ identisch sind, ist (3.10) streng genommen nicht mehr radial. Allerdings gilt dies nur für die euklidische Norm, denn $g_k(\boldsymbol{x})$ ist radial bzgl. der Norm $\|\boldsymbol{x} - \boldsymbol{c}_k\|_S^2 \equiv (\boldsymbol{x} - \boldsymbol{c}_k)^{\mathrm{T}} \boldsymbol{S}^{\mathrm{T}} \boldsymbol{S} (\boldsymbol{x} - \boldsymbol{c}_k)$ mit $\boldsymbol{S} = \mathrm{diag}(1/r_{k1}, \ldots, 1/r_{kD})$.

Methoden zur Generierung geeigneter Basisfunktionen werden im Abschnitt 3.3 über Termselektionsalgorithmen untersucht.

Das Modell (3.9) mit den radialen Basisfunktionen (3.10) lässt sich auch als neuronales Netz auffassen, genauer als *Single-Hidden-Layer Feed-Forward Neural Network*, wobei die Aktivierungsfunktionen hier Gauß'sche RBFs sind [24]. Darum wird ein solches neuronales Netz auch *Radial Basis Function Neural Network* (RBFNN) genannt[4]. In diesem Kontext werden auch die Zentren und Breitenskalierungen der Basisfunktionen als Parameter des Netzwerks aufgefasst und sind Gegenstand der Optimierung. Diese kann mit gradientenbasierten Verfahren erfolgen. Allerdings ist der MSE keine konvexe Funktion bezüglich der Zentren und Breiten, was das Auffinden eines globalen Minimums sehr schwierig macht. Die nichtlineare Optimierung von RBF-Modellen wird in Abschnitt 3.5 behandelt.

Neben den Gauß'schen sind noch andere radiale Basisfunktionen gebräuchlich, wie z.B. *Multiquadrics*

$$g_k(\boldsymbol{x}) = \sqrt{\|\boldsymbol{x} - \boldsymbol{c}_k\|_2^2 + r_k^2} \tag{3.11}$$

oder *inverse Multiquadrics* [37, 38]

$$g_k(\boldsymbol{x}) = \left(\|\boldsymbol{x} - \boldsymbol{c}_k\|_2^2 + r_k^2\right)^{-1/2} , \tag{3.12}$$

wobei die Multiquadrics allerdings mit zunehmender Entfernung vom Zentrum ansteigen und daher wie die polynomialen Modelle bei freier Iteration zu divergentem Verhalten neigen.

Welcher Typ von Basisfunktionen am besten zur Modellierung geeignet ist, lässt sich nicht pauschal sagen, sondern hängt vom konkreten Problem ab, insbesondere also von der zu approximierenden Funktion f in (3.6).

Für Trainingsdaten $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ lassen sich die Modellausgaben $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_N)^{\mathrm{T}}$ mit $\hat{y}_t = g(\boldsymbol{x}_t)$ für $t = 1, \ldots, N$ einfach in Matrixform schreiben. Dazu definiert man die *Design-Matrix*

$$\boldsymbol{G} = \begin{pmatrix} g_1(\boldsymbol{x}_1) & \ldots & g_M(\boldsymbol{x}_1) \\ \vdots & & \vdots \\ g_1(\boldsymbol{x}_N) & \ldots & g_M(\boldsymbol{x}_N) \end{pmatrix} \in \mathbb{R}^{N \times M} \tag{3.13}$$

[4] Gebräuchlich ist auch die Abkürzung RBFN für *Radial Basis Function Network*.

und erhält damit die Modellausgaben

$$\hat{\boldsymbol{y}} = \boldsymbol{G}\boldsymbol{w}\,. \tag{3.14}$$

Die Berechnung dieses Koeffizientenvektors durch Minimierung des MSE bzw. äquivalent des SSE (*sums of squared errors*)

$$\mathrm{SSE} = \sum_{t=1}^{N}(y_t - g(\boldsymbol{x}_t))^2 = \|\boldsymbol{y} - \boldsymbol{G}\boldsymbol{w}\|_2^2 = N \cdot \mathrm{MSE} \tag{3.15}$$

ist damit auf ein gewöhnliches lineares Ausgleichsproblem zurückgeführt und kann durch Standardmethoden der linearen Algebra erfolgen. Eine Lösung

$$\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}\in\mathbb{R}^M} \|\boldsymbol{y} - \boldsymbol{G}\boldsymbol{w}\|_2^2 \tag{3.16}$$

dieses Problems existiert immer und genügt den Normalengleichungen

$$\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{w} = \boldsymbol{G}^{\mathrm{T}}\boldsymbol{y}\,. \tag{3.17}$$

Allerdings ist die Lösung nur dann eindeutig, wenn $\boldsymbol{G}$ maximalen Rang hat, denn nur dann existiert $(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G})^{-1}$ und $\boldsymbol{w}^\star$ lässt sich schreiben als

$$\boldsymbol{w}^\star = (\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{y}\,. \tag{3.18}$$

Für $\mathrm{rang}(\boldsymbol{G}) < \min(M, N)$ gibt es unendlich viele Lösungen, die im $\mathbb{R}^M$ alle auf einem affin-linearen Unterraum liegen. Unter allen diesen Lösungen ist jedoch wiederum diejenige eindeutig bestimmt, die den geringsten euklidischen Abstand zum Ursprung hat. Diese lässt sich mit Hilfe der Singulärwertzerlegung (SVD) angeben. Die SVD von $\boldsymbol{G}$ ist gegeben durch

$$\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} \in \mathbb{R}^{N\times M}\,, \tag{3.19}$$

wobei gilt $\boldsymbol{U} \in \mathbb{R}^{N\times N}$ und $\boldsymbol{V} \in \mathbb{R}^{M\times M}$ mit jeweils paarweise orthonormalen Spalten sowie $\Sigma = \mathrm{diag}\,(\sigma_1, \ldots, \sigma_{\min(M,N)}) \in \mathbb{R}^{N\times M}$. Dabei sind $\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_{\min(M,N)} = 0$ die Singulärwerte von $\boldsymbol{G}$, und $r \in \{1, \ldots, \min(M, N)\}$ ist der Rang von $\boldsymbol{G}$. Aus der Singulärwertzerlegung ergibt sich die Pseudo-Inverse $\boldsymbol{G}^\dagger$ von $\boldsymbol{G}$ zu

$$\boldsymbol{G}^\dagger = \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^{\mathrm{T}} \in \mathbb{R}^{M\times N}\,, \tag{3.20}$$

wobei $\boldsymbol{\Sigma}^\dagger \equiv \mathrm{diag}\,(1/\sigma_1,\ldots,1/\sigma_r,0,\ldots,0) \in \mathbb{R}^{M\times N}$ ist. Damit lässt sich die Lösung (3.16) des Minimierungsproblems einfach schreiben als [39]

$$\boldsymbol{w}^\star = \boldsymbol{G}^\dagger \boldsymbol{y} = \sum_{i=1}^{r} \frac{\boldsymbol{u}_i{}^{\mathrm{T}} \boldsymbol{y}}{\sigma_i} \boldsymbol{v}_i \,. \tag{3.21}$$

Falls $\boldsymbol{G}$ maximalen Rang besitzt ($r = \min(M,N)$), gilt $\boldsymbol{G}^\dagger = (\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathrm{T}}$, und (3.21) ist die eindeutige Lösung des Minimierungsproblems, andernfalls ($r < \min(M,N)$) ist (3.21) unter allen möglichen Lösungen die eindeutig bestimmte Lösung mit minimaler euklidischer Norm $\|\boldsymbol{w}\|_2$.

Um eventuell vorhandene konstante und lineare Anteile des i. Allg. nichtlinearen Zusammenhangs zwischen gemessenen Daten einfacher modellieren zu können, wird den in dieser Arbeit verwendeten RBF-Modellen noch eine Konstante sowie ein linearer Term hinzugefügt, so dass das Modell letztlich die Gestalt

$$\begin{aligned} g(\boldsymbol{x}) &= w_0 + \sum_{d=1}^{D} w_d x_d + \sum_{k=D+1}^{D+M} w_k g_k(\boldsymbol{x}) \\ &= w_0 + \sum_{d=1}^{D} w_d x_d + \sum_{k=D+1}^{D+M} w_k \exp\left(-\sum_{d=1}^{D} \frac{(x_d - c_{kd})^2}{r_{kd}^2}\right) \end{aligned} \tag{3.22}$$

hat. Auch in diesem Fall lässt sich die Modellausgabe als Produkt einer Design-Matrix mit einem Koeffizientenvektor schreiben, indem man

$$\boldsymbol{G} = \begin{pmatrix} 1 & x_{11} & \ldots & x_{1D} & g_1(\boldsymbol{x}_1) & \ldots & g_M(\boldsymbol{x}_1) \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 1 & x_{N1} & \ldots & x_{ND} & g_1(\boldsymbol{x}_N) & \ldots & g_M(\boldsymbol{x}_N) \end{pmatrix} \in \mathbb{R}^{N\times(M+D+1)} \tag{3.23}$$

setzt und mit dem Koeffizientenvektor $\boldsymbol{w} \in \mathbb{R}^{M+D+1}$ die Modellausgaben zu $\hat{\boldsymbol{y}} = \boldsymbol{G}\boldsymbol{w}$ erhält. Hierbei enthält $\boldsymbol{w}$ nun die Koeffizienten *aller* Terme (konstant, linear und RBF). Für die folgenden Diskussionen spielt lediglich die Linearität des Modells in den Koeffizienten der Basisfunktionen eine Rolle, nicht jedoch, ob diese Basisfunktionen ausschließlich aus RBF-Termen bestehen oder auch den konstanten oder linearen Term mit einschließen, so dass zur Vereinfachung immer die Notation aus (3.14) mit einer $(N \times M)$-Matrix $\boldsymbol{G}$ und einem Koeffizientenvektor $\boldsymbol{w} \in \mathbb{R}^M$ verwendet wird, wenn nicht anders angegeben.

3.2.2 Modellkomplexität, Bias und Varianz

In diesem Abschnitt soll näher auf die Probleme eingegangen werden, die die Schätzung der Modellparameter auf einer nur endlich langen Trainingsdatenmenge mit sich bringt. Dazu sei wieder ein Trainingsdatensatz $\mathcal{D} = \{(\boldsymbol{x}_t, y_t)|t = 1, \ldots, N\}$ der Länge N als Realisation von Zufallsvariablen gegeben, zwischen denen die Beziehung (3.6) besteht. Das Ziel ist die Konstruktion eines Modells $g(\boldsymbol{x})$, das die deterministische Komponente $f(\boldsymbol{x})$ des i. Allg. nichtlinearen Zusammenhangs zwischen den Ein- und Ausgaben approximiert. Die Konstruktion des Modells erfolgt durch Minimierung des mittleren quadratischen Fehlers (3.4). Das bedeutet aber nun nicht, dass das Modell die Trainingsdaten perfekt beschreiben (d.h. interpolieren) können soll. Vielmehr soll es *beliebige* Realisationen der bedingten Wahrscheinlichkeitsverteilung $P(\mathrm{Y}|\mathrm{X})$ möglichst gut beschreiben und damit insbesondere auch Daten, die keinen Eingang ins Training gefunden haben, d.h. es soll die Fähigkeit zur *Generalisierung* besitzen. Im Falle von verrauschten Trainingsdaten hat aber selbst das perfekte Modell $g(\boldsymbol{x}) \equiv f(\boldsymbol{x})$ noch einen endlichen, nicht verschwindenden Vorhersagefehler, denn für eine beliebige Eingabe $\boldsymbol{x}$ gilt

$$\mathrm{E}[(y - f(\boldsymbol{x}))^2|\boldsymbol{x}] = \mathrm{E}[\epsilon^2|\boldsymbol{x}] = \sigma_\epsilon^2 \tag{3.24}$$

In der Praxis stehen nur endlich viele Trainingsdaten zur Verfügung. Falls diese keine Wiederholungen enthalten (d.h. mehr als einen y-Wert zur gleichen Eingabe $\boldsymbol{x}$), ist es natürlich möglich, durch Konstruktion eines hinreichend flexiblen Modells den MSE auf den Trainingsdaten beliebig klein zu machen oder sogar ganz zum Verschwinden zu bringen, z.B. indem man die Anzahl M der Basisfunktionen in (3.9) genauso groß macht wie die Anzahl N der Trainingsdaten. Ein solches Modell würde dann nicht nur den deterministischen Teil f in (3.6) abbilden, sondern auch das stochastische Signal ϵ und würde darum auf einem zweiten, unabhängigen Datensatz desselben Systems einen deutlich größeren Vorhersagefehler liefern. In diesem Fall ist das Modell überangepasst an die Trainingsdaten, was als *Overfitting* bezeichnet wird. Es existiert auch der gegenteilige Fall: Ist die Flexibilität des Modells zu gering, um auch nur den deterministischen Anteil f zu beschreiben, so wird es sowohl auf den Trainingsdaten wie auch auf einem unabhängigen Testdatensatz einen großen Vorhersagefehler liefern. Dies bezeichnet man als *Underfitting*; der systematische Fehler des Modells aufgrund der zu geringen Komplexität wird als *Bias* bezeichnet. Die Aussagen über Over- und Underfitting gelten natürlich nur im statistischen Mittel. Ein überangepasstes Modell kann durchaus auch einen zweiten Datensatz gut beschreiben, im Mit-

tel über (sehr) viele unabhängige Datensätze wird seine Performance allerdings signifikant schlechter sein als auf den Trainingsdaten.

Das Modell muss also einerseits flexibel genug sein, um auch komplizierte Zusammenhänge zwischen den Daten abbilden zu können, andererseits darf die Flexibilität (in der statistischen Lerntheorie auch *Kapazität der Lernmaschine* genannt) nicht zu groß sein, da sonst auch die statistischen Eigenheiten dieser nur *endlich* großen Realisierung $\mathcal{D}$ des stochastischen Prozesses (3.6) mitmodelliert würden. Hierzu verwendet man einen zweiten, unabhängigen Datensatz des gleichen Systems, der im Folgenden *Testdatensatz* genannt wird. Der mittlere quadratische Fehler des Modells auf diesen Testdaten wird *Generalisierungsfehler* genannt [24] und ist ein Maß für die Verallgemeinerungsfähigkeit des Modells. Der Testdatensatz kann durch eine erneute Messung am System gewonnen werden oder durch Aufteilung der Messdaten in eine Trainings- und eine Testmenge. Entscheidend ist dabei, dass Informationen der Testdaten in keiner Weise Eingang in die Modellkonstruktion finden dürfen. Die Verwendung eines zweiten, unabhängigen Datensatzes zur Validierung wird *Cross-Validation* genannt (siehe auch Abschnitt 3.2.3). Ein Modell mit guten Generalisierungseigenschaften wird auf dem Testdatensatz einen vergleichbaren MSE liefern wie auf den Trainingsdaten. Beim Overfitting hingegen wird der Generalisierungsfehler deutlich über dem Fehler auf den Trainingsdaten liegen. Beim Underfitting ergibt sich sowohl ein großer Trainings- als auch ein großer Testfehler.

Ein Modell kann also aus zwei Gründen schlecht geeignet sein, den durch die unbekannte Funktion f gegebenen Zusammenhang zwischen Ein- und Ausgabedaten zu beschreiben: Es kann einen großen Bias aufweisen oder es leidet an Overfitting. Um den Grund für seine schlechte Performance aufzudecken, lohnt sich eine genauere Betrachtung des Modellierungsfehlers in (3.8). Um die Abhängigkeit der geschätzten Modellparameter von der konkreten Realisierung der Trainingsdaten $\mathcal{D}$ deutlich zu machen, wird das auf $\mathcal{D}$ trainierte Modell mit $g(\boldsymbol{x};\mathcal{D})$ bezeichnet. Weiterhin bezeichnet $\mathrm{E}_{\mathcal{D}}[\cdot]$ den Erwartungswert über alle möglichen Trainingsdatensätze der Länge N. Nach (3.8) und (3.24) ist der Erwartungswert des quadratischen Fehlers für ein gegebenes $\boldsymbol{x}$

$$\begin{aligned}\mathrm{E}[(y-g(\boldsymbol{x};\mathcal{D}))^2|\boldsymbol{x},\mathcal{D}] &= \mathrm{E}[(y-\mathrm{E}[y|\boldsymbol{x}])^2|\boldsymbol{x},\mathcal{D}] + \big(\mathrm{E}[y|\boldsymbol{x}]-g(\boldsymbol{x};\mathcal{D})\big)^2 \\ &= \sigma_\epsilon^2 + \big(\mathrm{E}[y|\boldsymbol{x}]-g(\boldsymbol{x};\mathcal{D})\big)^2 .\end{aligned} \tag{3.25}$$

Der erste Term ist unabhängig vom Modell und vom konkreten Trainingsdatensatz. Er entspricht der Varianz des additiven Rauschens und ist somit eine untere

Grenze für den Generalisierungsfehler. Ein Modell mit einem kleineren MSE als σ_ϵ^2 auf den Trainingsdaten wird also mit großer Wahrscheinlichkeit an Overfitting leiden. Interessanter ist der zweite Term, denn er stellt den eigentlichen Modellierungsfehler als quadratischen Abstand der Modellausgabe zur Regressionsfunktion dar. Ziel muss es sein, diesen Abstand möglichst klein zu machen, und zwar nicht nur für einen bestimmten Trainingsdatensatz $\mathcal{D}$ (das würde u. U. wieder zum Overfitting führen), sondern im Mittel für alle möglichen Trainingsdatensätze. So ist es zwar möglich, dass $g(\boldsymbol{x};\mathcal{D})$ für einen bestimmten Trainingsdatensatz eine sehr gute Approximation an die Regression $\mathrm{E}[y|\boldsymbol{x}]$ ist, auf einem anderen Datensatz aber deutlich schlechtere Ergebnisse liefert, d.h. stark schwankt, oder aber im Mittel über alle $\mathcal{D}$ die Regression nur schlecht approximiert. Beide Fälle führen zu schlechten Modellen. Um die Ursache hierfür aufzudecken, bildet man den Erwartungswert $\mathrm{E}_\mathcal{D}[(\mathrm{E}[y|\boldsymbol{x}] - g(\boldsymbol{x};\mathcal{D}))^2]$ des Modellierungsfehlers über alle möglichen $\mathcal{D}$ und zerlegt diesen wie in (3.8):

$$
\begin{aligned}
&\mathrm{E}_\mathcal{D}[(\mathrm{E}[y|\boldsymbol{x}] - g(\boldsymbol{x};\mathcal{D}))^2] \\
&\quad= \mathrm{E}_\mathcal{D}\big[\big(\mathrm{E}[y|\boldsymbol{x}] - \mathrm{E}_\mathcal{D}[g(\boldsymbol{x};\mathcal{D})] + \mathrm{E}_\mathcal{D}[g(\boldsymbol{x};\mathcal{D})] - g(\boldsymbol{x};\mathcal{D})\big)^2\big] \\
&\quad= \mathrm{E}_\mathcal{D}\big[\big(\mathrm{E}[y|\boldsymbol{x}] - \mathrm{E}_\mathcal{D}[g(\boldsymbol{x};\mathcal{D})]\big)^2\big] + \mathrm{E}_\mathcal{D}\big[\big(\mathrm{E}_\mathcal{D}[g(\boldsymbol{x};\mathcal{D})] - g(\boldsymbol{x};\mathcal{D})\big)^2\big] \\
&\qquad+ 2\cdot \mathrm{E}_\mathcal{D}\big[\big(\mathrm{E}[y|x] - \mathrm{E}_\mathcal{D}[g(\boldsymbol{x};\mathcal{D})]\big)\big(\mathrm{E}_\mathcal{D}[g(\boldsymbol{x};\mathcal{D})] - g(\boldsymbol{x};\mathcal{D})\big)\big] \\
&\quad= \big(\mathrm{E}[y|\boldsymbol{x}] - \mathrm{E}_\mathcal{D}[g(\boldsymbol{x};\mathcal{D})]\big)^2 + \mathrm{E}_\mathcal{D}\big[\big(g(\boldsymbol{x};\mathcal{D}) - \mathrm{E}_\mathcal{D}[g(\boldsymbol{x};\mathcal{D})]\big)^2\big] \\
&\qquad+ 2\big(\mathrm{E}[y|\boldsymbol{x}] - \mathrm{E}_\mathcal{D}[g(\boldsymbol{x};\mathcal{D})]\big)\cdot \mathrm{E}_\mathcal{D}\big[\big(\mathrm{E}_\mathcal{D}[g(\boldsymbol{x};\mathcal{D})] - g(\boldsymbol{x};\mathcal{D})\big)\big] \\
&\quad= \underbrace{\big(\mathrm{E}[y|\boldsymbol{x}] - \mathrm{E}_\mathcal{D}[g(\boldsymbol{x};\mathcal{D})]\big)^2}_{\text{Bias}^2} + \underbrace{\mathrm{E}_\mathcal{D}\big[\big(g(\boldsymbol{x};\mathcal{D}) - \mathrm{E}_\mathcal{D}[g(\boldsymbol{x};\mathcal{D})]\big)^2\big]}_{\text{Varianz}}
\end{aligned}
\tag{3.26}
$$

Der Modellierungsfehler setzt sich also additiv aus einem quadratischen Bias-Term und einem Varianz-Term zusammen. Der Bias beschreibt, wie stark das Modell im Mittel über alle möglichen Trainingsdatensätze von der wahren Regression abweicht. Das Modell heißt *biasfrei*, wenn dieser Term verschwindet. Der Varianzterm gibt hier an, wie stark die einzelnen, auf den verschiedenen Trainingsdatensätzen trainierten Modelle gegenüber ihrem Mittelwert streuen, ist also ein Maß für die Empfindlichkeit des Modells gegenüber den Trainingsdaten.

Ursache eines großen Generalisierungsfehlers kann somit ein großer Bias oder eine große Varianz sein. Im Falle eines großen Bias fehlt dem Modell die nötige Flexibilität, um die Zusammenhänge zwischen den Daten zu beschreiben, das Modell leidet an Underfitting. Bei einem Modell der Form (3.9) entspricht das einer zu geringen Anzahl M der Basisfunktionen. Ist hingegen eine große Vari-

anz die Ursache für schlechte Generalisierung, so ist das Modell zu flexibel, da es auch die statistischen Eigenheiten der jeweiligen konkreten Realisierung der Trainingsdaten mit modelliert und somit starken Schwankungen bzgl. verschiedener Realisierungen unterworfen ist, also an Overfitting leidet. In diesem Fall ist die Anzahl der Basisfunktionen in (3.9) zu groß. Jede Form der Vermeidung von Overfitting resultiert automatisch in einem nichtverschwindenden Bias [40]. Je kleiner der Bias ist, desto größer ist die Varianz und umgekehrt. Dieses Verhalten wird von GEMAN et al. in [41] als das *Bias/Varianz-Dilemma* bezeichnet.

Zwei Beispiele verdeutlichen diesen Sachverhalt: Zur Vereinfachung seien N paarweise verschiedene Eingabedaten $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ fest vorgegeben. Die möglichen Trainings- und Testdatensätze $\mathcal{D}$ konstituieren sich dann durch unabhängige Messungen der zugehörigen Ausgaben, die wieder durch (3.6) verknüpft sind. Zufällig sind hier also immer nur die Ausgaben. Das erste Modell sei eine Interpolierende der Trainingsdaten (z.B. lineare Spline-Interpolation), also ein extrem flexibles Modell, das die Trainingsdaten exakt abbilden kann. Dieses Modell ist biasfrei, denn es gilt

$$\mathrm{E}_{\mathcal{D}}[g(\boldsymbol{x}_t;\mathcal{D})] = \mathrm{E}_{\mathcal{D}}[f(\boldsymbol{x}_t) + \epsilon_t] = f(\boldsymbol{x}_t) = \mathrm{E}[y|\boldsymbol{x}_t] \quad \forall t = 1, \ldots, N \tag{3.27}$$

Andererseits hängt dieses Modell sehr empfindlich von $\mathcal{D}$ ab, denn für seine Varianz ergibt sich

$$\mathrm{E}_{\mathcal{D}}\big[\big(g(\boldsymbol{x}_t;\mathcal{D}) - \mathrm{E}_{\mathcal{D}}[g(\boldsymbol{x}_t;\mathcal{D})]\big)^2\big] = \mathrm{E}_{\mathcal{D}}\big[\big(f(\boldsymbol{x}_t) + \epsilon_t - f(\boldsymbol{x}_t)\big)^2\big] = \sigma_\epsilon^2 \tag{3.28}$$

Das zweite Modell verkörpert das Gegenbeispiel: Nun sei $g(\boldsymbol{x};\mathcal{D}) \equiv h(\boldsymbol{x})$ für eine beliebige Funktion $h(\boldsymbol{x})$, also *unabhängig* von den Trainingsdaten und damit varianzfrei. Dieses Modell wird allerdings einen großen Bias haben, da es sich überhaupt nicht an die jeweiligen Trainingsdaten anpasst.

Um ein Modell mit guten Generalisierungseigenschaften zu erhalten, ist es also nötig, einen Kompromiss einzugehen, der darauf hinausläuft, einen gewissen Bias des Modells zuzulassen, um im Gegenzug eine kleine Varianz zu erreichen.

3.2.3 Modellvalidierung

Wie bereits im vorigen Abschnitt erläutert wurde, ist der mittlere quadratische Fehler auf den Trainingsdaten ein schlechter Schätzer für den Generalisierungsfehler eines Modells, denn der Trainingsfehler fällt monoton mit steigender Mo-

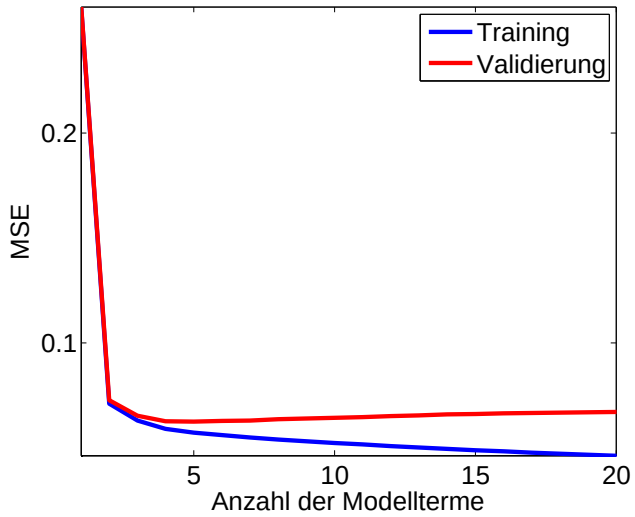


Abbildung 3.1: Typischer Verlauf von Trainings- und Validierungsfehler

dellkomplexität und bietet keinen Anhaltspunkt zur Erkennung von Overfitting. Abhilfe kann die schon erwähnte Aufteilung des Trainingsdatensatzes in eine Trainings- und eine Testdatenmenge liefern. Hastie et al. [24] schlagen sogar eine Dreiteilung in eine Trainings-, eine Validierungs- und eine Testdatenmenge vor. Dabei wird das Modell auf der Trainingsmenge konstruiert, die Validierungsmenge dient zur Erkennung von Overfitting und die Testmenge wird schließlich verwendet, um den Generalisierungsfehler des finalen Modells abzuschätzen. Für das aus einer Linearkombination von Basisfunktionen bestehende Modell (3.9) bedeutet das z.B., dass die Auswahl der Basisfunktionen sowie die Berechnung ihrer Koeffizienten anhand der Trainingsdaten vorgenommen wird, während die Validierungsdaten zur Bestimmung der optimalen Anzahl M der Terme dienen. Ein typischer Verlauf von Trainings- und Validierungsfehler über der Termanzahl und damit der Modellkomplexität ist in Abb. 3.1 dargestellt. Während der Traingsfehler mit steigender Termzahl monoton fällt, folgt ihm der Validierungsfehler zu Beginn, um bei einer gewissen Termzahl sein Minimum zu erreichen und anschließend wieder zu steigen. Das Minimum markiert die optimale Termanzahl. Auf diese Art und Weise könnte so eine Anzahl von Modellen konstruiert werden, aus der man das beste Modell als dasjenige identifizieren kann, das den kleinsten Validierungsfehler besitzt. Dieser jedoch wäre wiederum ein zu optimistischer Schätzer für den Generalisierungsfehler, da auf diese Art und Weise schließlich auch die Validierungsdaten in die Konstruktion bzw. Auswahl des finalen, besten Modells eingehen. Daher wird noch ein dritter, unabhängiger Datensatz –

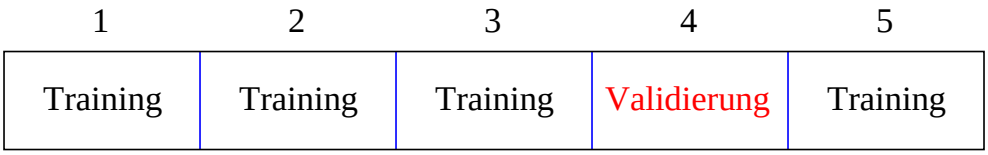


Abbildung 3.2: Aufteilung der Trainingsdaten für eine 5-fach Cross-Validation

die Testmenge – benötigt, auf der die Schätzung des Generalisierungsfehlers des besten Modells erfolgt.

Diese Art der Dreiteilung der Trainingsdaten erfordert allerdings eine sehr große Zahl an Trainingsdaten, denn effektiv wird so nur ein Teil der Daten zur Modellkonstruktion genutzt. Da die Genauigkeit statistischer Schätzungen aber mit abnehmender Länge der Stichprobe sinkt, ist es fraglich, ob man im Falle einer nicht sehr großen Datenmenge nicht ein besseres Modell erhalten kann, wenn man die gesamte Trainingsdatenmenge zur Modellkonstruktion heranzieht. Für diesen Fall relativ weniger Trainingsdaten wurden verschiedene analytische Verfahren zur Abschätzung der Diskrepanz zwischen Trainings- und Testfehler entwickelt (vgl. Abb. 3.1). Diese berechnen eine Schätzung des Testfehlers, indem sie dem Trainingsfehler einen Term hinzufügen, der die zu optimistische Schätzung des Testfehlers durch den Trainingsfehler kompensieren soll. Üblicherweise wächst dieser Zusatzterm mit der Komplexität des Modells (also z.B. mit der Anzahl der Basisfunktionen) und mit der Stärke des Rauschens und fällt mit der Anzahl der Trainingsdaten. Beispiele für solche Verfahren sind AIC (*Akaike information criterion*), BIC (*Bayesian information criterion*), C_p oder MDL (*Minimum description length*). Eine Beschreibung dieser Verfahren findet man z.B. in [24] oder [42]. Tatsächlich liefern diese Verfahren nur eine Schätzung des *In-sample*-Vorhersagefehlers, d.h. des erwarteten Fehlers für neue Ausgaben $y_i^{(\text{new})}$ zu den gleichen Eingaben $\boldsymbol{x}_i$, die auch für das Training verwendet wurden. Der eigentliche Generalisierungsfehler ist jedoch ein *Extra-sample*-Vorhersagefehler, bezieht sich also auf neue Ein- und die zugehörigen Ausgaben. Eine direkte Schätzung dieses Testfehlers auch für nicht sehr umfangreiche Trainingsdatensätze bietet die sogenannte *Cross-Validation* (CV, [43]). Dabei wird ein Trainingsdatensatz wiederholt in eine Trainings- und eine Validierungsmenge unterteilt, das Modell jeweils auf der Trainingsmenge konstruiert und auf der Validierungsmenge evaluiert. Bei der K-fachen Cross-Validation werden die Trainingsdatenpaare jeweils zufällig einer von K Klassen zugeordnet. Abb. 3.2 zeigt das Schema der Aufteilung für $K = 5$. Für jede Klasse $k = 1, \ldots, K$ wird das Modell auf den übrigen $K - 1$ Klassen

trainiert und der Vorhersagefehler dieses Modells auf der k-ten Klasse berechnet. Der Mittelwert dieser Vorhersagefehler ist der Cross-Validation Fehler. Genauer: Sei $\kappa : \{1, \ldots, N\} \mapsto \{1, \ldots, K\}$ eine Indexfunktion, die dem i-ten Trainingsdatenpaar $(\boldsymbol{x}_i, y_i)$ die Klasse $\kappa(i)$ zuordnet, und sei weiter $g_{-k}(\boldsymbol{x})$ das Modell, das auf allen außer der zur k-ten Klasse gehörenden Trainingsdaten trainiert wurde. Dann ist der CV-Fehler definiert als

$$J_{\mathrm{CV}}(g) = \frac{1}{N} \sum_{t=1}^{N} L(y_t, g_{-\kappa(t)}(\boldsymbol{x}_t)) \tag{3.29}$$

mit der Loss-Funktion (3.3). Enthält die Modellarchitektur Parameter zur Steuerung der Modellkomplexität, z.B. die Anzahl M der Basisfunktionen beim Modellansatz (3.9), so können diese durch Minimierung des CV-Fehlers bestimmt werden.

Eine extreme Form der Cross-Validation ist der Fall $K = N$, für den $\kappa(i) \equiv i$ ist. Hier wird jeweils lediglich ein einzelner Datenpunkt zur Validierung des auf allen übrigen $N - 1$ Trainingsdaten konstruierten Modells verwendet. Diese CV-Variante wird als *Leave-one-out Cross-Validation* (LOO-CV) oder auch als *Delete-1 Cross-Validation* bezeichnet. Die Summe der N quadratischen Vorhersagefehler für die Validierungsdaten wird nach Allen [44] als *PRESS* bezeichnet[5]. Um jedes Datum einmal zur Validierung zu verwenden, sind somit N Unterteilungen notwendig, wobei für jede Unterteilung das Modell neu trainiert werden muss, was nach (3.17) jedesmal eine Matrixinversion nach sich zieht und so auf den ersten Blick sehr rechenaufwändig zu sein scheint. Für solche Modelle, die linear in ihren Parametern sind, lassen sich die einzelnen Vorhersagefehler jedoch analytisch aus dem auf allen N Trainingsdaten trainierten Modell berechnen.

Dazu sei nun $\mathcal{D}_N = \{(\boldsymbol{x}_t, y_t) | t = 1, \ldots, N\}$ wieder die Menge der Trainingsdaten, $g(\boldsymbol{x})$ ein Model (3.9) aus M Termen, dessen lineare Koeffizienten $\boldsymbol{w}$ auf $\mathcal{D}_N$ durch Minimierung des MSE

$$\mathrm{MSE} = \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{G}\boldsymbol{w}\|_2^2 \tag{3.30}$$

bestimmt wurden, wobei $\boldsymbol{G}$ wieder die Designmatrix des Modells für $\mathcal{D}_N$ bezeichnet, $\hat{y}_i = g(\boldsymbol{x}_i)$ die Modellausgabe für die Eingabe $\boldsymbol{x}_i$ und $e_i = y_i - \hat{y}_i$ das zugehörige Residuum. $\mathcal{D}_{N,-i} = \mathcal{D}_N \setminus \{(\boldsymbol{x}_i, y_i)\}$ sei nun die aus $\mathcal{D}_N$ durch Entfernung des i-ten Trainingspaares hervorgehende Menge und $g_{-i}(\boldsymbol{x})$ das aus den gleichen Termen wie $g(\boldsymbol{x})$ bestehende Modell, dessen Koeffizientenvektor aber

[5] PRESS steht für **PRE**diction **S**um of **S**quares.

durch Minimierung von (3.30) auf $\mathcal{D}_{N,-i}$ bestimmt wurde. Die zugehörige Designmatrix $\boldsymbol{G}_{-i}$ entsteht aus $\boldsymbol{G}$ durch Entfernen der i-ten Zeile[6]. Schließlich bezeichnen $\hat{y}_{-i} = g_{-i}(\boldsymbol{x}_i)$ und $e_{-i} = y_i - \hat{y}_{-i}$ die Modellausgabe und das Residuum dieses Modells für die Eingabe $\boldsymbol{x}_i$. Dann ist der LOO-Fehler für das Modell $g(\boldsymbol{x})$ gegeben durch

$$J_{\mathrm{LOO}} = \frac{1}{N}\sum_{t=1}^{N}(y_t - g_{-t}(\boldsymbol{x}_t))^2 = \frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_{-t})^2 = \frac{1}{N}\sum_{t=1}^{N} e_{-t}^2 \,. \tag{3.31}$$

Mit Hilfe des *Sherman-Morrison-Woodbury-Theorems*[7] [39] lassen sich die einzelnen Summanden analytisch aus dem einmal auf allen N Trainingsdaten gefitteten Modell $g(\boldsymbol{x})$ berechnen, denn für die Koeffizienten $\boldsymbol{w}$ von $g(\boldsymbol{x})$ gilt

$$\boldsymbol{w} = \boldsymbol{H}^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{y} \tag{3.32}$$

mit der Abkürzung

$$\boldsymbol{H} = \boldsymbol{G}^{\mathrm{T}}\boldsymbol{G} \,. \tag{3.33}$$

Entsprechend gilt $\boldsymbol{w}_{-i} = (\boldsymbol{G}_{-i}{}^{\mathrm{T}}\boldsymbol{G}_{-i})^{-1}\boldsymbol{G}_{-i}{}^{\mathrm{T}}\boldsymbol{y}_{-i}$, wobei $\boldsymbol{y}_{-i}$ aus $\boldsymbol{y}$ durch Entfernen der i-ten Komponente y_i hervorgeht. Wegen[8] $\boldsymbol{G}_{-i}{}^{\mathrm{T}}\boldsymbol{G}_{-i} = \boldsymbol{G}^{\mathrm{T}}\boldsymbol{G} - \boldsymbol{G}(i,:)^{\mathrm{T}}\boldsymbol{G}(i,:)$ und $\boldsymbol{G}_{-i}{}^{\mathrm{T}}\boldsymbol{y}_{-i} = \boldsymbol{G}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{G}(i,:)^{\mathrm{T}}y_i$ gilt nun mit dem Sherman-Morrison-Woodbury-Theorem

$$\begin{aligned}
\hat{y}_{-i} &= \boldsymbol{G}(i,:)\boldsymbol{w}_{-i} \\
&= \boldsymbol{G}(i,:)\big(\boldsymbol{H} - \boldsymbol{G}(i,:)^{\mathrm{T}}\boldsymbol{G}(i,:)\big)^{-1}\big(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{G}(i,:)^{\mathrm{T}}y_i\big) \\
&= \boldsymbol{G}(i,:)\big(\boldsymbol{H}^{-1} + \boldsymbol{H}^{-1}\boldsymbol{G}(i,:)^{\mathrm{T}}d_i^{-1}\boldsymbol{G}(i,:)\boldsymbol{H}^{-1}\big)\big(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{G}(i,:)^{\mathrm{T}}y_i\big)
\end{aligned} \tag{3.34}$$

mit der skalaren Größe

$$d_i = 1 - \boldsymbol{G}(i,:)\boldsymbol{H}^{-1}\boldsymbol{G}(i,:)^{\mathrm{T}} \,. \tag{3.35}$$

Ausmultiplizieren der Klammern in (3.34) und die Verwendung von (3.32) und der Abkürzung (3.35) ergibt für das Residuum des auf $\mathcal{D}_{N,-i}$ gefitteten Modells

[6] Für den Koeffizientenvektor $\boldsymbol{w}_{-i}$ hingegen gilt das nicht!

[7] Danach gilt für $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ und $\boldsymbol{x} \in \mathbb{R}^n$: $(\boldsymbol{A}-\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}})^{-1} = \boldsymbol{A}^{-1}+\boldsymbol{A}^{-1}\boldsymbol{x}(1-\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{x})^{-1}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}^{-1}$, falls $\boldsymbol{A}$ und $1 - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{x}$ nichtsingulär sind.

[8] Dabei bezeichnet $\boldsymbol{G}(i,:) \in \mathbb{R}^{1\times M}$ nach Matlab-Notation die i-te Zeile von $\boldsymbol{G}$.

schließlich

$$\begin{aligned}
e_{-i} &= y_i - \hat{y}_{-i} \\
&= y_i - \boldsymbol{G}(i,:)\big[\boldsymbol{w} + \boldsymbol{H}^{-1}\boldsymbol{G}(i,:)^{\mathrm{T}} d_i^{-1}\boldsymbol{G}(i,:)\boldsymbol{w} - \boldsymbol{H}^{-1}\boldsymbol{G}(i,:)^{\mathrm{T}} y_i \\
&\quad - \boldsymbol{H}^{-1}\boldsymbol{G}(i,:)^{\mathrm{T}} d_i^{-1}\boldsymbol{G}(i,:)\boldsymbol{H}^{-1}\boldsymbol{G}(i,:)^{\mathrm{T}} y_i\big] \\
&= y_i - \boldsymbol{G}(i,:)\boldsymbol{w} - (1-d_i)d_i^{-1}\boldsymbol{G}(i,:)\boldsymbol{w} + (1-d_i)y_i \\
&\quad + (1-d_i)d_i^{-1}(1-d_i)y_i \\
&= y_i - d_i^{-1}\boldsymbol{G}(i,:)\boldsymbol{w} + (1-d_i)y_i + (d_i^{-1} - 2 + d_i)y_i \\
&= \frac{y_i - \boldsymbol{G}(i,:)\boldsymbol{w}}{d_i} \\
&= \frac{y_i - \hat{y}_i}{d_i} \\
&= \frac{e_i}{d_i}
\end{aligned} \tag{3.36}$$

und somit für den LOO-Fehler

$$J_{\mathrm{LOO}} = \frac{1}{N}\sum_{t=1}^{N}\left(\frac{y_t - \hat{y}_t}{1 - \boldsymbol{G}(t,:)(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G})^{-1}\boldsymbol{G}(t,:)^{\mathrm{T}}}\right)^2 . \tag{3.37}$$

Zur Berechnung dieser Größe müssen also nicht für alle N Aufteilungen die N Parameterfits berechnet werden, sondern es genügt die einmalige Inversion von $\boldsymbol{H} = \boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}$ für das auf allen Daten $\mathcal{D}_N$ gefittete Modell. In Abschnitt 3.3.3 wird ein Algorithmus vorgestellt, der den LOO-Fehler als Auswahlkriterium zur Termselektion verwendet.

Es stellt sich nun die Frage, in wie viele Klassen K die N Trainingsdaten für die Cross-Validation unterteilt werden sollten. Für $K = N$ ist der CV-Fehler näherungsweise biasfrei gegenüber dem wahren Testfehler, hat aber eine große Varianz, da die einzelnen Trainingsdatenmengen alle nahezu identisch sind. Ein kleiner Wert von z.B. $K = 5$ hat zwar eine kleine Varianz des CV-Fehlers zur Folge, allerdings kann es hier zu einem großen Bias kommen, je nachdem, wo auf der Lernkurve man sich befindet: Die Genauigkeit statistischer Schätzungen steigt mit der Anzahl der Trainingsdaten. Ist diese zu klein, ist keine genaue Schätzung möglich, was einen großen Bias zur Folge hat. Mit steigender Datenzahl wird die Schätzung besser und erreicht schließlich eine Sättigung, ab der eine weitere Vergrößerung der Trainingsmenge keine signifikante Verbesserung des Modells mehr bewirkt. Ist diese Sättigung auch für den $(K-1)/K$-ten Teil der Trainingsdaten schon erreicht, so hat die CV keine signifikante Vergrößerung des Bias zur Folge.

Ist die mit einer solchen Untermenge der Trainingsdaten erreichbare Genauigkeit jedoch deutlich schlechter als bei einem auf allen N Trainingsdaten trainierten Modell, so zieht dies eine entsprechende Vergrößerung des Bias nach sich. HASTIE et al. [24] schlagen als Kompromiss $K = 5$ oder $K = 10$ vor.

3.2.4 Regularisierung

In Abschnitt 3.2.2 wurde dargelegt, dass man sowohl Bias als auch Varianz möglichst klein halten muss, um ein Modell mit guten Generalisierungseigenschaften mit Hilfe einer beschränkten Menge an Trainingsdaten zu erhalten, und dass dies zwei gegensätzliche Forderungen sind, so dass ein Kompromiss eingegangen werden muss. Dieser resultiert meist in der Hoffnung, durch Inkaufnahme eines kleinen Bias' eine deutliche Reduktion der Varianz gegenüber dem biasfreien Modell und dadurch einen kleineren Generalisierungsfehler zu erreichen. Das bedeutet, dass die Komplexität des Modells durch Einführung eines geeigneten Bias' eingeschränkt werden muss. Bei der in dieser Arbeit verwendeten Architektur (3.9) einer Linearkombination von Basisfunktionen lässt sich dies z.B. durch die Beschränkung der Anzahl M der Basisfunktionen erreichen. Der Grad der Komplexität ist so allerdings nur in diskreten Schritten einstellbar, denn ein Term kann nur zum Modell dazugehören oder eben nicht. Dies kann wiederum eine große Varianz zur Folge haben, führt also nicht zwangsläufig zu einer guten Generalisierung. Eine Möglichkeit der kontinuierlichen Komplexitätssteuerung bietet die *Regularisierung*, die mit einer veränderten Kostenfunktion einhergeht. Dem MSE wird hier noch ein Term hinzugefügt, der eine Überanpassung des Modells verhindert, indem große Werte der Koeffizienten „bestraft“ werden[9]. Bei der populärsten Form der Regularisierung lässt sich die allgemeine Form der Kostenfunktion für ein Modell $g(\boldsymbol{x}|\boldsymbol{w})$ einer Linearkombination von M Basisfunktionen (3.9) schreiben als

$$J = \frac{1}{N}\|\boldsymbol{y} - \boldsymbol{G}\boldsymbol{w}\|_2^2 + \Omega_q(\boldsymbol{w})\,, \tag{3.38}$$

wobei der Regularisierungsterm die Form

$$\Omega_q(\boldsymbol{w}) = \lambda \sum_{k=1}^{M} |w_k|^q \tag{3.39}$$

[9] In der englischen Literatur werden diese Methoden daher auch als *shrinkage methods* bezeichnet.

hat. Mit Hilfe des Parameters $\lambda \geq 0$ lässt sich die Stärke der Regularisierung, also die Größe des Bias und damit die Einschränkung der Modellkomplexität, stufenlos steuern. Für die Wahl von $\lambda = 0$ entspricht (3.38) einfach der Minimierung des MSE, was bei einem komplexen Modell zu einem kleinen Bias aber auch zu großer Varianz führt. Mit größerem λ verringert sich die Varianz, aber der Bias steigt. Der optimale Kompromiss zeichnet sich durch einen minimalen Generalisierungsfehler aus und kann mit Hilfe der Cross-Validation bestimmt werden. Der Typus der Regularisierung wird durch den Parameter q spezifiziert. Gebräuchlich sind Werte $q \in \{0,1,2\}$. $q = 1$ ist der kleinste Wert, für den (3.39) und damit auch (3.38) eine konvexe Funktion von $\boldsymbol{w}$ ist und somit ein eindeutiges, globales Minimum besitzt. Diese Variante entspricht dem *Lasso*[10], $q = 2$ der *Ridge-Regression* oder *Tikhonov-Phillips-Regularisierung*. Auf diese beiden Varianten soll in den folgenden beiden Unterabschnitten näher eingegangen werden. Für $q = 0$ hingegen ist die Kostenfunktion nicht mehr konvex, was die Minimierung von (3.38) bedeutend erschwert. Diese Variante wurde daher in der vorliegenden Arbeit nicht verwendet.

Bemerkung: Bei Verwendung der Kostenfunktion (3.38) wird angenommen, dass die Ausgaben $\boldsymbol{y}$ sowie die Spalten der Design-Matrix zentriert sind, in $\boldsymbol{G}$ also kein konstanter Term enthalten ist. Eine Regularisierung des konstanten Terms würde die Lösung der Minimierung von (3.38) nämlich abhängig machen vom gewählten Ursprung. Weiterhin ist die Lösung des regularisierten Minimierungsproblems nicht invariant unter Skalierungen der Spalten der Design-Matrix. Um alle Spalten bei der Regularisierung gleich stark zu gewichten, bietet es sich daher an, diese mit ihrer Varianz zu skalieren. Für allgemeine, nicht zentrierte Daten und ein Modell $\hat{y} = w_0 + \sum_{k=1}^{M} w_k g_k(\boldsymbol{x})$ mit konstantem Term w_0 hat (3.38) die Form

$$J = \frac{1}{N} \sum_{t=1}^{N} \left(y_t - w_0 - \sum_{k=1}^{M} w_k g_{tk} \right)^2 + \lambda \sum_{k=1}^{M} |w_k|^q \,. \tag{3.40}$$

Durch die Einführung der zentrierten und skalierten Design-Matrix $\boldsymbol{G}' = (g'_{tk})$ mit

$$g'_{tk} = \frac{g_{tk} - \bar{g}_k}{s_k^2} \,, \tag{3.41}$$

wobei $\bar{g}_k = 1/N \sum_{t=1}^{N} g_{tk}$ der Mittelwert der k-ten Spalte und s_k^2 deren Varianz

[10] **l**east **a**bsolute **s**election and **s**hrinkage **o**perator

bezeichnen, sowie der Zentrierung der Ausgaben

$$y'_t = y_t - \bar{y} \tag{3.42}$$

mit $\bar{y} = 1/N \sum_{t=1}^{N} y_t$ wird (3.40) in die zu (3.38) äquivalente Form

$$J = \frac{1}{N} \sum_{t=1}^{N} \left(y'_t - \sum_{k=1}^{M} w'_k g'_{tk} \right)^2 + \lambda \sum_{k=1}^{M} |w'_k|^q \tag{3.43}$$

überführt, bei der der konstante Term verschwindet. Hat man eine Lösung gefunden, die (3.43) minimiert, ergeben sich die ursprünglichen Koeffizienten daraus zu

$$\begin{aligned} w_k &= w'_k / s_k^2 \,, \quad k = 1, \ldots, M \\ w_0 &= \bar{y} - \sum_{k=1}^{M} \frac{w'_k}{s_k^2} \bar{g}_k \,. \end{aligned} \tag{3.44}$$

In der Praxis zeigte sich allerdings kein großer Unterschied in den Ergebnissen, wenn statt der Normierung (3.41) lediglich die Ausgabedaten $\boldsymbol{y}$ durch Subtraktion von $\bar{y}$ mittelwertfrei gemacht wurden und der konstante Modellterm bei der Konstruktion des Modells ausgelassen und nach Abschluss des Trainings auf den Mittelwert $\bar{y}$ gesetzt wurde. Im Folgenden wird zur Vereinfachung der Notation angenommen, dass diese Zentrierung für $\boldsymbol{y}$ in (3.38) bereits durchgeführt wurde und der konstante Term somit bei der Konstruktion des Modells ausgelassen werden kann.

3.2.4.1 Ridge-Regression

Ridge-Regression [45] ist die populärste Form der Regularisierung, da sich in diesem Fall die Lösung für die Minimierung von (3.38) durch einen geschlossenen Ausdruck angeben lässt. Für $q = 2$ ist nämlich die Minimierung von (3.38) äquivalent zur Minimierung von

$$\mathrm{SSE}_{\mathrm{RR}} = \|\boldsymbol{y} - \boldsymbol{G}\boldsymbol{w}\|_2^2 + N\lambda\|\boldsymbol{w}\|_2^2 \,. \tag{3.45}$$

Erweitert man nun die Design-Matrix und den Vektor der Ausgaben und definiert

$$\tilde{\boldsymbol{G}} := \begin{pmatrix} \boldsymbol{G} \\ \sqrt{N\lambda} \cdot \mathbb{1}_M \end{pmatrix} \qquad \text{und} \qquad \tilde{\boldsymbol{y}} := \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{pmatrix}, \tag{3.46}$$

so lässt sich (3.45) schreiben als

$$\mathrm{SSE}_{\mathrm{RR}} = \left\| \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{pmatrix} - \begin{pmatrix} \boldsymbol{G} \\ \sqrt{N\lambda} \cdot \mathbb{1}_M \end{pmatrix} \boldsymbol{w} \right\|_2^2 = \|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{G}}\boldsymbol{w}\|_2^2 . \tag{3.47}$$

Die Lösung des regularisierten Minimierungsproblems

$$\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}\in\mathbb{R}^M} (\|\boldsymbol{y} - \boldsymbol{G}\boldsymbol{w}\|_2^2 + N\lambda\|\boldsymbol{w}\|_2^2) \tag{3.48}$$

ist damit zurückgeführt auf die Lösung des Standardproblems (3.16)

$$\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}\in\mathbb{R}^M} \|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{G}}\boldsymbol{w}\|_2^2 , \tag{3.49}$$

die sich analog zu (3.18) aus den Normalengleichungen (3.17) für $\tilde{\boldsymbol{G}}$ zu

$$\boldsymbol{w}^\star = (\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{y}} = (\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G} + N\lambda\mathbb{1})^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{y} \tag{3.50}$$

ergibt, ist also eine lineare Transformation der Ausgaben $\boldsymbol{y}$. Sie lässt sich auch mit Hilfe der SVD von $\boldsymbol{G}$ (3.19) ausdrücken, denn es gilt

$$\boldsymbol{w}^\star = \tilde{\boldsymbol{G}}^\dagger\tilde{\boldsymbol{y}} = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + N\lambda}(\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{y})\boldsymbol{v}_i \tag{3.51}$$

[46]. An (3.47) lässt sich direkt ablesen, dass die Tikhonov-Phillips-Regularisierung der Hinzunahme artifizieller Samples zu den Trainingsdaten entspricht, die in der Erweiterung der Design-Matrix $\boldsymbol{G}$ um die Diagonalmatrix $\sqrt{N\lambda} \cdot \mathbb{1}_M$ resultiert. Diese artifiziellen Samples erzwingen die Eindeutigkeit der Lösung von (3.49), da $\tilde{\boldsymbol{G}}$ für $\lambda > 0$ aufgrund der Diagonalgestalt dieser Erweiterung immer vollen Rang hat, selbst wenn $\boldsymbol{G}$ singulär ist. Effektiv bewirkt die Ridge-Regression ein betragsmäßiges Schrumpfen aller Koeffizienten Richtung Null, was sich direkt an (3.51) ablesen lässt. Im Unterschied zur Lasso-Regularisierung wird dabei aber i.Allg. keiner der Koeffizienten exakt zu Null.

3.2.4.2 Lasso

Beim Lasso [47] lautet die zu minimierende Kostenfunktion mit $q = 1$ (3.38)

$$J_{\text{Lasso}} = \frac{1}{N}\|\boldsymbol{y} - \boldsymbol{G}\boldsymbol{w}\|_2^2 + \lambda \sum_{k=1}^{M} |w_k| \,. \tag{3.52}$$

Im Unterschied zur Ridge-Regression hängt die Lösung $\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}\in\mathbb{R}^M} J_{\text{Lasso}}$ nichtlinear von den $\boldsymbol{y}$ ab und lässt sich nicht geschlossen angeben. Zur Lösung müssen daher numerische Optimierungsverfahren eingesetzt werden, wobei darauf zu achten ist, dass $\partial J_{\text{Lasso}}/\partial w_j$ für $w_j = 0$ nicht existiert. Wie bei der Ridge-Regression werden die Koeffizienten mit größer werdendem λ immer stärker geschrumpft, wobei diese beim Lasso tatsächlich exakt Null werden können. Auf diese Art und Weise entspricht die Wirkung des Lasso nicht nur einer Verringerung der Varianz durch eine Einschränkung der Modellkomplexität, sondern gleichzeitig auch einer Termselektion, indem nichtsignifikante Basisfunktionen durch Nullsetzen ihrer Koeffizienten aus dem Modell entfernt werden.

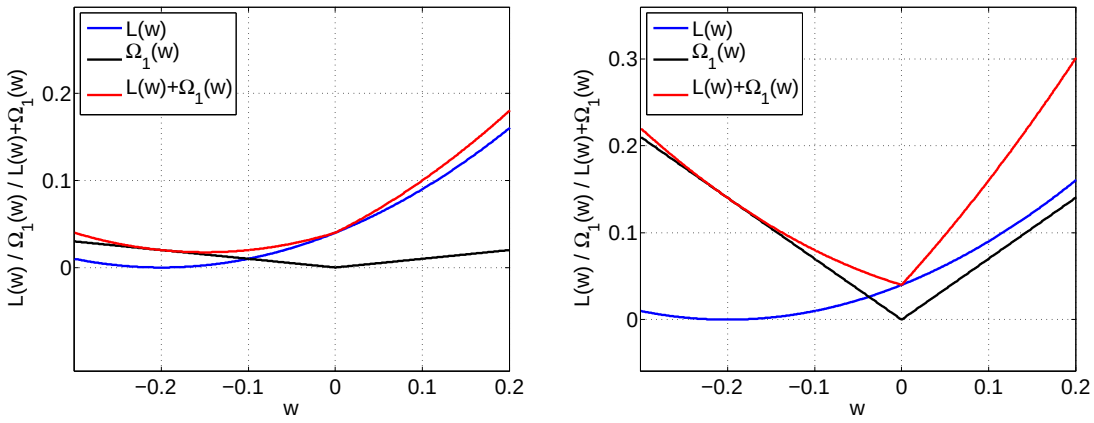
Der Grund dafür, dass bei der l_1-Regularisierung Koeffizienten häufig exakt auf den Wert Null geschrumpft werden, liegt darin, dass für die Ableitung $\Omega_1'(w)$ des Regularisierungsterms $\forall w < 0\ \Omega_1'(w) = -\lambda_1$ gilt, also auch für die linksseitige Ableitung im Limes $w \nearrow 0$: $\Omega_{1-}'(0) = \lim_{h\nearrow 0}(\Omega_1(h) - \Omega_1(0))/h = -\lambda_1$, während $\forall w > 0\ \Omega_1'(w) = \lambda_1$ und somit auch für die rechsseitige Ableitung $\Omega_{1+}'(0) = \lim_{h\searrow 0}(\Omega_1(h) - \Omega_1(0))/h = \lambda_1$. Entscheidend ist also die Diskontinuität der Ableitung der Betragsfunktion im Ursprung. Diese tritt bei der Ridge-Regression nicht auf, da dort der links- und rechtsseitige Grenzwert übereinstimmen und gegen die Ableitung konvergieren, die im Ursprung verschwindet: $\Omega_{2-}'(0) = \Omega_{2+}'(0) = \Omega_2'(0) = 0$. Den Effekt der Diskontinuität auf die Lage des Minimums der Lasso-Kostenfunktion (3.52) wird klar, wenn man diese in der Form

$$J_{\text{Lasso}}(\boldsymbol{w}) = \bar{L}(\boldsymbol{w}) + \Omega_1(\boldsymbol{w}) \tag{3.53}$$

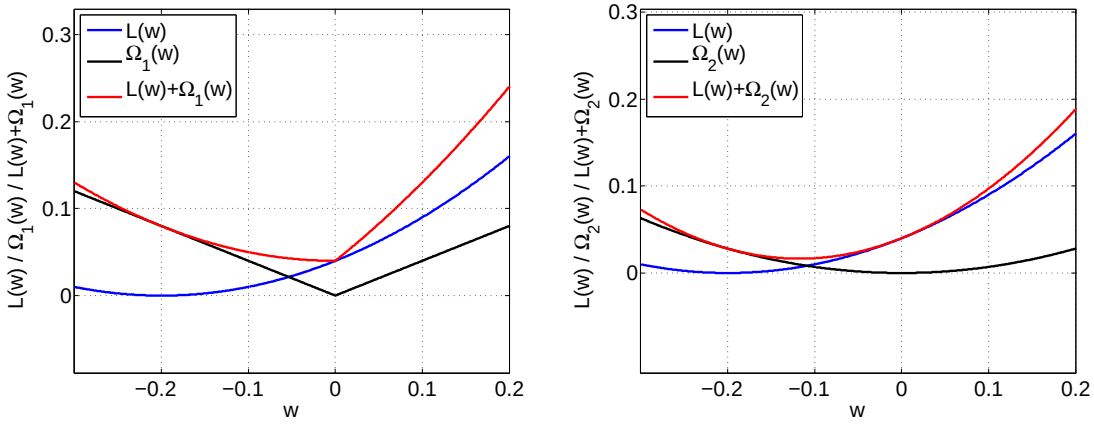
mit dem mittleren Loss $\bar{L}(\boldsymbol{w}) = 1/N \sum_{t=1}^{N}(y_t - g(\boldsymbol{x}_t|\boldsymbol{w}))^2$ schreibt. Dann ist

$$\frac{\partial J_{\text{Lasso}}}{\partial w_j} = \frac{\partial \bar{L}(\boldsymbol{w})}{\partial w_j} + \lambda_1 \operatorname{sign}(w_j) \quad \forall w_j \neq 0 \,. \tag{3.54}$$

Nun sei $w_j = 0$, und die Frage ist, ob sich der Wert der Kostenfunktion verkleinern lässt, wenn w_j aus der Nulllage verschoben wird. Dazu sei zunächst angenommen, dass die Ableitung des mittleren quadratischen Fehlers nach w_j größer



(a) Lasso, $\lambda_1 = 0\ 1$ (b) Lasso, $\lambda_1 = 0\ 7$



(c) Lasso, $\lambda_1 = 0\ 4$ (d) Ridge-Regression, $\lambda_2 = 0\ 7$

Abbildung 3.3: Effekt der Regularisierung $\Omega_q(w) = \lambda|w|^q$ auf die Lage des Minimums w_{min} der Kostenfunktion $J = L(w) + \Omega_q(w)$ am Beispiel der Loss-Funktion $L(w) = (w + 0\ 2)^2$ für $q = 1$ (Lasso) bzw. $q = 2$ (Ridge-Regression). In (a) ist $\lambda_1 < |L'(0)| = 0\ 4$ und damit $w_{\text{min}} = -0\ 15 < 0$. In (b) ist $\lambda_1 > |L'(0)|$ und (c) zeigt den Grenzfall $\lambda_1 = |L'(0)|$; in beiden Fällen überwiegt der Aufschlag durch die Regularisierung den Abfall des Loss, so dass $w_{\text{min}} = 0$. Zum Vergleich zeigt (d) die Ridge-Regression, für die $w_{\text{min}} = -0\ 4/(1+\lambda_2)$ gilt und das Minimum somit erst für $\lambda_2 \to \infty$ gegen Null konvergiert.

ist als λ_1: $\partial\bar{L}/\partial w_j > \lambda_1$. Um den Loss zu verringern, muss in diesem Fall w_j in negativer Richtung aus dem Ursprung verschoben werden. Eine infinitesimale Verschiebung in diese Richtung verringert sofort den Wert der Ableitung um λ_1. Da aber $\partial\bar{L}/\partial w_j - \lambda_1 > 0$, ist $w_j < 0$ im Minimum von J_{Lasso}. Mit dem gleichen Argument folgt für den Fall $\partial\bar{L}(\mathbf{0})/\partial w_j < -\lambda_1$, dass im Minimum von J_{Lasso} $w_j > 0$ gilt. In diesen beiden Fällen überwiegt die Verkleinerung des Loss also den Aufschlag durch den Regularisierungsterm, wenn w_j in Richtung des negativen Gradienten von $\bar{L}$ aus dem Ursprung verschoben wird. Die Regularisierung führt damit zwar zu einer Verkleinerung des Absolutwertes von w_j im Vergleich zum Fall ohne Regularisierung, aber der Koeffizient verschwindet nicht. Gilt aber andererseits $|\partial\bar{L}(\mathbf{0})/\partial w_j| \leq \lambda_1$, so wird bei Verschiebung von w_j aus dem Ursprung der Abfall des Loss durch den Aufschlag des Regularisierungsterms überkompensiert, so dass keine Verkleinerung von J_{Lasso} möglich ist, das Minimum also bei $w_j = 0$ liegt. Das Beispiel in Abb. 3.3 verdeutlicht diese Überlegungen nochmals grafisch und quantitativ für verschiedene Stärken der l_1-Regularisierung und im Vergleich zur Ridge-Regression.

Zusammenfassend gilt also, dass ein Koeffizient w_j ($j \in \{1, \ldots, M\}$) im Minimum von (3.53) für ein gegebenes $\lambda_1 \geq 0$ nur dann ungleich Null ist, wenn die Bedingung

$$\left|\frac{\partial\bar{L}(\mathbf{0})}{\partial w_j}\right| > \lambda_1 \tag{3.55}$$

erfüllt ist.

3.3 Termselektionsalgorithmen

Neben der Regularisierung ist die Termselektion eine Möglichkeit zur Steuerung der Modellkomplexität. Dabei werden aus einem großen Vorrat an Modelltermen, der im Folgenden oft als Kandidatenpool bezeichnet wird, nur die signifikanten Terme ausgewählt, wobei das Kriterium die Minimierung einer Kostenfunktion ist, z.B. des MSE. Im Fall der hier verwendeten RBF-Modelle (3.22) kann man den Kandidatenpool z.B. dadurch erzeugen, dass man im Eingaberaum jeden Ort der Trainingsdaten als Zentrum einer Basisfunktion verwendet – oder sogar mehrerer RBF-Terme unterschiedlicher Breitenskalierungen. Ein solches Modell wird sehr wahrscheinlich an Overfitting leiden. Seine Komplexität lässt sich zwar durch den Einsatz der in Abschnitt 3.2.4 besprochenen Regularisierung einschränken, jedoch führt insbesondere die Ridge-Regression nur zur Schrumpfung der Koeffizienten,

wobei i. Allg. keiner tatsächlich Null wird und den entsprechenden Term somit vom Modell ausschließt. Einerseits bietet ein großer Kandidatenpool Vorteile im Hinblick auf die erreichbare Genauigkeit eines zu konstruierenden Modells, andererseits soll das fertige Modell aber möglichst kompakt sein, die Architektur (3.9) also aus möglichst wenigen Basisfunktionen $g_k(\boldsymbol{x})$ bestehen, um den Rechenaufwand bei der Anwendung des Modells auf (neue) Daten gering zu halten und die Interpretierbarkeit des Modells zu verbessern. Die Termselektion bietet eine Möglichkeit zur Lösung dieses Problems und wird oft simultan mit der Regularisierung angewendet, indem als Selektionskriterium die Minimierung einer regularisierten Kostenfunktion wie (3.45) verwendet wird. Für einen gegebenen Kandidatenpool aus P Termen liegt die Aufgabe dann darin, zu bestimmen, *welches* die geeigneten Basisfunktionen sind und wie groß die optimale *Anzahl* M der Terme ist. Die naheliegende Möglichkeit, für jede Modellgröße von 1 bis P Termen die optimale Untermenge an Termen zu bestimmen, diese Modelle mit Hilfe der Cross-Validation zu validieren und M aus dem Minimum des CV-Fehlers zu ermitteln, ist in der Praxis meist nicht durchführbar, da es $\binom{P}{k}$ Möglichkeiten gibt, aus dem Kandidatenpool ein Modell aus k Termen zu konstruieren. Die Anzahl aller möglichen Auswahlmöglichkeiten für $k = 1, \ldots, P$ ist daher $2^P - 1$, wächst also exponentiell mit der Größe des Kandidatenpools und ist auch mit *Branch-and-bound*-Algorithmen [48] lediglich bis $P = 30$ oder $P = 40$ handhabbar. Der Suchraum für die Termauswahl muss also in geeigneter Weise eingeschränkt werden. Dies geschieht oft durch den Einsatz von *Greedy*-Algorithmen (dt. gierig, erfolgshungrig), die iterativ arbeiten und in jedem Iterationsschritt den Erfolg maximieren. All diesen Greedy-Verfahren ist gemein, dass sie lediglich eine bzgl. ihrer beschränkten Suche optimale Auswahl an Termen finden, die nicht notwendigerweise mit der global optimalen Auswahl aus der Menge aller möglichen Kombinationen übereinstimmt. Die einfachsten Greedy-Varianten zur Termauswahl sind die *Forward Selection* und die *Backward Elimination*[11]. Erstere startet mit einem leeren Modell und fügt in jedem Iterationsschritt den Term zum Modell hinzu, der zusammen mit den schon in früheren Schritten ausgewählten Termen die größte Reduktion der Kostenfunktion bewirkt. Bei der Backward Elimination startet man umgekehrt mit einem Modell, das zunächst alle Terme enthält, und entfernt in jedem Iterationsschritt den Term, der zum geringsten Anstieg der Kostenfunktion führt. Für den Fall einer quadratischen Loss-Funktion mit Tikhonov-Phillips-Regularisierung existieren für beide Methoden effiziente Algo-

[11] Geläufig sind auch die Bezeichnungen *Forward-stepwise Selection* bzw. *Backward-stepwise Selection*.

rithmen, die in den folgenden Unterabschnitten näher besprochen werden. Eine Besprechung verschiedener Termselektionsalgorithmen ist in [48] zu finden.

3.3.1 Forward Selection

Die Forward Selection ist ein iteratives Selektionsverfahren, bei der in jedem Iterationsschritt ein Term zum Modell hinzugefügt wird. Es sei wieder die Menge der Trainingsdaten $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)|i = 1, \ldots, N\}$ gegeben sowie ein Pool von P Kandidatentermen[12] $\{\boldsymbol{g}_j|j = 1, \ldots, P\}$. Im k-ten Iterationsschritt wird der Term $\boldsymbol{g}_{i_k}$ aus dem Kandidatenpool zum Modell hinzugefügt, der zusammen mit den zuvor ausgewählten Termen $\boldsymbol{g}_{i_1}, \ldots, \boldsymbol{g}_{i_{k-1}}$ die größte Reduktion des MSE ergibt. Mit der Indexmenge

$$\mathcal{I}_k = \{1, \ldots, P\} \setminus \{i_1, \ldots, i_{k-1}\} \tag{3.56}$$

der in Schritt k noch zur Verfügung stehenden Kandidatenterme und $\boldsymbol{G}_{k-1} = (\boldsymbol{g}_{i_1}, \ldots, \boldsymbol{g}_{i_{k-1}})$ ist das der Term $\boldsymbol{g}_{i_k}$, für den

$$i_k = \arg\min_{i \in \mathcal{I}_k} \|\boldsymbol{y} - (\boldsymbol{G}_{k-1}, \boldsymbol{g}_i)\tilde{\boldsymbol{w}}_k\|_2^2 \tag{3.57}$$

gilt, wobei $\tilde{\boldsymbol{w}}_k$ die Normalengleichungen

$$\tilde{\boldsymbol{G}}_k^{\mathrm{T}} \tilde{\boldsymbol{G}}_k \tilde{\boldsymbol{w}}_k = \tilde{\boldsymbol{G}}_k^{\mathrm{T}} \boldsymbol{y} \tag{3.58}$$

erfüllt und $\tilde{\boldsymbol{G}}_k = (\boldsymbol{G}_{k-1}, \boldsymbol{g}_i)$ ist. In jedem Iterationsschritt muss also jeder der verbleibenden Kandidatenterme testweise ins Modell aufgenommen und der zugehörige Koeffizientenvektor berechnet werden, der (3.58) erfüllt, um den besten Term nach (3.57) zu bestimmen. Für einen großen Kandidatenpool erscheint der hierfür notwendige Rechenaufwand zunächst beträchtlich. Die nötigen Berechnungen vereinfachen sich jedoch erheblich, falls die P Kandidaten aus dem Pool paarweise orthogonal sind, also $\boldsymbol{g}_i^{\mathrm{T}} \boldsymbol{g}_j = 0 \ \forall i, j = 1, \ldots, P, \ i \neq j$ gilt. Dann sind die Beiträge der einzelnen Terme zur Modellausgabe unabhängig voneinander und die Koeffizientenmatrix der Normalengleichungen hat Diagonalgestalt: $\boldsymbol{G}^{\mathrm{T}} \boldsymbol{G} = \mathrm{diag}(\kappa_1, \ldots, \kappa_P)$ mit $\kappa_i = \|\boldsymbol{g}_i\|_2^2$. Für den Koeffizient w_k des Terms $\boldsymbol{g}_k$

[12] Es wird hier zur Vereinfachung sowohl die Basisfunktion $g_j(\boldsymbol{x})$ als auch die entsprechende Spalte $\boldsymbol{g}_j \in \mathbb{R}^N$ der Designmatrix $\boldsymbol{G}$, die durch Einsetzen der Trainingsdaten in $g_j(\boldsymbol{x})$ entsteht, als Term bezeichnet.

gilt dann nach (3.58) einfach

$$w_k = \frac{\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{y}}{\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{g}_k} = \frac{\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{y}}{\kappa_k}\,, \tag{3.59}$$

und der quadratische Fehler für das aus allen P Termen bestehende Modell ergibt sich zu

$$\begin{aligned} \mathrm{SSE} &= \|\boldsymbol{y} - \boldsymbol{G}\boldsymbol{w}\|_2^2 \\ &= \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} - \sum_{k=1}^{P} w_k^2 \boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{g}_k \\ &= \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} - \sum_{k=1}^{P} \frac{(\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{y})^2}{\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{g}_k}\,. \end{aligned} \tag{3.60}$$

Die Aufnahme eines Terms $\boldsymbol{g}_k$ führt im Falle paarweise orthogonaler Kandidaten also zu einer Reduktion des SSE um

$$\frac{(\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{y})^2}{\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{g}_k}\,. \tag{3.61}$$

Dieser Ausdruck kann für alle Kandidaten berechnet werden. Der Kandidat mit dem größten Wert ist der erste aufzunehmende Term, der Kandidat mit dem zweitgrößten Wert ist der zweite aufzunehmende Term usw.

Nun sind die Bilder der Kandidatenterme i. Allg. aber nicht paarweise orthogonal. Ein naheliegender Ansatz ist, für die Design-Matrix $\boldsymbol{G}$ aller Kandidaten eine orthogonale Faktorisierung z.B. in der Form einer QR-Zerlegung

$$\boldsymbol{G} = \boldsymbol{Q}\boldsymbol{R}\,, \qquad \text{wobei} \quad \boldsymbol{Q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_P) \in \mathbb{R}^{N\times P}\,, \boldsymbol{R} \in \mathbb{R}^{P\times P} \tag{3.62}$$

mit einer oberen Dreiecksmatrix $\boldsymbol{R}$ und paarweise orthonormalen Vektoren $\boldsymbol{q}_i$ zu berechnen, die Modellausgabe in der Form $\hat{\boldsymbol{y}} = \boldsymbol{G}\boldsymbol{w} = \boldsymbol{Q}\boldsymbol{R}\boldsymbol{w} = \boldsymbol{Q}\boldsymbol{u}$ mit dem Koeffizientenvektor $\boldsymbol{u} = \boldsymbol{R}\boldsymbol{w}$ der Orthonormalbasis zu schreiben und die Termselektion für die $\boldsymbol{q}_i$ durchzuführen. Dann treten die oben genannten Vorzüge ein und die besten Terme können einfach anhand (3.60) bestimmt werden, indem dort die $\boldsymbol{q}_i$ für die $\boldsymbol{g}_i$ eingesetzt werden. Dieser Ansatz funktioniert allerdings i. Allg. nicht. Der Grund ist, dass bei einer QR-Zerlegung (3.62) zwar $\mathrm{span}(\boldsymbol{q}_1, \ldots, \boldsymbol{q}_j) = \mathrm{span}(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_j)\ \forall j = 1, \ldots, P$ gilt, aber es ist nicht notwendigerweise $\mathrm{span}(\boldsymbol{q}_i, \boldsymbol{q}_j) = \mathrm{span}(\boldsymbol{g}_i, \boldsymbol{g}_j)$ für beliebige i und j. Weiterhin ist $\boldsymbol{q}_k$ i. Allg. eine Linearkombination *aller* Vektoren $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k$. Wird also z.B. nur dieser eine

Term ausgewählt, so existiert unter den ursprünglichen Termen $\boldsymbol{g}_j$ nicht notwendigerweise *ein* Term, der genau den Raum span($\boldsymbol{q}_k$) aufspannt. Lediglich in dem Fall, dass es sich bei einer Auswahl von $k \leq P$ Termen tatsächlich um die ersten k Terme $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ handelt, entspricht dies der Auswahl von $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k$.

Das Problem lässt sich umgehen, indem die Orthogonalisierung der Terme simultan mit der Auswahl derselben erfolgt, und zwar derart, dass in jedem Iterationsschritt der Forward Selection alle Kandidatenterme lediglich orthogonal zu allen bereits aufgenommenen Modelltermen sind. Dies wird erreicht, indem nach Aufnahme eines Terms von allen verbleibenden Kandidaten die orthogonale Projektion auf den gerade aufgenommenen Term subtrahiert wird. Auf diese Art und Weise wird eine Orthogonalbasis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k-1}$ für die nach $k-1$ Iterationen ausgewählten Terme $\boldsymbol{g}_{i_1}, \ldots, \boldsymbol{g}_{i_{k-1}}$ *schrittweise* konstruiert. $\mathcal{I}^{(k)} = \{1, \ldots, P\} \setminus \{i_1, \ldots, i_{k-1}\}$ sei wie in (3.56) wieder die Menge der vor dem k-ten Schritt noch zur Verfügung stehenden Kandidatenterme $\{\boldsymbol{g}_i^{(k-1)} | i \in \mathcal{I}^{(k)}\}$, die orthogonal zu allen $\boldsymbol{v}_i$ sind, wobei zu Beginn $\boldsymbol{g}_i^{(0)} = \boldsymbol{g}_i \; \forall i = 1, \ldots, P$ gesetzt wurde. Im Schritt k ($1 \leq k \leq P$) wird nun der Term $\boldsymbol{v}_k = \boldsymbol{g}_{i_k}^{(k-1)}$ aufgenommen, der die größte Reduktion des quadratischen Fehlers bringt. Für diesen Term gilt analog zu (3.61)

$$\begin{aligned} \boldsymbol{v}_k &= \boldsymbol{g}_{i_k}^{(k-1)} \\ \text{mit} \qquad i_k &= \arg\max_{i \in \mathcal{I}^{(k)}} \frac{\left(\boldsymbol{y}^{\mathrm{T}} \boldsymbol{g}_i^{(k-1)}\right)^2}{\left\|\boldsymbol{g}_i^{(k-1)}\right\|_2^2}\,, \end{aligned} \tag{3.63}$$

und sein Koeffizient berechnet sich zu

$$\alpha_k = \frac{\boldsymbol{y}^{\mathrm{T}} \boldsymbol{v}_k}{\boldsymbol{v}_k^{\mathrm{T}} \boldsymbol{v}_k}\,. \tag{3.64}$$

Anschließend werden die verbleibenden Kandidaten mit den Indizes in $\mathcal{I}^{(k+1)} = \mathcal{I}^{(k)} \setminus \{i_k\}$ bzgl. $\boldsymbol{v}_k$ orthogonalisiert:

$$a_{ki} = \frac{\boldsymbol{v}_k^{\mathrm{T}} \boldsymbol{g}_i^{(k-1)}}{\boldsymbol{v}_k^{\mathrm{T}} \boldsymbol{v}_k} \tag{3.65}$$

$$\boldsymbol{g}_i^{(k)} = \boldsymbol{g}_i^{(k-1)} - a_{ki} \boldsymbol{v}_k \qquad \forall i \in \mathcal{I}^{(k+1)}\,. \tag{3.66}$$

Dabei kann es passieren, dass irgendwann $\left\|\boldsymbol{g}_i^{(k)}\right\|_2$ für einige $i \in \mathcal{I}^{(k+1)}$ sehr klein wird. Diese Kandidaten sind dann (fast) linear abhängig von den bereits ausgewählten Modelltermen und können keinen bedeutsamen Beitrag zur Reduktion des SSE liefern. Um numerische Probleme zu vermeiden, definiert man daher eine

untere Schranke $\rho > 0$ und entfernt nach jedem Iterationsschritt alle verbleibenden Kandidaten, für die $\left\|\boldsymbol{g}_i^{(k)}\right\|_2 < \rho$ gilt.

Nach M Schritten sei nun ein Abbruchkriterium erfüllt. Nach Umnummerierung der Terme durch Vertauschung der Indizes i_k und k für $k = 1, \ldots, M$ haben die ausgewählten Terme die Reihenfolge $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_M$. Für die Design-Matrix $\boldsymbol{G} = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_M)$ liegt dann eine orthogonale Faktorisierung

$$\boldsymbol{G} = \boldsymbol{V}\boldsymbol{A} \tag{3.67}$$

vor, wobei $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_M)$ aus den orthogonalisierten Modelltermen (3.63) besteht und

$$\boldsymbol{A} = \begin{pmatrix} 1 & a_{12} & \ldots & a_{1M} \\ & 1 & & \vdots \\ & & \ddots & a_{M-1,M} \\ 0 & & & 1 \end{pmatrix} \tag{3.68}$$

die obere Dreiecksmatrix mit Einsen auf der Diagonale ist, deren übrige Einträge die den ausgewählten Modelltermen entsprechenden Koeffizienten aus (3.65) sind. Die Modellausgabe ist dann

$$\hat{\boldsymbol{y}} = \boldsymbol{G}\boldsymbol{w} = \boldsymbol{V}(\boldsymbol{A}\boldsymbol{w}) = \boldsymbol{V}\boldsymbol{\alpha}\,, \tag{3.69}$$

und die Koeffizienten $\boldsymbol{w}$ der ursprünglichen Terme $\boldsymbol{g}_i$ können somit durch Rückwärtseinsetzen aus

$$\boldsymbol{A}\boldsymbol{w} = \boldsymbol{\alpha} \tag{3.70}$$

gewonnen werden. Dieses Schema führt also letztlich dazu, dass eine orthogonale Faktorisierung (3.67) nur für die Design-Matrix der M ausgewählten Modellterme durchgeführt wird und nicht für den gesamten Kandidatenpool. Bei einem sehr großen Pool und $M \ll P$ erfordert dies einen deutlich geringeren Rechenaufwand. Das Verfahren wird als *Forward Orthogonal Regression* (FOR) oder Forward *Orthogonal Least Squares* (OLS) bezeichnet und z.B. in [16, 27, 49] beschrieben. Chen et al. zeigen in [16], wie sich das FOR-Verfahren mit Hilfe von Householder-Transformationen und dem klassischen sowie dem modifizierten Gram-Schmidt-Verfahren realisieren lässt. Von Korenberg stammt eine *Fast Orthogonal Search* (FOS) genannte Variante der Forward Selection, die den Rechenaufwand noch einmal deutlich reduziert, indem die orthogonalen Basisvektoren $\boldsymbol{v}_k$ in (3.67) nicht explizit berechnet werden, sondern nur die obere Dreiecksmatrix [50, 51].

Dies wird durch die schrittweise Konstruktion einer Cholesky-Faktorisierung der Koeffizientenmatrix $\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}$ der Normalengleichungen (3.58) für die ausgewählten Modellterme erreicht. Während Korenberg hierfür in [50] noch das klassische Gram-Schmidt-Verfahren (CGS) verwendet, beschreibt er in [52] eine auf dem modifizierten Gram-Schmidt-Verfahren (MGS) basierende FOS-Variante, die wegen dessen größerer numerischer Robustheit weniger anfällig für Rundungsfehler ist [46].

Sowohl das OLS- wie auch das FOS-Verfahren verwenden als Kostenfunktion den quadratischen Fehler, arbeiten also ohne Regularisierung. Orr [53] verwendet eine Kombination von Forward Selection und Ridge Regression zur Vermeidung von Overfitting bei der Termselektion zur Konstruktion von RBF-Modellen, die jedoch nicht auf einer Orthogonalisierung der Design-Matrix beruht und daher einen wesentlich größeren Rechenaufwand erfordert als die OLS-Methode. Eine effizientere Variante, die den oben beschriebenen OLS-Algorithmus mit der Ridge-Regression verknüpft, wurde von Chen et al. entwickelt [54] und *Regularized Orthogonal Least Squares* (ROLS) genannt. Um die schrittweise Orthogonalisierung durchführen zu können, wendet Chen die Ridge-Regression nicht auf die Koeffizienten $\boldsymbol{w}$ der ursprünglichen Modellterme $\boldsymbol{g}_i$ an, sondern auf die Koeffizienten $\boldsymbol{\alpha}$ der orthogonalisierten Vektoren $\boldsymbol{v}_i$ (3.69), die über die lineare Beziehung (3.70) miteinander verknüpft sind. Die in jedem Schritt zu minimierende Kostenfunktion lautet anstelle von (3.45) dann

$$\mathrm{SSE}_{\mathrm{RR}} = \|\boldsymbol{y} - \boldsymbol{V}\boldsymbol{\alpha}\|_2^2 + N\lambda\|\boldsymbol{\alpha}\|_2^2 \,. \tag{3.71}$$

Über die Beziehung (3.70) wirkt sich der regularisierende Effekt auch auf die ursprünglichen Koeffizienten $\boldsymbol{w}$ aus. Analog zur mittleren Gleichung in (3.60) verringert sich der Wert von (3.71) nach Aufnahme eines Terms $\boldsymbol{v}_k$ um $\alpha_k^2(\boldsymbol{v}_k^{\mathrm{T}}\boldsymbol{v}_k + N\lambda)$ (für Details siehe [54]).

Diese Variante ist zwar deutlich schneller als die Methode von Orr in [53], erfordert aber immer noch die explizite Berechnung der orthogonalen Basisvektoren $\boldsymbol{v}_i$ und ist damit wesentlich rechenaufwändiger als der ohne Regularisierung arbeitende Fast Orthogonal Search-Algorithmus von Korenberg. Im Rahmen dieser Arbeit wurde eine Möglichkeit gefunden, den FOS-Algorithmus um die Ridge-Regression zu erweitern, wobei die Regularisierung hier direkt auf die Koeffizienten $\boldsymbol{w}$ angewendet wird. Die Herleitung orientiert sich an der in [51] dargestellten Ableitung des originalen FOS-Algorithmus.

Es seien also die Trainingsdaten $\mathcal{D} = \{(\boldsymbol{x}_t, y_t)|t = 1, \ldots, N\}$ gegeben sowie der Pool von Kandidatentermen $\{\boldsymbol{g}_j|j = 1, \ldots, M\}$. Weiterhin sei $\boldsymbol{G}_k = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k) \in \mathbb{R}^{N\times k}$ die Designmatrix nach dem k-ten Iterationsschritt (evtl. nach Umnummerierung der Terme zur Vereinfachung der Notation, d.h. wird im ersten Iterationsschritt z.B. der Term $\boldsymbol{g}_{12}$ ausgewählt, so wird die Nummerierung der Terme $\boldsymbol{g}_1$ und $\boldsymbol{g}_{12}$ vertauscht, so dass $\boldsymbol{g}_1$ der ausgewählte Term ist und allgemein nach j Iterationsschritten die Terme $\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_j\}$ ausgewählt sind). Die linearen Koeffizienten $\boldsymbol{w}_k$ werden so bestimmt, dass analog zu (3.45) die regularisierte Kostenfunktion

$$\mathrm{SSE}_{\mathrm{RR}}^{(k)} = \|\boldsymbol{y} - \boldsymbol{G}_k\boldsymbol{w}_k\|_2^2 + N\lambda\|\boldsymbol{w}_k\|_2^2 = \|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{G}}_k\boldsymbol{w}_k\|_2^2 \tag{3.72}$$

für ein fest gewähltes $\lambda \geq 0$ minimiert wird. $\tilde{\boldsymbol{G}}_k$ und $\tilde{\boldsymbol{y}}$ sind wie in (3.46) definiert. Mit der Definition

$$\tilde{\boldsymbol{e}}_k = \tilde{\boldsymbol{y}} - \tilde{\boldsymbol{G}}_k\boldsymbol{w}_k \tag{3.73}$$

ist

$$\mathrm{SSE}_{\mathrm{RR}}^{(k)} = \|\tilde{\boldsymbol{e}}_k\|_2^2 \,. \tag{3.74}$$

Eine Lösung $\boldsymbol{w}_k$, die (3.72) minimiert, erfüllt die Normalengleichungen

$$\begin{aligned} & \tilde{\boldsymbol{G}}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_k\boldsymbol{w}_k = \tilde{\boldsymbol{G}}_k^{\mathrm{T}}\tilde{\boldsymbol{y}} \\ \Leftrightarrow \quad & \left(\boldsymbol{G}_k^{\mathrm{T}}\boldsymbol{G}_k + N\lambda \cdot \mathbb{1}_k\right)\boldsymbol{w}_k = \boldsymbol{G}_k^{\mathrm{T}}\boldsymbol{y} \,. \end{aligned} \tag{3.75}$$

Ein Term wird natürlich nur dann ins Modell aufgenommen, wenn er eine hinreichende Verringerung des Werts der Kostenfunktion (3.72) bewirkt. Deshalb sind alle Spalten von $\boldsymbol{G}_k$ paarweise linear unabhängig voneinander, so dass $\boldsymbol{G}_k$ und damit auch $\tilde{\boldsymbol{G}}_k$ maximalen Rang k haben. Daraus folgt aber, dass die symmetrische Matrix $\tilde{\boldsymbol{G}}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_k$ positiv definit ist und somit eine Cholesky-Zerlegung besitzt [39]:

$$\tilde{\boldsymbol{G}}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_k = \boldsymbol{G}_k^{\mathrm{T}}\boldsymbol{G}_k + N\lambda \cdot \mathbb{1}_k = \boldsymbol{R}_k^{\mathrm{T}}\boldsymbol{R}_k \tag{3.76}$$

mit der oberen Dreiecksmatrix $\boldsymbol{R}_k \in \mathbb{R}^{k\times k}$. Einsetzen der Cholesky-Zerlegung (3.76) in die Normalengleichungen (3.75) ergibt

$$\begin{aligned} & \boldsymbol{R}_k^{\mathrm{T}}\boldsymbol{R}_k\boldsymbol{w}_k = \boldsymbol{G}_k^{\mathrm{T}}\boldsymbol{y} \\ \Leftrightarrow \quad & \boldsymbol{R}_k^{\mathrm{T}}\boldsymbol{z}_k = \boldsymbol{G}_k^{\mathrm{T}}\boldsymbol{y} \quad \text{mit} \quad \boldsymbol{z}_k = \boldsymbol{R}_k\boldsymbol{w}_k \,. \end{aligned} \tag{3.77}$$

Die erste Gleichung der zweiten Zeile kann leicht durch Vorwärtseinsetzen nach

$\boldsymbol{z}_k$ gelöst werden und die zweite Gleichung bei bekanntem $\boldsymbol{z}_k$ dann durch Rückwärtseinsetzen nach $\boldsymbol{w}_k$.

Zunächst soll untersucht werden, welchen Einfluss die Aufnahme eines weiteren Terms aus dem Pool auf den Wert der zu minimierenden Kostenfunktion (3.72) hat, um daraus ein Kriterium für die Auswahl des Terms abzuleiten. Im Schritt $k+1$ werde also o.B.d.A. der Term $\boldsymbol{g}_{k+1}$ zum Modell hinzu genommen. Die Design-Matrix ist dann $\boldsymbol{G}_{k+1} = \begin{pmatrix} \boldsymbol{G}_k & \boldsymbol{g}_{k+1} \end{pmatrix}$ und damit

$$\begin{aligned} \tilde{\boldsymbol{G}}_{k+1}^{\mathrm{T}} \tilde{\boldsymbol{G}}_{k+1} &= \boldsymbol{G}_{k+1}^{\mathrm{T}} \boldsymbol{G}_{k+1} + N\lambda \cdot \mathbb{1}_{k+1} \\ &= \begin{pmatrix} \boldsymbol{G}_k^{\mathrm{T}} & \sqrt{N\lambda} \cdot \mathbb{1}_k & \boldsymbol{0} \\ \boldsymbol{g}_{k+1}^{\mathrm{T}} & \boldsymbol{0}^{\mathrm{T}} & \sqrt{N\lambda} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{G}_k & \boldsymbol{g}_{k+1} \\ \sqrt{N\lambda} \cdot \mathbb{1}_k & \boldsymbol{0} \\ \boldsymbol{0}^{\mathrm{T}} & \sqrt{N\lambda} \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{G}_k^{\mathrm{T}} \boldsymbol{G}_k + N\lambda \cdot \mathbb{1}_k & \boldsymbol{G}_k^{\mathrm{T}} \boldsymbol{g}_{k+1} \\ \boldsymbol{g}_{k+1}^{\mathrm{T}} \boldsymbol{G}_k & \boldsymbol{g}_{k+1}^{\mathrm{T}} \boldsymbol{g}_{k+1} + N\lambda \end{pmatrix} . \end{aligned} \tag{3.78}$$

Der Cholesky-Faktor $\boldsymbol{R}_{k+1}$ geht aus $\boldsymbol{R}_k$ durch Erweiterung um einen Vektor $\boldsymbol{r}_{k+1}$ und einen Skalar ρ_{k+1} hervor, so dass

$$\boldsymbol{R}_{k+1} = \begin{pmatrix} \boldsymbol{R}_k & \boldsymbol{r}_{k+1} \\ \boldsymbol{0} & \rho_{k+1} \end{pmatrix} , \tag{3.79}$$

denn dann ist

$$\begin{aligned} \boldsymbol{R}_{k+1}^{\mathrm{T}} \boldsymbol{R}_{k+1} &= \begin{pmatrix} \boldsymbol{R}_k^{\mathrm{T}} & \boldsymbol{0} \\ \boldsymbol{r}_{k+1}^{\mathrm{T}} & \rho_{k+1} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{R}_k & \boldsymbol{r}_{k+1} \\ \boldsymbol{0} & \rho_{k+1} \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{R}_k^{\mathrm{T}} \boldsymbol{R}_k & \boldsymbol{R}_k^{\mathrm{T}} \boldsymbol{r}_{k+1} \\ \boldsymbol{r}_{k+1}^{\mathrm{T}} \boldsymbol{R}_k & \boldsymbol{r}_{k+1}^{\mathrm{T}} \boldsymbol{r}_{k+1} + \rho_{k+1}^2 \end{pmatrix} , \end{aligned} \tag{3.80}$$

und Gleichsetzen von (3.78) und (3.80) ergibt für die Cholesky-Faktorisierung $\tilde{\boldsymbol{G}}_{k+1}^{\mathrm{T}} \tilde{\boldsymbol{G}}_{k+1} = \boldsymbol{R}_{k+1}^{\mathrm{T}} \boldsymbol{R}_{k+1}$ für das um $\boldsymbol{g}_{k+1}$ erweiterte Modell

$$\begin{pmatrix} \boldsymbol{G}_k^{\mathrm{T}} \boldsymbol{G}_k + N\lambda \cdot \mathbb{1}_k & \boldsymbol{G}_k^{\mathrm{T}} \boldsymbol{g}_{k+1} \\ \boldsymbol{g}_{k+1}^{\mathrm{T}} \boldsymbol{G}_k & \boldsymbol{g}_{k+1}^{\mathrm{T}} \boldsymbol{g}_{k+1} + N\lambda \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}_k^{\mathrm{T}} \boldsymbol{R}_k & \boldsymbol{R}_k^{\mathrm{T}} \boldsymbol{r}_{k+1} \\ \boldsymbol{r}_{k+1}^{\mathrm{T}} \boldsymbol{R}_k & \boldsymbol{r}_{k+1}^{\mathrm{T}} \boldsymbol{r}_{k+1} + \rho_{k+1}^2 \end{pmatrix} . \tag{3.81}$$

Zunächst sei angenommen, dass $\boldsymbol{r}_{k+1}$ und ρ_{k+1} für alle Terme aus dem Kandidatenpool bekannt sind. Die Cholesky-Zerlegung eingesetzt in die Normalenglei-

chungen im $(k+1)$-ten Schritt ergibt nun

$$\begin{aligned} & \boldsymbol{R}_{k+1}^{\mathrm{T}}\boldsymbol{R}_{k+1}\boldsymbol{w}_{k+1} = \boldsymbol{G}_{k+1}^{\mathrm{T}}\boldsymbol{y} \\ \Leftrightarrow \quad & \begin{pmatrix} \boldsymbol{R}_k^{\mathrm{T}} & \boldsymbol{0} \\ \boldsymbol{r}_{k+1}^{\mathrm{T}} & \rho_{k+1} \end{pmatrix} \boldsymbol{z}_{k+1} = \begin{pmatrix} \boldsymbol{G}_k^{\mathrm{T}}\boldsymbol{y} \\ \boldsymbol{g}_{k+1}^{\mathrm{T}}\boldsymbol{y} \end{pmatrix} \\ & \text{mit} \quad \boldsymbol{z}_{k+1} = \begin{pmatrix} \boldsymbol{R}_k & \boldsymbol{r}_{k+1} \\ \boldsymbol{0} & \rho_{k+1} \end{pmatrix} \boldsymbol{w}_{k+1}\,. \end{aligned} \tag{3.82}$$

Wie in (3.77) kann $\boldsymbol{z}_{k+1}$ durch Vorwärtseinsetzen und damit dann $\boldsymbol{w}_{k+1}$ durch Rückwärtseinsetzen berechnet werden. Vorwärtseinsetzen in der mittleren Gleichung von (3.82) ergibt

$$\boldsymbol{z}_{k+1} = \begin{pmatrix} \boldsymbol{z}_k \\ \frac{1}{\rho_{k+1}}(\boldsymbol{g}_{k+1}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{r}_{k+1}^{\mathrm{T}}\boldsymbol{z}_k) \end{pmatrix} = \begin{pmatrix} \boldsymbol{z}_k \\ \frac{\alpha_{k+1}}{\rho_{k+1}} \end{pmatrix} \tag{3.83}$$

mit

$$\alpha_{k+1} := \boldsymbol{g}_{k+1}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{r}_{k+1}^{\mathrm{T}}\boldsymbol{z}_k\,. \tag{3.84}$$

Der Vektor $\boldsymbol{z}_k$ ist als Lösung von (3.77) schon aus dem vorigen Iterationsschritt bekannt. Die ersten k Komponenten von $\boldsymbol{z}_{k+1}$ hängen also gar nicht vom gewählten Term $\boldsymbol{g}_{k+1}$ ab, und $\boldsymbol{z}_{k+1}$ entsteht durch Erweiterung von $\boldsymbol{z}_k$ um die skalare Größe α_{k+1}/ρ_{k+1}. Aus den Cholesky-Faktorisierungen im k-ten Schritt (3.77) und im $(k+1)$-ten Schritt (3.82) und mit (3.83) gilt für die Kostenfunktion (3.72) nach Aufnahme des Terms $\boldsymbol{g}_{k+1}$

$$\begin{aligned} \mathrm{SSE}_{\mathrm{RR}}^{(k+1)} &= (\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{G}}_{k+1})^{\mathrm{T}}(\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{G}}_{k+1}) \\ &= \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{w}_{k+1}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{k+1}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{k+1}\boldsymbol{w}_{k+1} \\ &= \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{z}_{k+1}^{\mathrm{T}}\boldsymbol{z}_{k+1} \\ &= \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{z}_k{}^{\mathrm{T}}\boldsymbol{z}_k - \left(\frac{\alpha_{k+1}}{\rho_{k+1}}\right)^2\,. \end{aligned} \tag{3.85}$$

Dieser Ausdruck liefert ein Kriterium für die Auswahl des besten Terms. Seien $\{\boldsymbol{g}_j | j = k+1, \ldots, P\}$ die nach Abschluss des k-ten Schrittes verbleibenden Kandidatenterme und seien $\boldsymbol{r}_j^{(k+1)}$, $\alpha_j^{(k+1)}$ und $\rho_j^{(k+1)}$ die zugehörigen, in (3.78) bzw. (3.84) eingeführten und für alle Kandidaten bekannten Hilfsgrößen. Dann führt im $(k+1)$-ten Schritt der Term $\boldsymbol{g}_{j_{k+1}}$ zum kleinsten quadratischen Fehler, für den

der Quotient in (3.85) maximal ist, für den also gilt

$$j_{k+1} = \arg \max_{j=k+1,\ldots,P} \left(\frac{\alpha_j^{(k+1)}}{\rho_j^{(k+1)}} \right)^2 . \tag{3.86}$$

Durch Umnummerierung kann wieder erreicht werden, dass $\boldsymbol{g}_{k+1}$ der aufgenommene Term ist und $\boldsymbol{g}_{k+2}, \ldots, \boldsymbol{g}_P$ die verbleibenden Kandidaten. Nun muss noch bestimmt werden, wie die aktualisierten, für den nächsten Schritt $k+2$ gültigen Werte $\boldsymbol{r}_j^{(k+2)}$, $\alpha_j^{(k+2)}$ und $\rho_j^{(k+2)}$ der Hilfsgrößen der verbleibenden Kandidaten aus ihren Werten im $(k+1)$-ten Schritt hervorgehen. Für $\boldsymbol{r}_j^{(k+2)}$ und $\rho_j^{(k+2)}$ muss wieder die Cholesky-Zerlegung von $(\boldsymbol{G}_{k+1}\ \boldsymbol{g}_j)^{\mathrm{T}} \cdot (\boldsymbol{G}_{k+1}\ \boldsymbol{g}_j) + N\lambda \cdot \mathbb{1}_{k+2}$ gelten:

$$\begin{aligned}
&\begin{pmatrix} \boldsymbol{G}_{k+1}^{\mathrm{T}} \\ \boldsymbol{g}_j^{\mathrm{T}} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{G}_{k+1} & \boldsymbol{g}_j \end{pmatrix} + N\lambda \cdot \mathbb{1}_{k+2} \\
&\qquad = \begin{pmatrix} \boldsymbol{G}_{k+1}^{\mathrm{T}}\boldsymbol{G}_{k+1} + N\lambda\mathbb{1}_{k+1} & \boldsymbol{G}_{k+1}^{\mathrm{T}}\boldsymbol{g}_j \\ \boldsymbol{g}_j^{\mathrm{T}}\boldsymbol{G}_{k+1} & \boldsymbol{g}_j^{\mathrm{T}}\boldsymbol{g}_j + N\lambda \end{pmatrix} \\
&\qquad \stackrel{!}{=} \begin{pmatrix} \boldsymbol{R}_{k+1}^{\mathrm{T}} & \boldsymbol{0} \\ (\boldsymbol{r}_j^{(k+2)})^{\mathrm{T}} & \rho_j^{(k+2)} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{R}_{k+1} & \boldsymbol{r}_j^{(k+2)} \\ \boldsymbol{0} & \rho_j^{(k+2)} \end{pmatrix} \\
&\qquad = \begin{pmatrix} \boldsymbol{R}_{k+1}^{\mathrm{T}}\boldsymbol{R}_{k+1} & \boldsymbol{R}_{k+1}^{\mathrm{T}}\boldsymbol{r}_j^{(k+2)} \\ (\boldsymbol{r}_j^{(k+2)})^{\mathrm{T}}\boldsymbol{R}_{k+1} & (\boldsymbol{r}_j^{(k+2)})^{\mathrm{T}}\boldsymbol{r}_j^{(k+2)} + (\rho_j^{(k+2)})^2 \end{pmatrix} .
\end{aligned} \tag{3.87}$$

Damit ergibt sich $\boldsymbol{r}_j^{(k+2)}$ als Lösung von $\boldsymbol{R}_{k+1}^{\mathrm{T}}\boldsymbol{r}_j^{(k+2)} = \boldsymbol{G}_{k+1}^{\mathrm{T}}\boldsymbol{g}_j$, d.h. es gilt

$$\begin{pmatrix} \boldsymbol{R}_k^{\mathrm{T}} & \boldsymbol{0} \\ \boldsymbol{r}_{k+1}^{\mathrm{T}} & \rho_{k+1} \end{pmatrix} \boldsymbol{r}_j^{(k+2)} = \begin{pmatrix} \boldsymbol{G}_k^{\mathrm{T}} \\ \boldsymbol{g}_{k+1}^{\mathrm{T}} \end{pmatrix} \boldsymbol{g}_j \, . \tag{3.88}$$

Im Schritt k gilt die Cholesky-Zerlegung (3.78) mit $\boldsymbol{g}_j$, $\boldsymbol{r}_j^{(k)}$ und $\rho_j^{(k)}$ statt $\boldsymbol{g}_{k+1}$, $\boldsymbol{r}_{k+1}$ und ρ_{k+1}, d.h. $\boldsymbol{r}_j^{(k)}$ ist die aus dem k-ten Schritt bekannte Lösung von $\boldsymbol{R}_k^{\mathrm{T}}\boldsymbol{r}_j^{(k)} = \boldsymbol{G}_k^{\mathrm{T}}\boldsymbol{g}_j$. Mittels Vorwärtseinsetzen in (3.88) ergibt sich

$$\boldsymbol{r}_j^{(k+2)} = \begin{pmatrix} \boldsymbol{r}_j^{(k+1)} \\ \frac{1}{\rho_{k+1}} \left(\boldsymbol{g}_{k+1}^{\mathrm{T}}\boldsymbol{g}_j - \boldsymbol{r}_{k+1}^{\mathrm{T}}\boldsymbol{r}_j^{(k+1)} \right) \end{pmatrix} =: \begin{pmatrix} \boldsymbol{r}_j^{(k+1)} \\ \beta_j \end{pmatrix} \tag{3.89}$$

mit

$$\beta_j := \frac{1}{\rho_{k+1}} \left(\boldsymbol{g}_{k+1}^{\mathrm{T}}\boldsymbol{g}_j - \boldsymbol{r}_{k+1}^{\mathrm{T}}\boldsymbol{r}_j^{(k+1)} \right) . \tag{3.90}$$

Algorithmus 3.1 Fast Orthogonal Search mit Ridge-Regression

Require: Kandidatenpool $\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_P\} \in \mathbb{R}^N$, Modellierungsziel $\boldsymbol{y} \in \mathbb{R}^N$, Regularisierungsparameter $\lambda \geq 0$, Indexmenge $\mathcal{J} = \{1, \ldots, P\}$ der Kandidatenterme.

1: {Initialisierung und 1. Schritt:}
2: $\forall j = 1, \ldots, P: \ \alpha_j := \boldsymbol{g}_j^{\mathrm{T}} \boldsymbol{y}, \ \rho_j^2 := \boldsymbol{g}_j^{\mathrm{T}} \boldsymbol{g}_j + N\lambda$
3: Bestimme $k = \max\limits_{j=1,\ldots,P} \alpha_j^2/\rho_j^2$ als Index des ersten ausgewählten Terms.
4: Setze $R = \rho_k$, $\mathcal{J} \leftarrow \mathcal{J} \setminus \{k\}$.
5: $\forall j \in \mathcal{J}: \ \beta_j = (\boldsymbol{g}_k^{\mathrm{T}} \boldsymbol{g}_j)/\rho_k, \ \boldsymbol{r}_j = \beta_j, \ \rho_j^2 \leftarrow \rho_j^2 - \beta_j^2, \ \alpha_j \leftarrow \alpha_j - (\beta_j \alpha_k/\rho_k)$.
6: {Schritte 2, 3, ...:}
7: **while** (noch kein Abbruchkriterium erfüllt) **do**
8: Bestimme $k = \max\limits_{j \in \mathcal{J}} \alpha_j^2/\rho_j^2$ {Index des nächsten Terms}
9: {Update der Hilfsgrößen:}
10: $R \leftarrow \begin{pmatrix} R & \boldsymbol{r}_k \\ 0 & \rho_k \end{pmatrix}$ {Update des Cholesky-Faktors nach (3.79)}
11: $\mathcal{J} \leftarrow \mathcal{J} \setminus \{k\}$ {Entferne neuen Modellterm aus dem Kandidatenpool}
12: **for** $j \in \mathcal{J}$ **do**
13: $\beta_j = (\boldsymbol{g}_k^{\mathrm{T}} \boldsymbol{g}_j - \boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{r}_j)/\rho_k$ {Definition von β_j nach (3.90)}
14: $\boldsymbol{r}_j \leftarrow \begin{pmatrix} \boldsymbol{r}_j \\ \beta_j \end{pmatrix}$ {Update von $\boldsymbol{r}_j$ nach (3.89)}
15: $\rho_j^2 \leftarrow \rho_j^2 - \beta_j^2$ {Update von ρ_j^2 nach (3.92)}
16: $\alpha_j \leftarrow \alpha_j - \beta_j \alpha_k/\rho_k$ {Update von α_j nach (3.96)}
17: **end for**
18: {Überprüfe mögliche Abbruchbedingungen.}
19: **end while**

Aus (3.87) ergibt sich $\rho_j^{(k+2)}$, denn es gilt

$$(\rho_j^{(k+2)})^2 = \boldsymbol{g}_j^{\mathrm{T}} \boldsymbol{g}_j - (\boldsymbol{r}_j^{(k+2)})^{\mathrm{T}} \boldsymbol{r}_j^{(k+2)} + N\lambda \overset{(3.89)}{=} \boldsymbol{g}_j^{\mathrm{T}} \boldsymbol{g}_j - (\boldsymbol{r}_j^{(k+1)})^{\mathrm{T}} \boldsymbol{r}_j^{(k+1)} + N\lambda - \beta_j^2 \,. \tag{3.91}$$

Nach (3.81) gilt aber $(\rho_j^{(k+1)})^2 = \boldsymbol{g}_j^{\mathrm{T}} \boldsymbol{g}_j - (\boldsymbol{r}_j^{(k+1)})^{\mathrm{T}} \boldsymbol{r}_j^{(k+1)} + N\lambda$, und damit erhält man schließlich eine rekursive Formel für $\rho_j^{(k+2)}$:

$$(\rho_j^{(k+2)})^2 = (\rho_j^{(k+1)})^2 - \beta_j^2 \,. \tag{3.92}$$

Schließlich ergibt sich $\alpha_j^{(k+2)}$ aus der Definition von α_j in (3.84):

$$\alpha_j^{(k+1)} = \boldsymbol{g}_j^{\mathrm{T}} \boldsymbol{y} - \left(\boldsymbol{r}_j^{(k+1)}\right)^{\mathrm{T}} \boldsymbol{z}_k \tag{3.93}$$

$$\Rightarrow \quad \alpha_j^{(k+2)} = \boldsymbol{g}_j^{\mathrm{T}} \boldsymbol{y} - \left(\boldsymbol{r}_j^{(k+2)}\right)^{\mathrm{T}} \boldsymbol{z}_{k+1} \tag{3.94}$$

$$= \boldsymbol{g}_j^{\mathrm{T}} \boldsymbol{y} - \left(\left(\boldsymbol{r}_j^{(k+1)}\right)^{\mathrm{T}} \quad \beta_j\right) \cdot \begin{pmatrix} \boldsymbol{z}_k \\ \frac{\alpha_{k+1}}{\rho_{k+1}} \end{pmatrix} \tag{3.95}$$

$$= \alpha_j^{(k+1)} - \frac{\beta_j \alpha_{k+1}}{\rho_{k+1}} \,. \tag{3.96}$$

Eine Formulierung des regularisierten FOS-Verfahrens in Pseudocode ist in Alg. 3.1 angegeben. Nach Abbruch der Iteration bestehe das Modell aus $M \leq P$ Termen. Der Koeffizientenvektor $\boldsymbol{w}_M$ kann dann per Vorwärts- und anschließendem Rückwärtseinsetzen analog zu (3.77) aus

$$\boldsymbol{R}_M^{\mathrm{T}} \boldsymbol{R}_M \boldsymbol{w}_M = \boldsymbol{G}_M^{\mathrm{T}} \boldsymbol{y} \tag{3.97}$$

bestimmt werden.

3.3.2 Backward Elimination

Bei der Backward Elimination wird mit einem großen Modell mit vielen Termen begonnen, aus dem nacheinander irrelevante Terme entfernt werden. Der Algorithmus ist wie bei der Forward Selection iterativer Natur und verwendet die gleiche Kostenfunktion (3.15) bzw. (3.45). Ausgehend von dem vollen Modell wird in jedem Iterationsschritt der Term entfernt, der zum geringsten Anstieg der Kostenfunktion führt. Dies wird so lange wiederholt, bis ein Abbruchkriterium erfüllt ist. Auch für die Backward Elimination existiert ein effizienter Algorithmus, der aber auf eine (implizite) Orthogonalisierung der Modellterme verzichtet, da dies hier keine Vorteile bringt. Die Argumentation ist die gleiche wie im vorigen Abschnitt bei der Forward Selection, wo dargelegt wurde, warum es keinen Vorteil bringt, eine orthogonale Faktorisierung des kompletten Kandidatenpools zu berechnen: Hat man nämlich eine solche z.B. in Form einer QR-Zerlegung $\boldsymbol{G} = \boldsymbol{Q}\boldsymbol{R}$ der Design-Matrix des aus allen P Termen bestehenden Modells vorliegen, so kann man zwar wie in (3.61) einfach entscheiden, welcher der orthogonalen Terme $\boldsymbol{q}_i$ bei ihrer Entfernung zum geringsten Anstieg des SSE führen, aber dies bringt nur dann einen Vorteil, wenn es sich dabei um den letzten Term $\boldsymbol{g}_P$ handelt. Wegen

$\text{span}(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k) = \text{span}(\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k)\ \forall k = 1, \ldots, P$ und weil $\boldsymbol{q}_k$ i. Allg. eine Linearkombination *aller* Spalten $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k$ ist, entspricht zwar die Entfernung des Terms $\boldsymbol{q}_P$ aus dem Modell der Entfernung von $\boldsymbol{g}_P$, aber die Entfernung von $\boldsymbol{q}_k$ mit $k < P$ entspricht i. Allg. nicht der Entfernung *eines* Terms $\boldsymbol{g}_j$.

Von Reeves [55] stammt ein effizientes, *Backward Greedy Algorithm* genanntes Verfahren für die Backward Elimination, das anstelle einer orthogonalen Faktorisierung die Formel für die Inverse einer Block-Matrix verwendet (siehe z.B. [56]) und die Auswirkung der Entfernung eines Terms direkt aus der Aktualisierung der Lösung der Normalengleichungen bestimmt. Im Rahmen dieser Arbeit wurde der Algorithmus von Reeves um die Ridge-Regression erweitert. Die im Folgenden gebrachte Herleitung dieser erweiterten Backward Elimination orientiert sich an [55]. Es sei ein Modell mit P Termen gegeben, die Design-Matrix für die Trainingsdaten sei $\boldsymbol{G}_P = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_P) \in \mathbb{R}^{N \times P}$ mit $\text{rang}(\boldsymbol{G}_P) = P$, und der Regularisierungsparameter $\lambda \geq 0$ sei fest gewählt. Mit der Notation (3.46) gilt für die Kostenfunktion

$$\text{SSE}_{\text{RR}}^{(P)} = \|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{G}}_P \boldsymbol{w}_P\|_2^2 = \|\boldsymbol{y} - \boldsymbol{G}_P \boldsymbol{w}_P\|_2^2 + N\lambda \|\boldsymbol{w}_P\|_2^2 \,. \tag{3.98}$$

Für die eindeutige Lösung $\boldsymbol{w}_P^\star$ des Minimums der Kostenfunktion gilt

$$\boldsymbol{w}_P^\star = (\tilde{\boldsymbol{G}}_P^{\mathrm{T}} \tilde{\boldsymbol{G}}_P)^{-1} \tilde{\boldsymbol{G}}_P^{\mathrm{T}} \tilde{\boldsymbol{y}} \,. \tag{3.99}$$

Einsetzen dieser Lösung in die Kostenfunktion (3.98) ergibt

$$\begin{aligned} \text{SSE}_{\text{RR}}^{(P)} &= \|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{G}}_P (\tilde{\boldsymbol{G}}_P^{\mathrm{T}} \tilde{\boldsymbol{G}}_P)^{-1} \tilde{\boldsymbol{G}}_P^{\mathrm{T}} \tilde{\boldsymbol{y}}\|_2^2 && (3.100) \\ &= \tilde{\boldsymbol{y}}^{\mathrm{T}} \tilde{\boldsymbol{y}} - \tilde{\boldsymbol{y}}^{\mathrm{T}} \tilde{\boldsymbol{G}}_P (\tilde{\boldsymbol{G}}_P^{\mathrm{T}} \tilde{\boldsymbol{G}}_P)^{-1} \tilde{\boldsymbol{G}}_P^{\mathrm{T}} \tilde{\boldsymbol{y}} \,. && (3.101) \end{aligned}$$

Nur der zweite Term in (3.101) hängt also von $\tilde{\boldsymbol{G}}_P$ ab. Die Bestimmung des Terms, dessen Entfernung zum geringsten Anstieg der Kostenfunktion führt, ist damit gleichbedeutend zur Suche des Terms, nach dessen Entfernung der zweite Term in (3.101) maximal ist.

Es sei nun ein Zustand erreicht, in dem noch $i \leq P$ Terme im Modell vorhanden sind. Die erweiterte Design-Matrix dieses Modells sei $\tilde{\boldsymbol{G}} \in \mathbb{R}^{(N+i) \times i}$. Man betrachtet jetzt den Effekt auf den Wert der Kostenfunktion $\text{SSE}_{\text{RR}}^{(i)}$, wenn der Term $\boldsymbol{g}_k$ aus dem Modell entfernt wird. Durch eine geeignete Permutationsmatrix $\boldsymbol{\Pi}$ kann

die Spalte $\tilde{\boldsymbol{g}}_k$ an die letzte Position von $\tilde{\boldsymbol{G}}$ gebracht werden:

$$\tilde{\boldsymbol{G}} = \begin{pmatrix} \tilde{\boldsymbol{G}}_{-k} & \tilde{\boldsymbol{g}}_k \end{pmatrix} \boldsymbol{\Pi}^{\mathrm{T}} . \tag{3.102}$$

Dabei ist $\tilde{\boldsymbol{G}}_{-k}$ die Matrix, die aus $\tilde{\boldsymbol{G}}$ durch Entfernen der k-ten Spalte entsteht. Damit gilt

$$(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1} = \boldsymbol{\Pi} \begin{pmatrix} \tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k} & \tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{g}}_k \\ \tilde{\boldsymbol{g}}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k} & \tilde{\boldsymbol{g}}_k^{\mathrm{T}}\tilde{\boldsymbol{g}}_k \end{pmatrix} \boldsymbol{\Pi}^{\mathrm{T}} . \tag{3.103}$$

Nun werden die Matrizen $(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}$ und $(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}\tilde{\boldsymbol{G}}^{\mathrm{T}}$ folgendermaßen partitioniert:

$$(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1} =: \boldsymbol{\Pi} \begin{pmatrix} \boldsymbol{A}_k & \boldsymbol{a}_k \\ \boldsymbol{a}_k^{\mathrm{T}} & \gamma_k \end{pmatrix} \boldsymbol{\Pi}^{\mathrm{T}} \quad \text{mit} \quad \begin{cases} \boldsymbol{A}_k \in \mathbb{R}^{(i-1)\times(i-1)} \\ \boldsymbol{a}_k \in \mathbb{R}^{i-1} \\ \gamma_k \in \mathbb{R} \end{cases} \tag{3.104}$$

$$(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}\tilde{\boldsymbol{G}}^{\mathrm{T}} =: \boldsymbol{\Pi} \begin{pmatrix} \boldsymbol{D}_k^{\mathrm{T}} \\ \boldsymbol{d}_k^{\mathrm{T}} \end{pmatrix} \quad \text{mit} \quad \begin{cases} \boldsymbol{D}_k \in \mathbb{R}^{(N+i)\times(i-1)} \\ \boldsymbol{d}_k \in \mathbb{R}^{N+i} \end{cases} \tag{3.105}$$

$$= \boldsymbol{\Pi} \begin{pmatrix} \boldsymbol{A}_k\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}} + \boldsymbol{a}_k\tilde{\boldsymbol{g}}_k^{\mathrm{T}} \\ \boldsymbol{a}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}} + \gamma_k\tilde{\boldsymbol{g}}_k^{\mathrm{T}} \end{pmatrix} . \tag{3.106}$$

Gleichung (3.106) ergibt sich dabei durch Multiplikation von (3.104) mit (3.102). Die Multiplikation von $\tilde{\boldsymbol{G}} = \begin{pmatrix} \tilde{\boldsymbol{G}}_{-k} & \tilde{\boldsymbol{g}}_k \end{pmatrix} \boldsymbol{\Pi}^{\mathrm{T}}$ mit (3.106) ergibt nun

$$\tilde{\boldsymbol{G}}(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}\tilde{\boldsymbol{G}}^{\mathrm{T}} = \tilde{\boldsymbol{G}}_{-k}\boldsymbol{A}_k\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}} + \tilde{\boldsymbol{G}}_{-k}\boldsymbol{a}_k\tilde{\boldsymbol{g}}_k^{\mathrm{T}} + \tilde{\boldsymbol{g}}_k\boldsymbol{a}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}} + \gamma_k\tilde{\boldsymbol{g}}_k\tilde{\boldsymbol{g}}_k^{\mathrm{T}} . \tag{3.107}$$

Der Vergleich von (3.103) mit (3.104) liefert somit

$$\begin{pmatrix} \boldsymbol{A}_k & \boldsymbol{a}_k \\ \boldsymbol{a}_k^{\mathrm{T}} & \gamma_k \end{pmatrix} = \begin{pmatrix} \tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k} & \tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{g}}_k \\ \tilde{\boldsymbol{g}}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k} & \tilde{\boldsymbol{g}}_k^{\mathrm{T}}\tilde{\boldsymbol{g}}_k \end{pmatrix}^{-1} . \tag{3.108}$$

Hieraus folgt mit der Formel für die Inverse einer Blockmatrix (siehe z.B. [56], S. 18) die Beziehung

$$(\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k})^{-1} = \boldsymbol{A}_k - \frac{1}{\gamma_k}\boldsymbol{a}_k\boldsymbol{a}_k^{\mathrm{T}} . \tag{3.109}$$

Multiplikation von (3.109) von links mit $\tilde{\boldsymbol{G}}_{-k}$ und von rechts mit $\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}$ ergibt unter

Verwendung von (3.107)

$$\begin{aligned}\tilde{\boldsymbol{G}}_{-k}(\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k})^{-1}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}} &= \tilde{\boldsymbol{G}}_{-k}\boldsymbol{A}_k\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}} - \frac{1}{\gamma_k}\tilde{\boldsymbol{G}}_{-k}\boldsymbol{a}_k\boldsymbol{a}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}} \\ &= \tilde{\boldsymbol{G}}(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}\tilde{\boldsymbol{G}}^{\mathrm{T}} - \tilde{\boldsymbol{G}}_{-k}\boldsymbol{a}_k\tilde{\boldsymbol{g}}_k^{\mathrm{T}} - \tilde{\boldsymbol{g}}_k\boldsymbol{a}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}} \\ &\quad - \gamma_k\tilde{\boldsymbol{g}}_k\tilde{\boldsymbol{g}}_k^{\mathrm{T}} - \frac{1}{\gamma_k}\tilde{\boldsymbol{G}}_{-k}\boldsymbol{a}_k\boldsymbol{a}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\,.\end{aligned} \tag{3.110}$$

Der Vergleich von (3.105) mit (3.106) liefert $\boldsymbol{d}_k = \tilde{\boldsymbol{G}}_{-k}\boldsymbol{a}_k + \gamma_k\tilde{\boldsymbol{g}}_k$, woraus folgt

$$\frac{1}{\gamma_k}\boldsymbol{d}_k\boldsymbol{d}_k^{\mathrm{T}} = \tilde{\boldsymbol{G}}_{-k}\boldsymbol{a}_k\tilde{\boldsymbol{g}}_k^{\mathrm{T}} + \tilde{\boldsymbol{g}}_k\boldsymbol{a}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}} + \gamma_k\tilde{\boldsymbol{g}}_k\tilde{\boldsymbol{g}}_k^{\mathrm{T}} + \frac{1}{\gamma_k}\tilde{\boldsymbol{G}}_{-k}\boldsymbol{a}_k\boldsymbol{a}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\,. \tag{3.111}$$

Einsetzen von (3.111) in (3.110) liefert schließlich

$$\tilde{\boldsymbol{G}}_{-k}(\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k})^{-1}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}} = \tilde{\boldsymbol{G}}(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}\tilde{\boldsymbol{G}}^{\mathrm{T}} - \frac{1}{\gamma_k}\boldsymbol{d}_k\boldsymbol{d}_k^{\mathrm{T}} \tag{3.112}$$

und damit

$$\tilde{\boldsymbol{y}}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k}(\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k})^{-1}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{y}}^{\mathrm{T}}\tilde{\boldsymbol{G}}(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{y}} - \frac{1}{\gamma_k}(\tilde{\boldsymbol{y}}^{\mathrm{T}}\boldsymbol{d}_k)^2\,. \tag{3.113}$$

Das bedeutet nun folgendes: Werden alle i Terme benutzt (d.h. alle Spalten von $\boldsymbol{G}$), so ist der Wert der regularisierten Kostenfunktion nach (3.101) $\mathrm{SSE}_{\mathrm{RR}}^{(i)} = \tilde{\boldsymbol{y}}^{\mathrm{T}}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{y}}^{\mathrm{T}}\tilde{\boldsymbol{G}}(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{y}}$. Weglassen der k-ten Spalte von $\boldsymbol{G}$, also der Übergang von $\boldsymbol{G}$ zu $\boldsymbol{G}_{-k}$, vergrößert diesen Wert auf

$$\mathrm{SSE}_{\mathrm{RR}}^{(i,-k)} = \tilde{\boldsymbol{y}}^{\mathrm{T}}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{y}}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k}(\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k})^{-1}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{y}} \overset{(3.113)}{=} \mathrm{SSE}_{\mathrm{RR}}^{(i)} + \frac{1}{\gamma_k}(\tilde{\boldsymbol{y}}^{\mathrm{T}}\boldsymbol{d}_k)^2\,. \tag{3.114}$$

Für eine Implementierung des Algorithmus' im Computer müssen die Inverse $(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}$ und der Lösungsvektor $\boldsymbol{w} = (\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{y}$ im Speicher des Rechners vorgehalten werden. Um den Term zu finden, dessen Entfernung aus dem Modell den geringsten Anstieg der (regularisierten) Kostenfunktion bewirkt, muss die Größe $(\tilde{\boldsymbol{y}}^{\mathrm{T}}\boldsymbol{d}_k)^2/\gamma_k$ für alle k berechnet werden. Es wird dann derjenige Term $\boldsymbol{g}_k$ entfernt, für den dieser Ausdruck den kleinsten Wert annimmt. Dabei ist $\tilde{\boldsymbol{y}}^{\mathrm{T}}\boldsymbol{d}_k$ aber gerade die k-te Komponente des Lösungsvektors für das Modell mit allen i Termen (vgl. (3.99) und (3.105)), und γ_k ist nach (3.104) das k-te Diagonalelement von $(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}$. Es werde nun der Term $\boldsymbol{g}_k$ entfernt. Anschließend erfolgt das Update der im Speicher vorgehaltenen Hilfsgrößen $(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}$ und $\boldsymbol{w}$, also der Übergang von $\boldsymbol{G}$ und $\boldsymbol{w}$ zu $\boldsymbol{G}_{-k}$ und $\boldsymbol{w}_{-k}$: Nach (3.109) berechnet sich die

aktualisierte Inverse zu

$$(\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k})^{-1} = \boldsymbol{A}_k - \frac{1}{\gamma_k}\boldsymbol{a}_k\boldsymbol{a}_k^{\mathrm{T}}\,. \tag{3.115}$$

Damit und durch Vergleich von (3.105) mit (3.106) ergibt sich der aktualisierte Lösungsvektor zu

$$\begin{aligned}\boldsymbol{w}_{-k} &= (\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k})^{-1}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{y}} \\ &= \boldsymbol{A}_k\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{y}} - \frac{1}{\gamma_k}\boldsymbol{a}_k\boldsymbol{a}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{y}} \\ &= \boldsymbol{D}_k^{\mathrm{T}}\tilde{\boldsymbol{y}} - \boldsymbol{a}_k\tilde{\boldsymbol{g}}_k^{\mathrm{T}}\tilde{\boldsymbol{y}} - \frac{1}{\gamma_k}\boldsymbol{a}_k\boldsymbol{a}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{y}}\,.\end{aligned} \tag{3.116}$$

Dabei ist nach (3.106) $\boldsymbol{D}_k^{\mathrm{T}}\tilde{\boldsymbol{y}}$ aber gerade gleich $(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{y}}$ nach Weglassen der k-ten Komponente, also einfach der alte Lösungsvektor $\boldsymbol{w}$ nach Entfernen der k-ten Komponente $\tilde{\boldsymbol{y}}^{\mathrm{T}}\boldsymbol{d}_k$. $\boldsymbol{A}_k$ ist die Matrix, die sich aus $(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}$ durch Streichen der k-ten Zeile und Spalte ergibt. $\boldsymbol{a}_k$ hingegen ist die k-te Spalte von $(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1}$ nach Weglassen der k-ten Komponente, also nach Weglassen des Diagonalelements, welches gerade γ_k ist. Schließlich entsteht $\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{y}}$ aus $\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{y}}$ durch Entfernung der k-ten Komponente $\tilde{\boldsymbol{g}}_k^{\mathrm{T}}\tilde{\boldsymbol{y}}$.

Algorithmus 3.2 Backward Elimination mit Ridge-Regression

Require: Modell $\hat{\boldsymbol{y}} = \boldsymbol{G}\boldsymbol{w}$ aus P Termen mit der Design-Matrix $\boldsymbol{G} = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_P) \in \mathbb{R}^{N\times P}$, Modellierungsziel $\boldsymbol{y} \in \mathbb{R}^N$, Ridgeparameter $\lambda \geq 0$, Inverse $(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1} = (\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G} + N\lambda\mathbb{1}_P)^{-1}$, Produkt $\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{y}} = \boldsymbol{G}^{\mathrm{T}}\boldsymbol{y}$.

1: Setze den Iterationszähler $j = 1$.
2: **while** (noch kein Abbruchkriterium erfüllt) **do**
3: {Bestimme den Index k des zu entfernenden Terms:}
4: $k = \underset{i=1,\ldots,P-j+1}{\arg\min}\ (\boldsymbol{d}_i^{\mathrm{T}}\hat{\boldsymbol{y}})/\gamma_k = \underset{i=1,\ldots,P-j+1}{\arg\min}\ w_i/((\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1})_{ii}$
{Update der Inversen nach (3.109):}
5: $(\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{G}})^{-1} \leftarrow (\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k})^{-1} = \boldsymbol{A}_k - \frac{1}{\gamma_k}\boldsymbol{a}_k\boldsymbol{a}_k^{\mathrm{T}}$
{Update von $\boldsymbol{w}$ nach (3.116):}
6: $\boldsymbol{w} \leftarrow \boldsymbol{w}_{-k} = \boldsymbol{D}_k^{\mathrm{T}}\tilde{\boldsymbol{y}} - \boldsymbol{a}_k\tilde{\boldsymbol{g}}_k^{\mathrm{T}}\tilde{\boldsymbol{y}} - \frac{1}{\gamma_k}\boldsymbol{a}_k\boldsymbol{a}_k^{\mathrm{T}}\tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{y}}$
{Update von $\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{y}}$ durch Streichen der k-ten Komponente $\tilde{\boldsymbol{g}}_k^{\mathrm{T}}\tilde{\boldsymbol{y}}$:}
7: $\tilde{\boldsymbol{G}}^{\mathrm{T}}\tilde{\boldsymbol{y}} \leftarrow \tilde{\boldsymbol{G}}_{-k}^{\mathrm{T}}\tilde{\boldsymbol{y}}$
8: $j \leftarrow j + 1$
9: {Überprüfe mögliche Abbruchbedingungen.}
10: **end while**

Eine Formulierung des Algorithmus' in Pseudocode ist in Alg. 3.2 dargestellt. Der in Zeile 4 bestimmte Index k bezieht sich dabei immer auf die noch verbleibenden Terme, die im Schritt j von 1 bis $P-j+1$ durchnummeriert werden, wodurch es ggf. von einem zum nächsten Schritt zu einer Umnummerierung kommen kann.

3.3.3 Forward Selection mit LOO-Kriterium

Die in Abschnitt 3.3.1 beschriebene Forward Selection verwendet als Selektionskriterium den quadratischen Fehler (3.15) bzw. den regularisierten quadratischen Fehler (3.45) für die Trainingsdaten. Dieser fällt monoton mit der Anzahl der Terme und eignet sich daher nicht als Schätzer für den zu erwartenden Testfehler auf neuen Daten. Zur Erkennung und Vermeidung von Overfitting wird daher oft die Cross-Validation verwendet und so die Termanzahl bestimmt, bei der der CV-Fehler minimal ist.

In diesem Abschnitt wird nun ein Algorithmus vorgestellt, der die Forward Selection mit der Leave-one-out Cross-Validation (vgl. Abschnitt 3.2.3) verbindet, indem der LOO- oder PRESS-Fehler (3.31) als Termselektionskriterium verwendet wird. Wie bei der in Abschnitt 3.3.1 beschriebenen, gewöhnlichen Forward Selection wird beginnend mit null Termen in jedem Iterationsschritt der Term aus dem Kandidatenpool zum Modell hinzugefügt, der die größte Reduktion des LOO-Fehlers bewirkt. Ähnlich wie der MSE auf den Trainingsdaten fällt auch der LOO-Fehler zu Beginn der Iteration monoton, erreicht im Gegensatz zu ersterem aber schließlich ein Minimum und beginnt bei weiterer Hinzunahme von Termen dann zu steigen [57]. Daraus leitet sich direkt ein Abbruchkriterium des Algorithmus' her, indem als finales Modell dasjenige gewählt wird, bei dem der LOO-Fehler sein Minimum erreicht.

Es sei nun wieder $\mathcal{D}_N = \{(\boldsymbol{x}_t, y_t) | t = 1, \ldots, N\}$ die Menge der Trainingsdaten, $g(\boldsymbol{x})$ ein Model aus M Termen, dessen lineare Koeffizienten $\boldsymbol{w}$ auf $\mathcal{D}_N$ durch Minimierung der regularisierten Kostenfunktion

$$J = \frac{1}{N}\|\boldsymbol{y} - \boldsymbol{G}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2 \tag{3.117}$$

bestimmt wurden, wobei $\boldsymbol{G}$ wieder die Design-Matrix des Modells bezeichnet, $\hat{y}_t = g(\boldsymbol{x}_t)$ die Modellausgabe für die Eingabe $\boldsymbol{x}_t$ und $e_t = y_t - \hat{y}_t$ das zugehörige Residuum. $\mathcal{D}_{N,-t} = \mathcal{D}_N \setminus \{(\boldsymbol{x}_t, y_t)\}$ sei die aus $\mathcal{D}_N$ durch Entfernung des t-ten Trainingspaares hervorgehende Menge und $g_{-t}(\boldsymbol{x})$ das aus den gleichen Termen

wie $g(\boldsymbol{x})$ bestehende Modell, dessen Koeffizientenvektor aber durch Minimierung von (3.117) auf $\mathcal{D}_{N,-t}$ bestimmt wurden. Schließlich bezeichnen $\hat{y}_{-t} = g_{-t}(\boldsymbol{x}_t)$ und $e_{-t} = y_t - \hat{y}_{-t}$ wieder die Modellausgabe und das Residuum dieses Modells für die Eingabe $\boldsymbol{x}_t$. Einen Ausdruck für den LOO-Fehler unter Verwendung der Ridge-Regression erhält man völlig analog zu der in Abschnitt 3.2.3 gebrachten Herleitung von (3.37) für den Fall ohne Regularisierung. Es muss lediglich die Definition von $\boldsymbol{H}$ in (3.33) durch $\boldsymbol{H} = \boldsymbol{G}^{\mathrm{T}}\boldsymbol{G} + N\lambda\mathbb{1}$ ersetzt werden (vgl. (3.50)). Dann ergibt sich für den LOO-Fehler des Modells $g(\boldsymbol{x})$

$$\begin{aligned} J_{\mathrm{LOO}} &= \frac{1}{N}\sum_{t=1}^{N}(y_t - g_{-t}(\boldsymbol{x}_t))^2 \\ &= \frac{1}{N}\sum_{t=1}^{N} e_{-t}^2 \\ &= \frac{1}{N}\sum_{t=1}^{N}\left(\frac{y_t - \hat{y}_t}{1 - \boldsymbol{G}(t,:)(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G} + N\lambda\mathbb{1})^{-1}\boldsymbol{G}(t,:)^{\mathrm{T}}}\right)^2, \end{aligned} \tag{3.118}$$

wobei wieder nach Matlab-Notation $\boldsymbol{G}(t,:) \in \mathbb{R}^{1\times M}$ die t-te Zeile von $\boldsymbol{G}$ ist. Bei der LOO-CV wird die Datenmenge also N mal unterschiedlich in $N-1$ Trainingsdaten und 1 Validierungsdatum aufgeteilt, so dass jedes Datum einmal zur Validierung verwendet wird, wobei die Modellkoeffizienten jeweils auf den $N-1$ übrigen Trainingsdaten gefittet werden. Die aus diesen N Aufteilungen berechneten quadrierten Validierungsfehler werden aufsummiert und ihr Mittelwert ergibt den LOO-CV-Fehler (3.118). Diese Formel zeigt, dass die N Parameterfits für die N Aufteilungen der Daten und die mit jedem Fit einhergehende Matrixinversion (3.18) gar nicht explizit durchgeführt werden müssen, sondern es genügt ein einziger Fit der Koeffizienten für das auf allen N Trainingsdaten konstruierte Modell. Die damit einhergehende Reduktion des Aufwands zur Berechnung des LOO-Fehlers für ein gegebenes Modell ist zwar beträchtlich. Soll (3.118) aber als Termauswahlkriterium für die Forward Selection verwendet werden, so muss der Parameterfit auf allen Trainingsdaten in jedem Iterationsschritt für jeden der verbleibenden Kandidatenterme durchgeführt werden, was bei einem großen Kandidatenpool immer noch einen enormen Rechenaufwand bedeutet. Hong et al. beschreiben in [57] eine Variante, die auf einer schrittweisen orthogonalen Faktorisierung der Design-Matrix beruht und eine rekursive Berechnung der LOO-Fehler während des Iterationsprozesses erlaubt. In [58] und [59] erweitern die Autoren diese Variante um die Tikhonov-Phillips-Regularisierung, die sie im Gegensatz

zu (3.117) allerdings auf die Koeffizienten der orthogonalisierten Basisfunktionen anwenden, was prinzipiell gleichbedeutend ist, da zwischen beiden ein linearer Zusammenhang besteht. Hong et al. verwenden in [58] die orthogonale Faktorisierung

$$\boldsymbol{G} = \boldsymbol{V} \cdot \boldsymbol{A} \tag{3.119}$$

der Design-Matrix mit der oberen Dreiecksmatrix

$$\boldsymbol{A} = \begin{pmatrix} 1 & a_{12} & \dots & a_{1M} \\ & 1 & & \vdots \\ & & \ddots & a_{M-1,M} \\ 0 & & & 1 \end{pmatrix} \tag{3.120}$$

mit Einsen auf der Diagonale und

$$\boldsymbol{V} = (\boldsymbol{v}_1, \dots, \boldsymbol{v}_M) \text{ mit } \boldsymbol{v}_i{}^{\mathrm{T}} \boldsymbol{v}_j = 0 \text{ für } i \neq j\,, \tag{3.121}$$

die wie bei der Besprechung der OLS-Methode in Abschnitt 3.3.1 simultan mit der Termauswahl und somit schrittweise konstruiert wird. Die Modellausgabe lässt sich nun schreiben als

$$\hat{\boldsymbol{y}} = \boldsymbol{G}\boldsymbol{w} = \boldsymbol{V}\boldsymbol{\alpha}\,, \tag{3.122}$$

wobei $\boldsymbol{\alpha} = \boldsymbol{A}\boldsymbol{w}$ der Koeffizientenvektor der Orthogonalbasis ist. Die zu minimierende Kostenfunktion lautet dann[13]

$$J = \|\boldsymbol{y} - \boldsymbol{V}\boldsymbol{\alpha}\|_2^2 + N\lambda\|\boldsymbol{\alpha}\|_2^2\,. \tag{3.123}$$

Für den LOO-Fehler gilt analog zu (3.118)

$$J_{\mathrm{LOO}} = \frac{1}{N}\sum_{t=1}^{N} \frac{y_t - \hat{y}_t}{1 - \boldsymbol{V}(t,:)(\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} + N\lambda\mathbb{1})^{-1}\boldsymbol{V}(t,:)^{\mathrm{T}}} = \frac{1}{N}\sum_{t=1}^{N} \frac{e_t^2}{\eta_t^2} \tag{3.124}$$

mit den Gewichtungsfaktoren $\eta_t = 1 - \boldsymbol{V}(t,:)(\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} + N\lambda\mathbb{1})^{-1}\boldsymbol{V}(t,:)^{\mathrm{T}}$. Der Vorteil durch die Verwendung der Orthogonalbasis liegt nun darin, dass es sich bei der Inversen im Nenner von (3.124) um die Inverse einer Diagonalmatrix handelt, die sehr einfach zu berechnen ist, und dass hierdurch die Berechnung des LOO-Fehlers (3.124) einfach rekursiv durchgeführt werden kann: $e_{l,t}$, $\eta_{l,t}$ und $J_{\mathrm{LOO},l}$

[13] Im Gegensatz zu [58] wird hier der Regularisierungsterm noch mit der Anzahl N der Trainingsdaten multipliziert, um die Wahl von λ unabhängig von der Anzahl der Datenpunkte zu machen.

seien Residuum, Gewichtungsfaktor und LOO-Fehler im l-ten Iterationsschritt der Forward Selection ($l = 1,2,\ldots$), und $v_{l,t}$ sei die t-te Komponente von $\boldsymbol{v}_l$. Dann gilt

$$e_{l,t} = y_t - \sum_{k=1}^{l} \alpha_k v_{k,t} = e_{l-1,t} - \alpha_l v_{l,t} \tag{3.125}$$

$$\eta_{l,t} = 1 - \sum_{k=1}^{l} \frac{v_{l,t}^2}{\boldsymbol{v}_k^{\mathrm{T}} \boldsymbol{v}_k + N\lambda} = \eta_{l-1,t} - \frac{v_{l,t}^2}{\boldsymbol{v}_l^{\mathrm{T}} \boldsymbol{v}_l + N\lambda} \tag{3.126}$$

$$J_{\mathrm{LOO},l} = \frac{1}{N} \sum_{t=1}^{N} \left(\frac{e_{l,t}}{\eta_{l,t}} \right)^2 \tag{3.127}$$

Nach der rekursiven Berechnung der Residuen und Gewichtungsfaktoren aus dem vorigen Iterationsschritt kann damit leicht der LOO-Fehler für den aktuellen Iterationsschritt berechnet werden. Die folgende Darstellung eines Algorithmus' zur Forward Selection mit dem LOO-Fehler als Termauswahlkriterium lehnt sich an die Darstellung in [59] an und verwendet das modifizierte Gram-Schmidt-Verfahren (MGS, siehe z.B. [39]) zur schrittweisen Orthogonalisierung der Design-Matrix. Dazu sei ein Pool von P Kandidatentermen $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_P$ gegeben, und $\boldsymbol{G}^{(l-1)}$ sei definiert durch

$$\boldsymbol{G}^{(l-1)} = \left(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{l-1}, \boldsymbol{g}_l^{(l-1)}, \ldots, \boldsymbol{g}_P^{(l-1)} \right) , \tag{3.128}$$

wobei die ersten $l-1$ Spalten durch Anwendung des MGS bereits paarweise orthogonal zueinander sind und die übrigen Spalten $\boldsymbol{g}_l^{(l-1)}, \ldots, \boldsymbol{g}_P^{(l-1)}$ orthogonal zu $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{l-1}$ sind. Wird im Schritt l der Term j_l ausgewählt, so werden die Spalten l und j_l von $\boldsymbol{G}^{(l-1)}$ und ihre Nummerierung aus Gründen der Übersichtlichkeit vertauscht und das Ergebnis weiterhin als $\boldsymbol{G}^{(l-1)}$ bezeichnet. Weiterhin bezeichnet $\alpha_l^{(j)}$ den Koeffizient des j-ten Kandidatenterms, falls dieser in Schritt l aufgenommen wird, und $e_{l,t}^{(j)}$, $\eta_{l,t}^{(j)}$ und $J_{\mathrm{LOO},l}^{(j)}$ die zugehörigen Residuen bzw. Gewichtungsfaktoren und den LOO-Fehler bei Aufnahme dieses Terms. Dann lässt sich der Algorithmus folgendermaßen in Pseudocode angeben:

Algorithmus 3.3 Forward Selection mit Leave-One-Out Cross-Validation

Require: Design-Matrix der Kandidatenterme $\boldsymbol{G} = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_P) \in \mathbb{R}^{N \times P}$, Modellierungsziel $\boldsymbol{y} \in \mathbb{R}^N$, Regularisierungsparameter $\lambda \geq 0$.

1: {Initialisierung:}
2: Setze $l = 0$, $J_{\mathrm{LOO},0} = \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}/N$, $\boldsymbol{G}^{(0)} = \boldsymbol{G}$, $\forall t = 1, \ldots, N:\ e_{0,t} = y_t,\ \eta_{0,t} = 1$
3: **while** (noch kein Abbruchkriterium erfüllt) **do**
4: $\quad l \leftarrow l + 1$
5: $\quad$ **for** $j = l$ to P **do**
6: $\quad\quad \alpha_l^{(j)} = \frac{\left(\boldsymbol{g}_j^{(l-1)}\right)^{\mathrm{T}} \boldsymbol{e}_{l-1}}{\left(\boldsymbol{g}_j^{(l-1)}\right)^{\mathrm{T}} \boldsymbol{g}_j^{(l-1)} + N\lambda}$
7: $\quad\quad \forall t = 1, \ldots, N:\ e_{l,t}^{(j)} = e_{l-1,t} - g_{j,t}^{(l-1)} \alpha_l^{(j)}, \quad \eta_{l,t}^{(j)} = \eta_{l-1,t} - \frac{\left(g_{j,t}^{(l-1)}\right)^2}{\left(\boldsymbol{g}_j^{(l-1)}\right)^{\mathrm{T}} \boldsymbol{g}_j^{(l-1)} + N\lambda}$
8: $\quad\quad J_{\mathrm{LOO},l}^{(j)} = \frac{1}{N} \sum_{t=1}^{N} \left(\frac{e_{l,t}^{(j)}}{\eta_{l,t}^{(j)}}\right)^2$
9: $\quad$ **end for**
10: $\quad j_l = \arg\min_{j=l,\ldots,P} \left\{ J_{\mathrm{LOO},l}^{(j)} \right\}$
11: $\quad J_{\mathrm{LOO},l} = J_{\mathrm{LOO},l}^{(j_l)}$
12: $\quad$ **if** $J_{\mathrm{LOO},l} > J_{\mathrm{LOO},l-1}$ **then**
13: $\quad\quad$ Abbruch; verwerfe den letzten Term wieder und behalte das Modell aus Schritt $l - 1$.
14: $\quad$ **else**
15: $\quad\quad$ Vertausche die Spalten l und j_l von $\boldsymbol{G}^{(l-1)}$.
16: $\quad\quad$ Vertausche $\boldsymbol{A}(1 : l-1, l)$ mit $\boldsymbol{A}(1 : l-1, j_l)$.
17: $\quad\quad \boldsymbol{v}_l = \boldsymbol{g}_l^{(l-1)}$
18: $\quad\quad$ **for** $j = l + 1$ to P **do**
19: $\quad\quad\quad a_{lj} = \frac{\boldsymbol{v}_l^{\mathrm{T}} \boldsymbol{g}_j^{(l-1)}}{\boldsymbol{v}_l^{\mathrm{T}} \boldsymbol{v}_l}$
20: $\quad\quad\quad \boldsymbol{g}_j^{(l)} = \boldsymbol{g}_j^{(l-1)} - a_{lj} \boldsymbol{v}_l$
21: $\quad\quad$ **end for**
22: $\quad\quad \alpha_l = \frac{\boldsymbol{v}_l^{\mathrm{T}} \boldsymbol{e}_{l-1}}{\boldsymbol{v}_l^{\mathrm{T}} \boldsymbol{v}_l + N\lambda}$
23: $\quad\quad \forall t = 1, \ldots, N:\ e_{l,t} = e_{l-1,t} - \alpha_l v_{l,t}, \quad \eta_{l,t} = \eta_{l-1,t} - \frac{v_{l,t}^2}{\boldsymbol{v}_l^{\mathrm{T}} \boldsymbol{v}_l + N\lambda}$
24: $\quad$ **end if**
25: $\quad$ {Überprüfe mögliche Abbruchbedingungen.}
26: **end while**

Die Berechnung der Residuen und Gewichtungsfaktoren und damit des LOO-Fehlers für die Kandidatenterme erfolgt in der **for**-Schleife ab Zeile 5. Der beste

Kandidat im Sinne der Minimierung dieses Fehlers wird in Zeile 10 bestimmt. Die Schleife ab Zeile 18 führt die Orthogonalisierung der übrigen Kandidatenterme bzgl. des gerade ausgewählten Terms nach dem MGS-Verfahren durch. Schließlich werden in den Zeilen 22 und 23 der Koeffizient des neuen Terms sowie rekursiv die neuen Residuen bzw. Gewichtungsfaktoren berechnet. Der Algorithmus bricht nach $M \leq P$ Schritten ab. $\boldsymbol{G}^{(M)}$ enthält dann in den ersten M Spalten die Orthogonalbasis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_M$ für die Design-Matrix der ausgewählten Terme und $\boldsymbol{A}(1:M,1:M)$ ist die zugehörige obere Dreiecksmatrix, mit deren Hilfe sich aus den orthogonalen Koeffizienten $\boldsymbol{\alpha}(1:M)$ der Koeffizientenvektor $\boldsymbol{w}$ des Modells durch Lösen von $\boldsymbol{\alpha} = \boldsymbol{A}\boldsymbol{w}$ mittels Rückwärtseinsetzen bestimmen lässt.

Miller [48] bemerkt, dass es sich bei der hier dargestellten, effizienten Art der Berechnung des LOO-Fehlers nach (3.118) nicht um eine *echte* Cross-Validation handelt. Eine solche muss nämlich auch die Auswahl der Terme selbst für jede Unterteilung der Trainingsdaten erneut durchführen. Dabei kann es passieren, dass für verschiedene Unterteilungen unterschiedliche Untermengen von Termen gewählt werden. Dann sind (3.118) und (3.124) aber nicht mehr anwendbar, denn dort wird die gleiche Termauswahl für jede Unterteilung vorausgesetzt. Indem in Algorithmus 3.3 in jedem Iterationsschritt der Term gewählt wird, der den LOO-Fehler am stärksten reduziert, fließen letztlich doch *alle* N Daten in die Bestimmung dieses Terms ein, und das jeweilige Validierungsdatum ist nicht mehr unabhängig von den Trainingsdaten. Daher fällt die Schätzung des Testfehlers durch den LOO-Fehler gewöhnlich zu optimistisch aus, d.h. der wahre Testfehler ist größer als der LOO-Fehler.

3.3.4 Sequential Replacement

Eine Schwäche der bisher betrachteten Greedy-Algorithmen ist, dass sie immer nur von einem zum nächsten Schritt auf „maximalen Gewinn“ aus sind. Im Falle der Forward Selection wird z.B. immer nur der nächstbeste Term ausgewählt, der zusammen mit den schon zuvor ausgewählten Termen die größte Fehlerreduktion bringt. Es kann jedoch der Fall auftreten, dass sich ein zu Beginn als „wichtig“ erscheinender und darum früh aufgenommener Term im Laufe der Iteration später als überflüssig erweist. Einmalig aufgenommene Terme verbleiben jedoch bei diesem Verfahren im Modell.

Die Methode des *Sequential Replacement* ist ein Verfahren, bei dem sequentiell versucht wird, Terme aus dem Modell gegen Terme aus dem Pool auszutauschen,

um so das Modell zu verbessern.

Seien nun ein aus M Termen bestehendes Modell und ein aus P Termen bestehender Pool gegeben (wobei das Modell z.B. mittels Forward Selection erzeugt wurde). Dies ist das *initiale* Modell. Die Idee des Sequential Replacement ist die folgende: Es wird testweise ein Modellterm $\boldsymbol{g}_i$ gegen einen Term $\boldsymbol{u}$ aus dem Pool ausgetauscht und die Kostenfunktion dieses so veränderten Modells berechnet. Auf diese Art und Weise wird durch Ausprobieren jedes Terms aus dem Pool der beste Ersatz $\boldsymbol{u}_{\text{opt}}$ für $\boldsymbol{g}_i$ bestimmt. Liefert dieser beste Ersatz einen kleineren Wert der Kostenfunktion als der Originalterm $\boldsymbol{g}_i$, so wird der alte Modellterm $\boldsymbol{g}_i$ dauerhaft durch $\boldsymbol{u}_{\text{opt}}$ ersetzt, d.h. $\boldsymbol{u}_{\text{opt}}$ ist jetzt im Modell und $\boldsymbol{g}_i$ im Pool. Anschließend wird zum nächsten Modellterm übergegangen und versucht, diesen ebenfalls durch einen besseren Kandidaten aus dem Pool zu ersetzen, dann zum übernächsten usw., bis alle Modellterme durchgegangen wurden. Ein kompletter Durchlauf des Sequential Replacement Algorithmus ist dann abgeschlossen, wenn versucht wurde, jeden Modellterm einmal durch jeden Poolterm zu ersetzen. Hat nach einem solchen kompletten Durchlauf irgendwo eine Ersetzung eines Modellterms stattgefunden, so wird wieder ein neuer Durchlauf beginnend mit dem ersten Modellterm gestartet. Auf diese Weise wird so lange fortgefahren, bis keine zu einem besseren Modell führende Termsubstitution mehr möglich ist.

3.3.5 Grafting

Ein Nachteil der bisher betrachteten Termselektionsalgorithmen Forward Selection und Backward Elimination ist, dass die Entscheidungen über die Aufnahme bzw. Entfernung eines Terms in jedem Iterationsschritt „definitiv“ sind: Bei der Forward Selection verbleibt ein einmal aufgenommener Term im Modell, selbst wenn er sich im Verlaufe der Aufnahme weiterer Terme als überflüssig herausstellt. Entsprechend kann bei der Backward Elimination ein einmal entfernter Term nicht wieder hinzugenommen werden. Mit der im vorigen Abschnitt beschriebenen Methode des Sequential Replacement kann im Nachhinein versucht werden, Terme des Modells gegen möglicherweise bessere Terme aus dem Kandidatenpool auszutauschen. Einen anderen Weg geht das hier beschriebene *Grafting*-Verfahren[14] [60,61], bei dem die Forward Selection mit einer l_1-regularisierten Kostenfunktion durchgeführt wird. Diese setzt sich zusammen aus einer Loss-Funktion (hier die quadratische Abweichung) sowie in der allgemeinsten Form je

[14] Der Begriff leitet sich ab von ***gra****dient* ***f****eature tes****ting***.

einem Term für eine l_0-, l_1- und l_2-Regularisierung:

$$J = \frac{1}{N}\sum_{t=1}^{N}(y_t - g(\boldsymbol{x}_t|\boldsymbol{w}))^2 + \lambda_2 \sum_{k=1}^{M} w_k^2 + \lambda_1 \sum_{k=1}^{M} |w_k| + \lambda_0 \sum_{k=1}^{M} k\,. \tag{3.129}$$

Alle Terme bis auf den der l_0-Regularisierung sind konvex, so dass auch die gesamte Kostenfunktion bei Verzicht auf diese Regularisierung konvex ist und ein globales Optimum besitzt, das mit gradientenbasierten Optimierungsverfahren erreicht werden kann. Der l_0-Term hingegen ist nichtkonvex, was die Optimierung bedeutend komplexer macht. Daher wurde die l_0-Regularisierung in dieser Arbeit nicht verwendet, so dass die Kostenfunktion nun

$$\begin{aligned} J &= \frac{1}{N}\sum_{t=1}^{N}(y_t - g(\boldsymbol{x}_t|\boldsymbol{w}))^2 + \lambda_2 \sum_{k=1}^{M} w_k^2 + \lambda_1 \sum_{k=1}^{M} |w_k| \\ &= \bar{L}(\boldsymbol{w}) + \lambda_2 \sum_{k=1}^{M} w_k^2 + \lambda_1 \sum_{k=1}^{M} |w_k| \end{aligned} \tag{3.130}$$

lautet. Grafting ist ein iteratives Verfahren, bei dem in jedem Schritt versucht wird, aus dem Vorrat an Modelltermen den geeignetsten Kandidaten ins Modell aufzunehmen. Die Trennung des gesamten Termvorrats in eine Modellmenge und einen Kandidatenpool, wie sie z.B. beim Sequential Replacement vollzogen wurde, entfällt hier allerdings. Vielmehr folgt das Verfahren der Beobachtung, dass ein Term g_k aus der Menge der Kandidaten genau dann zum Modell gehört, wenn sein linearer Koeffizient w_k einen von Null verschiedenen Wert hat. Der Grafting-Algorithmus funktioniert nun folgendermaßen: Sämtliche verfügbaren (Kandidaten-)Terme bzw. ihre linearen Koeffizienten w_j werden in zwei Klassen eingeteilt: $\mathcal{F}$ ist die Menge aller „freien“ Terme, deren Koeffizienten im Laufe der Optimierung angepasst werden können (das sind die Terme, aus denen das Modell im Moment besteht), während $\mathcal{Z}$ die Menge aller Terme bzw. Koeffizienten bezeichnet, deren Werte fest auf Null gehalten werden und die somit momentan nicht zur Modellausgabe beitragen. Im k-ten Schritt wird nun versucht, einen Term ins Modell aufzunehmen, also eines der Gewichte aus $\mathcal{Z}$ nach $\mathcal{F}$ zu überführen. Nach Aufnahme des Terms wird die gesamte Architektur durch Minimierung von (3.130) bzgl. aller Koeffizienten w_j mit $j \in \mathcal{F}$ optimiert, so dass anschließend gilt $\partial J/\partial w_j = 0\ \forall j \in \mathcal{F}$. Anstatt nun aber jeden Term aus $\mathcal{Z}$ einmal nach $\mathcal{F}$ zu übernehmen und die vollständige Optimierung bzgl. sämtlicher Modellkoeffizienten zu starten, um so den Term aus $\mathcal{Z}$ zu finden, der

(3.130) am stärksten reduziert, was einen sehr großen Rechenaufwand erfordern würde, verwendet Grafting eine schnelle Heuristik, um den *voraussichtlich* besten Kandidat zu finden: Als solcher wird der Term g_k angesehen, für den der Betrag der partiellen Ableitung der Kostenfunktion nach w_k an der aktuellen Position $w_k = 0$ maximal ist. Dieser Term wird allerdings nur dann aufgenommen, wenn er tatsächlich zu einer Verringerung der Kostenfunktion führt, was durch die Regularisierung nicht zwangsläufig der Fall ist. In Abschnitt 3.2.4.2 über die l_1-Regularisierung wurde dargelegt, dass $w_k \neq 0$ im Minimum von (3.130) genau dann gilt, falls die Bedingung

$$\left| \frac{\partial \bar{L}(\boldsymbol{w})}{\partial w_k} \right| > \lambda_1 \qquad \text{für} \qquad w_k = 0 \tag{3.131}$$

erfüllt ist. Andernfalls ist $w_k = 0$ im Minimum von (3.130). Diese Bedingung kann zur Definition eines Subgradienten der Kostenfunktion für alle w_j verwendet werden:

$$\frac{\partial J(\boldsymbol{w})}{\partial w_j} = \begin{cases} \bar{L}'(\boldsymbol{w}) + 2\lambda_2 w_j & \\ \quad +\lambda_1 \operatorname{sign}(w_j), & \text{falls } w_j \neq 0, \\ \bar{L}'(\boldsymbol{w}) + \lambda_1, & \text{falls } w_j = 0 \wedge \bar{L}'(\boldsymbol{w}) < -\lambda_1, \\ \bar{L}'(\boldsymbol{w}) - \lambda_1, & \text{falls } w_j = 0 \wedge \bar{L}'(\boldsymbol{w}) > \lambda_1, \\ 0, & \text{falls } w_j = 0 \wedge -\lambda_1 \leq \bar{L}'(\boldsymbol{w}) \leq \lambda_1. \end{cases} \tag{3.132}$$

Damit ist die Richtung des negativen Gradienten immer die Richtung des steilsten Abfalls der Kostenfunktion.

So wird nun im k-ten Grafting-Schritt für alle $w_j \in \mathcal{Z}$ der Absolutwert $|\partial J / \partial w_j|$ berechnet und der Term g_k bestimmt, für den dieser Ausdruck maximal ist. Falls für diesen Term das Aufnahmekriterium (3.131) erfüllt ist, wird dieser Term aus $\mathcal{Z}$ entfernt, zur Menge $\mathcal{F}$ der freien Modellterme hinzugefügt und eine Optimierung aller Koeffizienten $w_j \in \mathcal{F}$ durchgeführt. Andernfalls bricht der Algorithmus ab, da keine weitere Verbesserung des Modells möglich ist. Im Zuge der Optimierung kann es vorkommen, dass einige der Koeffizienten $w_j \in \mathcal{F}$ zu Null werden. Diese werden dann aus $\mathcal{F}$ entfernt und wieder der Menge $\mathcal{Z}$ hinzugefügt, fallen also aus dem Modell heraus. Dies lässt sich in der Praxis oft beobachten: Im Gegensatz z.B. zur Forward Selection verbleiben in früheren Iterationsschritten aufgenommene Terme nicht in jedem Fall im Modell, sondern können sich im Laufe des Iterationsprozesses als überflüssig erweisen und wieder aus dem Mo-

dell entfernt werden. Die Optimierung selbst kann mit einem gradientenbasierten Verfahren erfolgen, da nach der Erläuterung oben im Text klar ist, wie der Gradient der Kostenfunktion zu berechnen ist. Die Autoren von [60] schlagen hierzu das *Broyden-Fletcher-Goldfarb-Shanno*-Verfahren (kurz BFGS) vor [62, 63]. In der praktischen Anwendung kann es bei der Optimierung aufgrund der Unstetigkeit der Ableitung des l_1-Terms im Nullpunkt zu Problemen kommen, die es ggf. erforderlich machen, sehr klein werdende Koeffizienten explizit auf Null zu setzen. Als Startnäherung bei der Optimierung wird für den neuen Term dessen bisheriger Wert $w_k = 0$ verwendet und für die übrigen Terme deren Endwerte nach der Optimierung aus dem vorigen Grafting-Schritt.

Algorithmus 3.4 Grafting

Require: Design-Matrix der Kandidatenterme $\boldsymbol{G} = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_P) \in \mathbb{R}^{N \times P}$, Modellierungsziel $\boldsymbol{y} \in \mathbb{R}^N$, Regularisierungsparameter $\lambda_1, \lambda_2 \geq 0$, Koeffizientenvektor $\boldsymbol{w}^{(0)} \equiv \boldsymbol{0} \in \mathbb{R}^P$, Iterationszähler $l = 0$.

repeat
  {Bestimme den aussichtsreichsten Kandidaten $\boldsymbol{g}_k$:}
  $\forall j \in \mathcal{Z}: \; d_j = |\partial \bar{L}(\boldsymbol{w}^{(l)})/\partial w_j|$
  $k = \arg\max_{j \in \mathcal{Z}} d_j$
  if $d_k > \lambda_1$ **then**
    {Entferne k aus $\mathcal{Z}$ und füge es zu $\mathcal{F}$ hinzu:}
    $\mathcal{Z} \leftarrow \mathcal{Z} \setminus \{k\}$
    $\mathcal{F} \leftarrow \mathcal{F} \cup \{k\}$
    Bestimme $\boldsymbol{w}^{(l+1)}$ als Ergebnis der Minimierung von (3.130) bzgl. aller $w_j^{(l)}$ mit $j \in \mathcal{F}$ unter Verwendung von (3.132).
    {Falls bei der Optimierung Koeffizienten zu Null werden, entferne die entsprechenden Indizes aus $\mathcal{F}$ und füge sie $\mathcal{Z}$ hinzu:}
    $\mathcal{I} = \{j \in \mathcal{F} | w_j^{(l+1)} = 0\}$
    $\mathcal{F} = \mathcal{F} \setminus \mathcal{I}$
    $\mathcal{Z} = \mathcal{Z} \cup \mathcal{I}$
  end if
  $l \leftarrow l + 1$
until ($\mathcal{Z} = \emptyset$ oder es gibt kein w_j mit $j \in \mathcal{Z}$ mehr, das die Bedingung (3.131) erfüllt.)

Damit die oben besprochene Heuristik, in jedem Grafting-Schritt den aussichtsreichsten Term basierend auf dem Betrag der Ableitung am Ort $w = 0$ zu bestim-

men, zu sinnvollen Resultaten führen kann, ist es nötig, die Spalten der Design-Matrix wie in (3.41) zu normieren, damit sie mittelwertfrei sind und gleiche Varianz haben.

Da die zu minimierende Kostenfunktion (3.130) mit quadratischem Loss und l_1- sowie l_2-Regularisierung konvex ist, besitzt sie ein eindeutiges, globales Minimum. Der Algorithmus bricht garantiert ab: entweder, wenn alle Terme aufgenommen wurden oder, wenn kein $w_j \in \mathcal{Z}$ mehr die Bedingung (3.131) erfüllt. Das dann erreichte lokale Minimum von J ist also zugleich das globale Minimum von (3.130). Zur Bestimmung der optimalen Modellkomplexität kann wieder die Cross-Validation herangezogen werden.

Eine Formulierung des Algorithmus' in Pseudocode zeigt Alg. 3.4. Nach dem Abbruch des Algorithmus' enthält $\mathcal{F}$ die Indizes der ausgewählten Terme, d.h. für $j \in \mathcal{F}$ ist $\boldsymbol{g}_j$ ein ausgewählter Term und $w_j^{(l)}$ der zugehörige Koeffizient.

3.4 Vergleich verschiedener Termselektionsalgorithmen

Zunächst sollen einige für alle betrachteten Termselektionsalgorithmen gültige Effekte betrachtet werden. Als Beispiel des zu modellierenden Systems wird die Ramp-Hill-Funktion [64, 65] benutzt, die definiert ist durch

$$\begin{aligned} d_b &= 2\sqrt{(x_1 + 0{,}4)^2 + (x_2 + 0{,}4)^2} \\ y_l &= 2x_1 + 2{,}5x_2 - 0{,}5 \\ y_b &= \begin{cases} 2\cos(\pi d_b/2) & \text{falls } d_b \leq 1 \\ 0 & \text{sonst} \end{cases} \\ y &= \begin{cases} y_b - 1 & \text{falls } y_l < 0 \\ y_b + y_l - 1 & \text{falls } 0 \leq y_l \leq 2 \\ y_b + 1 & \text{sonst} \end{cases} \end{aligned} \tag{3.133}$$

Diese Funktion $y(x_1, x_2)$, die in Abb. 3.4 dargestellt ist, besteht aus einem linearen Anstieg (Ramp), einer Cosinus-Funktion (Hill) und zwei konstanten Gebieten, durch die sie für ein RBF-Modell schwierig zu approximieren ist. Es wurden Trainingsdatensätze mit 100, 500 und 5000 Ein- und Ausgabepaaren erzeugt, wobei

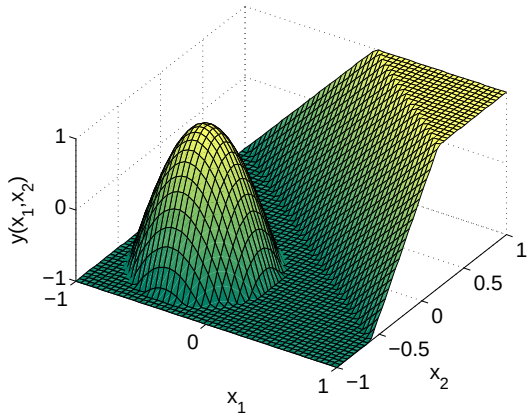
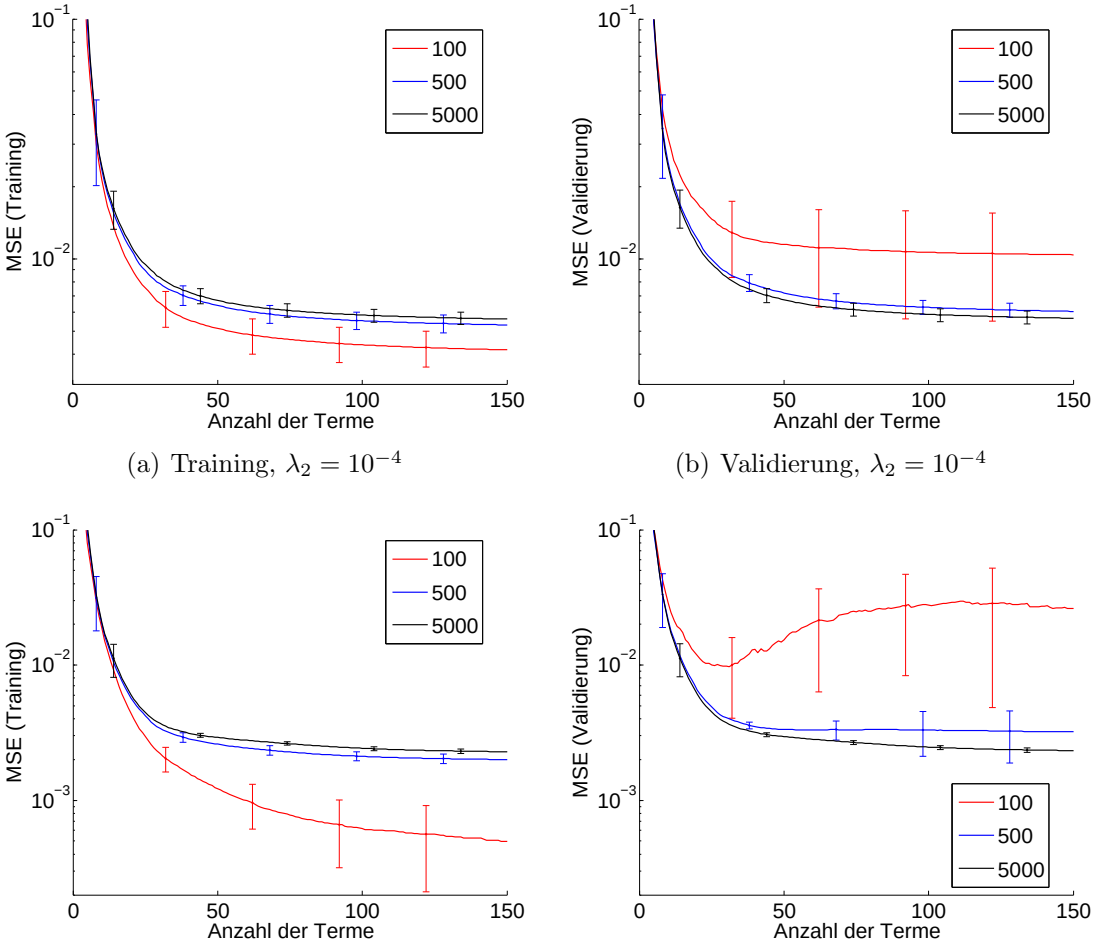


Abbildung 3.4: Die Ramp-Hill-Funktion (3.133)

die Eingaben zufällig gleichverteilt aus dem Intervall $[-1,1] \times [-1,1]$ gewählt wurden, sowie ein Validierungsdatensatz der Größe 5000, dessen Eingaben ebenfalls zufällig gleichverteilt aus demselben Intervall gewählt wurden. Für diese Trainingsdatensätze wurden RBF-Modelle durch Termselektion erzeugt, wobei immer der gleiche Vorrat von 2000 Kandidatentermen verwendet wurde, deren Zentren ebenfalls dem oben angegebenen Intervall entstammen und deren Breiten auf zufällige Werte aus dem 0,3- bis 3-fachen des mittleren euklidischen Abstandes zwischen allen Zentren gesetzt wurden. Weiterhin enthielt der Kandidatenpool noch einen linearen Term. Ein konstanter Term wurde dem Modell fest hinzugefügt und immer auf den Mittelwert der Ausgabedaten gesetzt. Als Termselektionsalgorithmus kam die Forward Selection mit Ridge-Regression zum Einsatz, wobei als Regularisierungsparameter die Werte $\lambda_2 = 10^{-4}$ und $\lambda_2 = 10^{-9}$ verwendet wurden. Die Termselektion wurde nach 150 aufgenommenen Termen abgebrochen. Dabei wurde in jedem Iterationsschritt zusätzlich zum Trainingsfehler auch der MSE auf den Validierungsdaten bestimmt und gespeichert. Insgesamt wurden so für jeden der 3 Trainingsdatensätze sowie für beide Werte des Regularisierungsparameters jeweils 100 Modelle trainiert und die Modellierungsfehler für die jeweiligen Modellgrößen über diese 100 Werte gemittelt. In diesen 100 Durchläufen wurden jeweils unterschiedliche, zufällig gewählte Trainings- und Validierungsdaten verwendet.

Abb. 3.5 zeigt als Ergebnisse die Mittelwerte der Modellierungsfehler (MSE) auf den Trainings- bzw. Validierungsdaten für die verschiedenen Größen der Trainingsdatensätze und die beiden Werte von λ_2. Zusätzlich ist für einige Termanzahlen auch die Standardabweichung über die 100 Durchläufe als Fehlerbalken



(a) Training, $\lambda_2 = 10^{-4}$

(b) Validierung, $\lambda_2 = 10^{-4}$

(c) Training, $\lambda_2 = 10^{-9}$

(d) Validierung, $\lambda_2 = 10^{-9}$

Abbildung 3.5: Der MSE für die Trainingsdaten und eine unabhängige Validierungsmenge in Abhängigkeit von der Anzahl der aufgenommenen Modellterme bei der Forward Selection für 2 verschiedene Werte des Regularisierungsparameters λ_2 und für verschiedene Größen des Trainingsdatensatzes. Die Werte in der Legende geben die Anzahl der verwendeten Trainingsdatenpaare an. Modelliert wurden Daten der Ramp-Hill-Funktion.

mit eingezeichnet. Man erkennt, dass bei gleicher Modellgröße der Trainingsfehler umso niedriger ist, je weniger Daten zum Training verwendet werden. Beim Fehler für die unabhängige Validierungsmenge ist es genau umgekehrt: Der Validierungsfehler sinkt bei einer *Vergrößerung* des Trainingsdatensatzes. Für den kleinsten Trainingsdatensatz der Größe 100 unterscheiden sich Trainings- und Validierungsfehler sehr stark, während sich ihre Verläufe mit zunehmender Vergrößerung der Trainingsmenge immer ähnlicher werden. Der Grund hierfür ist leicht ersichtlich: Je kleiner die Trainingsdatenmenge ist, desto statistisch ungenauer ist

die Approximation der zugrundeliegenden Funktion, was einen größeren Fehler für Daten, die keinen Eingang in das Training genommen haben, zur Folge hat. Diese bei kleinen Datensätzen auftretende große statistische Unsicherheit äußert sich auch deutlich in den entsprechend großen Fehlerbalken. Bei einer gegebenen Modellgröße ist das Verhältnis der Anzahl der Terme zur Anzahl der Trainingsdaten umso größer, je weniger Trainingsdaten verwendet werden, so dass mit der Anzahl der Trainingsdaten auch der Modellierungsfehler sinkt. Dabei ist zu beobachten, dass zwischen 100 und 500 Trainingsdaten ein großer Unterschied der Modellierungsfehler besteht. Eine weitere Vergrößerung auf 5000 Trainingsdaten bewirkt jedoch nur noch eine relativ geringe Verbesserung des Modellierungsergebnisses. Es tritt hier also ein Sättigungseffekt ein, jedoch ist die Chance, ein Modell mit guten Generalisierungseigenschaften zu erhalten, beim größeren Trainingsdatensatz deutlich besser, was sich in kleineren Streuungen des Validierungsfehlers bei $\lambda_2 = 10^{-9}$ in Abb. 3.5(d) äußert. Dort lässt sich auch sehr deutlich der Effekt des Overfitting erkennen: Der Regularisierungsparameter ist zu klein, um bei einer Größe des Trainingsdatensatzes von nur 100 Daten ein Overfitting wirkungsvoll zu verhindern, so dass der Validierungsfehler im Mittel über alle 100 Modelle bei einer Modellgröße von 31 Termen sein Minimum erreicht und bei einer weiteren Aufnahme von Termen wieder zu steigen beginnt. Werden mehr Daten zum Training des Modells benutzt, kommt es wegen der größeren statistischen Genauigkeit nicht so schnell zum Overfitting. Ein größerer Wert des Regularisierungsparameters schützt zwar besser vor dem Auftreten von Overfitting, verhindert jedoch auch eine genaue Approximation der zugrundeliegenden Funktion: Die Validierungsfehler für $\lambda_2 = 10^{-9}$ in Abb. 3.5(d) fallen tiefer als die für $\lambda_2 = 10^{-4}$ in Abb. 3.5(b).

Die geschilderten Beobachtungen hängen natürlich von der zu approximierenden Funktion ab und können bei einem anderen System ganz andere Werte für die Größe des Trainingsdatensatzes, die optimale Anzahl der Modellterme (die einen minimalen Validierungsfehler zur Folge hat) und den Wert des Regularisierungsparameters erforderlich machen. Ist die Zahl der zur Verfügung stehenden Trainingsdaten sehr gering, muss zur Vermeidung von Overfitting die Regularisierung entsprechend stark sein, was zu einer prinzipiellen Einschränkung der erreichbaren Genauigkeit der Modellierung führt. Zur Bestimmung der optimalen Modellgröße und der optimalen Regularisierungsstärke kann die in Abschnitt 3.2.3 behandelte Cross-Validation herangezogen werden.

Keine zufriedenstellenden Ergebnisse konnten mit der Grafting-Methode erzielt

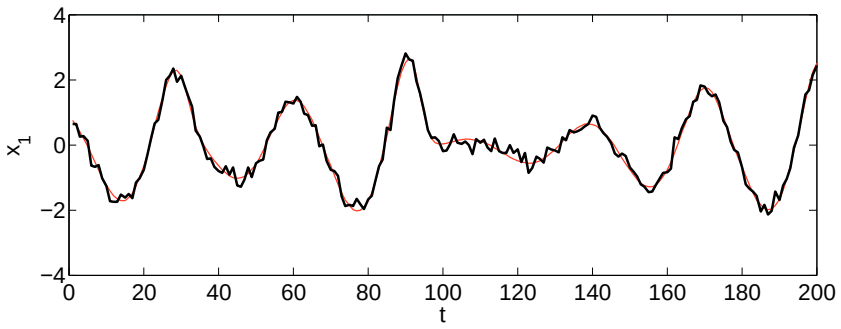


Abbildung 3.6: Ein Ausschnitt der verwendeten Zeitreihe des Rössler-Systems (Schwarz). Die rote Kurve zeigt zusätzlich den Verlauf der unverrauschten Daten.

werden, die darum bei den Vergleichen der Algorithmen ausgelassen wird.

Als nächstes folgt ein Vergleich der Forward Selection (FS) mit der Backward Elimination (BE). Da die Backward Elimination einen deutlich größeren Rechenaufwand erfordert, wurde weiterhin untersucht, inwieweit es sinnvoll ist, die Backward Elimination nur auf eine reduzierte Teilmenge des Kandidatenpools anzuwenden, deren Terme zuvor mit Hilfe der Forward Selection ausgewählt wurden. D.h. man selektiert zuerst mit Hilfe der Forward Selection einen Teil der Terme (und zwar mehr, als sich mit Cross-Validation als optimale Anzahl ergibt) aus dem Kandidatenpool in der Hoffnung, dabei auf jeden Fall alle signifikanten Terme zu erfassen, und entfernt aus dieser Teilmenge wieder die sich als überflüssig erweisenden Basisfunktionen mittels Backward Elimination. Zusätzlich wird noch die FS-Variante mit der Minimierung des LOO-Fehlers als Auswahlkriterium zum Vergleich herangezogen. Als Beispiel dient eine Zeitreihe $\{s_t\}$ aus numerisch generierten Daten des Rössler-Oszillators, die zusätzlich mit einem mittelwertfreien, normalverteilten Rauschsignal $\{\epsilon_t\}$ belegt wurde, wobei das Verhältnis der Standardabweichungen $\sigma_s/\sigma_\epsilon = 0{,}03$ betrug. Abb. 3.6 zeigt einen Auschnitt der verwendeten Zeitreihe. Die Länge der zum Training des Modells verwendeten Zeitreihe betrug 10000 Samples. Die nur zur Validierung verwendete unabhängige zweite Zeitreihe hatte ein Länge von 20000 Samples. Aus den Zeitreihen wurden Delay-Vektoren konstruiert, wobei eine Einbettungsdimension von $D = 7$ mit einem Delay $\tau = 4\Delta t$ bei $\Delta t = 0{,}2$ verwendet wurde. 2004 dieser Vektoren wurden aus den Trainingsdaten zufällig als Zentren der Kandidatenterme ausgewählt. Die Breiten der RBF-Terme wurden wieder zufällig in einem kleinen Intervall um den mittleren euklidischen Abstand zwischen den Daten festgelegt. Die direkte Vorhersageschrittweite betrug 10 Samples. Bei der Hintereinanderschaltung von Forward Selection und Backward Elimination wurde die Aufnahme von Termen

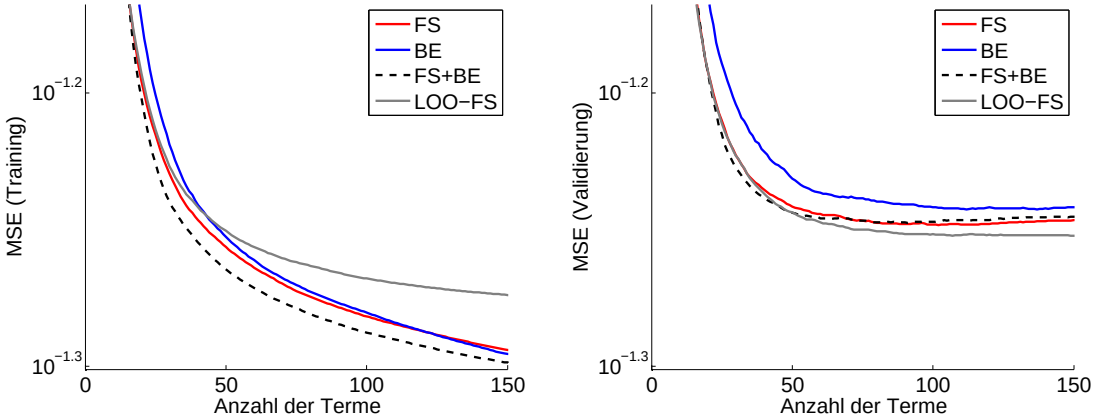


Abbildung 3.7: MSE für die Trainingsdaten (links) und für die Validierungsdaten (rechts) bei der Modellierung der verrauschten Rössler-Daten mit der Forward Selection (FS), der Backward Elimination (BE), einer der FS nachgeschalteten Backward Elimination (FS+BE) sowie der Forward Selection mit LOO-Kriterium (LOO-FS). Ausnahme: Die Kurve für LOO-FS in der linken Grafik zeigt die regularisierten LOO-Fehler (3.118).

durch die FS bei einer erreichten Modellgröße von 300 Termen abgebrochen und anschließend dieses Modell durch die Backward Elimination schrittweise wieder verkleinert. Die Ergebnisse der Termselektion zeigt Abb. 3.7. Dort ist der Verlauf des MSE für die Trainings- bzw. Validierungsdaten über der Anzahl der Modellterme dargestellt, wobei die Kurven jeweils über 20 Selektionsdurchläufe mit jeweils zufälliger Auswahl der Zentren und Breiten gemittelt wurden. Überraschenderweise schneidet die Backward Elimination hier am schlechtesten ab, was den Fehler auf den Validierungsdaten betrifft. Auf den Trainingsdaten ist der MSE bei einer Modellgröße von mehr als ca. 120 Termen niedriger als bei der Forward Selection, für kleinere Termanzahlen liegt der Fehler aber auch dort oberhalb dem für die Forward Selection. Eine mögliche Erklärung für dieses Phänomen liegt in der Tatsache, dass bei der Backward Elimination viel mehr Iterationsschritte nötig sind, um den Anfangspool von 2004 Termen auf z.B. eine in den Fehlerkurven gezeigte Größe von 100 Termen zu verringern (nämlich 1904) als bei der Forward Selection, die, bei 0 Termen beginnend, bereits nach 100 Schritten bei dieser Modellgröße angelangt ist. Da einmal verworfene Terme bei der Backward Elimination nicht wieder aufgenommen werden bzw. einmal aufgenommene Terme bei der Forward Selection nicht wieder entfernt werden können, ist bei vielen Iterationsschritten die Wahrscheinlichkeit, mit der immer nur von Iterationsschritt zu Iterationsschritt optimierenden Greedy-Strategie eine Entscheidung zu treffen, die sich später als falsch erweist, viel größer. Kurz ausgedrückt: Bei wenigen

Iterationsschritten kann man nur wenig falsch machen, bei vielen hingegen viel.

Am besten schnitt die Forward Selection mit LOO-Kriterium ab, wobei die Absolutwerte der Fehler nicht ganz direkt vergleichbar sind, da bei diesem Algorithmus wie in Abschnitt 3.3.3 beschrieben die Regularisierung auf die Koeffizienten der orthogonalisierten Basisfunktionen angewendet wird. Die Kurve im linken Teil der Abb. 3.7 zeigt hier nicht den MSE auf den Trainingsdaten, sondern den regularisierten LOO-Fehler, der bei Erreichen seines Minimums als Abbruchkriterium des Algorithmus' dient. Hier deutet sich aus dem Verlauf von regularisiertem LOO-Fehler und dem Validierungsfehler die auch in der Literatur [24] beschriebene Tendenz der Leave-one-out Cross-Validation zum Overfitting an: Auch bei einer Modellgröße von 150 Termen fällt der LOO-Fehler weiter, während der Validierungsfehler bereits in Sättigung gegangen ist.

3.5 Nichtlineare Optimierung des Modells

Die in dieser Arbeit eingesetzten Modelle bestehen aus einer Linearkombination Gauß'scher radialer Basisfunktionen sowie einem konstanten und einem linearen Term und haben die Gestalt

$$\begin{aligned} g(\boldsymbol{x}) &= w_0 + \sum_{d=1}^{D} w_d x_d + \sum_{k=D+1}^{D+M} w_k g_k(\boldsymbol{x}) \\ &= w_0 + \sum_{d=1}^{D} w_d x_d + \sum_{k=D+1}^{D+M} w_k \exp\left(- \sum_{d=1}^{D} \frac{(x_d - c_{kd})^2}{r_{kd}^2} \right) . \end{aligned} \tag{3.134}$$

Der in den vorangehenden Abschnitten beschriebene Ansatz, für einen gegebenen Satz von Messdaten ein Modell mit guten Generalisierungseigenschaften zu konstruieren, besteht in der Optimierung der linearen Modellkoeffizienten $\boldsymbol{w}$ für einen Satz von Basisfunktionen $g_k(\boldsymbol{x}, \boldsymbol{c}_k, \boldsymbol{r}_k)$ mit festen Parametern $\boldsymbol{c}_k$ und $\boldsymbol{r}_k$. Diese Optimierung besteht in der Minimierung des quadratischen Fehlers SSE, dem ggf. ein Regularisierungsterm hinzugefügt wird. Eine Optimierung der übrigen Parameter des RBF-Modells, also der Zentrenkoordinaten c_{kd} und der Breitenskalierungen r_{kd}, wurde bisher nur dahingehend durchgeführt, dass ein großer Vorrat an Kandidatentermen mit vielen verschiedenen Werten dieser Parameter konstruiert wurde, aus dem ein Termselektionsalgorithmus möglichst die geeignetsten Terme auswählt. Dabei wird für jede Unterauswahl an Kandidatentermen eine Optimierung durchgeführt, nämlich eine bzgl. der linearen Koeffizienten. Der große Vorteil

dieser Methode ist, dass es sich hierbei um ein konvexes Minimierungsproblem handelt, das somit ein eindeutiges, globales Minimum besitzt und dessen Lösung sich auch im Falle der Ridge-Regression durch einen geschlossenen Ausdruck angeben lässt. Weiterhin kann diese Lösung durch geschickten Einsatz einer Orthogonalisierung der Design-Matrix sehr effizient berechnet werden (vgl. den Abschnitt 3.3 über Termselektion). Werden hingegen auch die Zentren und Breiten in die Optimierung einbezogen, so ist der (regularisierte) SSE nichtkonvex bzgl. dieser Parameter und besitzt viele lokale Minima. Es ist dann meist nicht möglich, das globale Minimum der Kostenfunktion zu bestimmen. Deckt der zur Termselektion herangezogene Kandidatenpool jedoch einen großen Bereich möglicher Zentren und Breiten ab und schafft es der Selektionsalgorithmus, hieraus die signifikanten Terme auszuwählen, so ist die Hoffnung berechtigt, mit dem nach Abschluss der Termselektion vorliegenden Modell zumindest in die Nähe eines Optimums gelangt zu sein. Die Parameter dieses Modells können dann als initiale Werte für eine nichtlineare Optimierung der Zentren und Breiten mit dem Ziel einer weiteren Verbesserung des Modells dienen. Wie in Abschnitt 3.2.1 über die Wahl der Modellarchitektur beschrieben wurde, kann ein RBF-Modell auch als neuronales Netz (RBFNN) aufgefasst werden. Die gradientenbasierte, nichtlineare Optimierung wird in diesem Kontext auch *Back-Propagation* genannt [24]. Im Folgenden wird zunächst die nichtlineare Optimierung des Einschritt-Vorhersagefehlers behandelt, der nichts anderes ist als der (regularisierte) SSE. Diese Bezeichnung entstammt dem Kontext der Zeitreihenvorhersage und bezieht sich auf die einmalige, direkte Vorhersage (3.1) eines zukünftigen Zeitreihenwertes. Anschließend wird die (nur bei der iterierten Zeitreihenvorhersage anwendbare) nichtlineare Optimierung des Mehrschritt-Vorhersagefehlers behandelt. Dabei wird das Modell gemäß (3.2) über mehrere Schritte frei iteriert, wobei die in den einzelnen Iterationen auftretenden Vorhersagefehler akkumuliert werden, um so die der freien Iteration entstammende Zeitreihe über mehrere Zeitschritte hinweg an die gemessene Zeitreihe zu binden.

3.5.1 Minimierung des Einschritt-Vorhersagefehlers

Als zu minimierende Kostenfunktion wird wieder der quadratische Fehler mit einem Regularisierungsterm verwendet:

$$\begin{aligned} J &= \sum_{t=1}^{N}(y_t - g(\boldsymbol{x}_t))^2 + N\lambda \sum_{k=1}^{D+M} |w_k|^q \\ &= \sum_{t=1}^{N}\left(y_t - \sum_{d=1}^{D} w_d x_d - \sum_{k=D+1}^{D+M} w_k \exp\left(-\sum_{d=1}^{D}\frac{(x_d - c_{kd})^2}{r_{kd}^2}\right)\right)^2 \\ &\quad + N\lambda \sum_{k=1}^{D+M} |w_k|^q \,, \end{aligned} \tag{3.135}$$

wobei $q = 1$ (Lasso) oder $q = 2$ (Ridge-Regression). Es wird wieder eine Zentrierung der Ausgabedaten angenommen, so dass der konstante Term w_0 in (3.135) entfällt. Die Minimierung dieser Kostenfunktion wird nun neben den Koeffizienten w_k auch bzgl. der Zentren c_{kd} und Breiten r_{kd} durchgeführt. Da sich die partiellen Ableitungen der Kostenfunktion nach diesen Parametern problemlos berechnen lassen, bietet es sich an, ein gradientenbasiertes Optimierungsverfahren zu benutzen, denn deren Konvergenzgeschwindigkeit ist meist deutlich höher als bei Verfahren, die ohne Gradienteninformationen auskommen. Solche Verfahren arbeiten iterativ, d.h. beginnend mit einem Startwert werden sukzessive neue Näherungen bestimmt, bis der Algorithmus entweder konvergiert oder divergiert (im letzten Fall sollte die Optimierung mit einer anderen Startnäherung neu gestartet werden). In dieser Arbeit wird das Levenberg-Marquardt-Verfahren benutzt, das sich durch besonders gute Konvergenzeigenschaften für quadratische Fehlerfunktionen auszeichnet [62]. Im Falle der Ridge-Regression existieren bzgl. der Optimierung der linearen Modellkoeffizienten zwei Möglichkeiten: Entweder sie werden einfach mitoptimiert, oder sie werden aus dem iterativen Prozess herausgelöst, indem ihre Werte nach jedem Iterationsschritt z.B. mit Hilfe einer QR-Faktorisierung der Design-Matrix berechnet werden, was die Dimensionalität des Optimierungsproblems reduziert.

Die partiellen Ableitungen der RBF-Terme in (3.134) nach den Zentren- bzw.

Breitenkoordinaten berechnen sich zu

$$\begin{aligned}\frac{\partial g_k(\boldsymbol{x}, \boldsymbol{c}_k, \boldsymbol{r}_k)}{\partial c_{kl}} &= 2\frac{x_l - c_{kl}}{r_{kl}^2} \cdot g_k(\boldsymbol{x}, \boldsymbol{c}_k, \boldsymbol{r}_k) \\ \frac{\partial g_k(\boldsymbol{x}, \boldsymbol{c}_k, \boldsymbol{r}_k)}{\partial r_{kl}} &= 2\frac{(x_l - c_{kl})^2}{r_{kl}^3} \cdot g_k(\boldsymbol{x}, \boldsymbol{c}_k, \boldsymbol{r}_k)\,.\end{aligned} \tag{3.136}$$

Damit gilt für die partiellen Ableitungen der Kostenfunktion (3.135) nach diesen Parametern:

$$\begin{aligned}\frac{\partial J}{\partial c_{kl}} &= -4\sum_{t=1}^{N}(y_t - g(\boldsymbol{x}_t))g_k(\boldsymbol{x}_t)w_k\frac{x_{tl} - c_{kl}}{r_{kl}^2} \\ \frac{\partial J}{\partial r_{kl}} &= -4\sum_{t=1}^{N}(y_t - g(\boldsymbol{x}_t))g_k(\boldsymbol{x}_t)w_k\frac{(x_{tl} - c_{kl})^2}{r_{kl}^3}\,,\end{aligned} \tag{3.137}$$

wobei $k = D{+}1, \ldots, D{+}M$, $l = 1, \ldots, D$ und c_{kl} bzw. r_{kl} die l-te Komponente des Zentren- bzw. Breitenvektors des k-ten RBF-Terms bezeichnen. Verschiedene Optimierungsstrategien sind möglich: z.B. simultane Optimierung aller Parameter aller Modellterme oder eine verschachtelte Optimierung, die jeweils nur die Zentren oder Breiten der Modellterme optimiert, oder aber eine zyklische, termweise Optimierung der Modellparameter, bei der immer alle Parameter eines Modellterms gleichzeitig optimiert werden. Im Allgemeinen führt die Anwendung solcher verschiedener Optimierungsstrategien zu unterschiedlichen Ergebnissen, die alle meist lokalen Minima der Kostenfunktion entsprechen, da die nichtlineare Abhängigkeit von den Modellparametern eine sehr komplizierte Fehlerlandschaft mit vielen Nebenminima bedingt, in der ein globales Minimum nur sehr schwer zu finden ist. Die letztgenannte Strategie der zyklischen, termweisen Optimierung stellt sich als die erfolgversprechendste heraus und wird in dieser Arbeit angewendet: Dabei werden alle Zentren- und Breitenkordinaten nur eines Terms simultan optimiert. Nach jedem Iterationsschritt des Optimierers werden die linearen Koeffizienten aller Terme angepasst. Konvergiert die Optimierung der Modellparameter des Terms in einem (lokalen) Minimum, wird zum nächsten Term übergegangen. Wurden auf diese Art und Weise alle Terme einmal optimiert, so ist ein Optimierungszyklus abgeschlossen. Findet während eines Zyklus' eine Veränderung des Modells statt, so wird im Anschluss ein neuer Zyklus gestartet, indem wieder beim ersten Term begonnen wird. Die Optimierung wird beendet, wenn entweder eine festgelegte Obergrenze für die Anzahl der durchlaufenen Zyklen erreicht ist oder wenn in einem kompletten Zyklus keine Veränderung des Modells mehr

erreicht werden kann bzw. die Verkleinerung des Wertes der Kostenfunktion eine festgelegte Toleranzgrenze unterschreitet.

Durch die Optimierung wird eine bessere Anpassung des Modells an die Trainingsdaten erreicht, wobei die Gefahr des Overfittings besteht. Aus diesem Grund ist es ratsam, auch bei der nichtlinearen Optimierung die Cross-Validation einzusetzen. So kann nach jedem Iterationsschritt des Optimierers die neue Näherung auf den Validierungsdaten evaluiert werden. Die Optimierung wird dann u. U. nicht erst durch die Konvergenz gegen ein (lokales) Minimum beendet, sondern sie wird im Minimum des CV-Fehlers abgebrochen. Diese Strategie wird auch als *Early Stopping* bezeichnet [24].

Die Optimierung der Zentren oder Breiten hat zur Folge, dass sich von Iteration zu Iteration die Bilder der Basisfunktionen, also die Spalten $\boldsymbol{g}_k$ der Design-Matrix für die Trainingsdaten, verändern. Dies kann eine drastische Änderung der Korrelationsverhältnisse der Terme untereinander sowie zum Modellierungsziel $\boldsymbol{y}$ zur Folge haben. Dabei können (fast) lineare Abhängigkeiten entstehen, die dazu führen, dass sich einige Terme im Laufe der Optimierung als überflüssig erweisen und die sich z.B. in einem „Herauswandern“ der Zentren aus dem Gebiet der Trainingsdaten im Eingaberaum äußern oder durch extrem große oder kleine Breiten der Gaußfunktionen. Solche Terme werden während der Optimierung entfernt.

3.5.2 Minimierung des Mehrschritt-Vorhersagefehlers

Die bisher betrachteten Kostenfunktionen bzw. Optimierungskriterien basieren alle auf der Minimierung des Einschritt-Vorhersagefehlers und liefern damit keine Informationen darüber, wie sich das Modell bei freier Iteration verhält. Einen Ausweg bietet die Minimierung des Mehrschritt-Vorhersagefehlers, bei der das Modell frei über mehrere Zeitschritte iteriert und die dabei auftretenden Abweichungen von der Zieltrajektorie akkumuliert werden. Auf diese Art und Weise wird versucht, die durch freie Modelliteration erzeugte Trajektorie im rekonstruierten Zustandsraum über mehrere Zeitschritte hinweg an die Zieltrajektorie zu binden.

Zur Vereinfachung der Notation werden im Folgenden alle zu optimierenden Modellparameter im Vektor $\boldsymbol{q} \in \mathbb{R}^n$ zusammengefasst. L sei die Anzahl der freien Iterationen, über die der Vorhersagefehler zu akkumulieren ist. Die zugrundeliegende Zeitreihe sei $\{s_t\}$, aus der durch Delayrekonstruktion mit einer Verzögerung τ und der Einbettungsdimension D rekonstruierte Zustandsraumvektoren

$\boldsymbol{x}_t = (s_t, s_{t-\tau}, \ldots, s_{t-(D-1)\tau}) \in \mathbb{R}^D$ gebildet werden. Die Vorhersageschrittweite sei r Samples (d.h. $\boldsymbol{x}_t$ wird abgebildet auf den Zustand $y_t = s_{t+r}$, $r \in \mathbb{N}$).

Bei der freien Modelliteration fließen Ausgabewerte des Modells in zukünftige Eingaben mit ein. Dies kann, ausgehend vom Start-Eingabevektor $\boldsymbol{x}_t$, beschrieben werden durch die Definition des Vektors $\boldsymbol{\xi}_l^t$ als Eingabevektor im Iterationsschritt l ($l = 0, \ldots, L-1$), der sich für $r = 1$ schreiben lässt als

$$\begin{aligned}
\boldsymbol{\xi}_0^t &= \boldsymbol{x}_t = (s_t, s_{t-\tau}, \ldots, s_{t-(D-1)\tau}) \\
\boldsymbol{\xi}_1^t &= (g(\boldsymbol{\xi}_0^t), s_{t+1-\tau}, \ldots, s_{t+1-(D-1)\tau}) \\
&\vdots \\
\boldsymbol{\xi}_l^t &= (g(\boldsymbol{\xi}_{l-1}^t), g(\boldsymbol{\xi}_{l-1-\tau}^t), \ldots, g(\boldsymbol{\xi}_{l-1-(j-1)\tau}^t), s_{t+l-j\tau}, \ldots, s_{t+l-(D-1)\tau}) \\
&\vdots
\end{aligned} \tag{3.138}$$

Im l-ten Schritt hängen nur die ersten j Komponenten der Eingabe von den Modellparametern ab, wobei gilt $j = \max_k\{l - 1 - (k-1)\tau \geq 0\}$. Für die d-te Komponente von $\boldsymbol{\xi}_l^t$ gilt dann im allgemeinen Fall $r \in \mathbb{N}$

$$\xi_{l,d}^t = \begin{cases} g(\boldsymbol{\xi}_{l-r-(d-1)\tau}^t) & \text{falls } l - r - (d-1)\tau \geq 0, \\ s_{t+l-(k-1)\tau} & \text{falls } l - r - (d-1)\tau < 0 \end{cases}, \quad d = 1, \ldots, D\,. \tag{3.139}$$

Damit lässt sich die zu minimierende Kostenfunktion schreiben als

$$J_L = \frac{1}{L} \sum_{l=0}^{L-1} \sum_{t=1}^{N} \left(y_{t+l} - g(\boldsymbol{\xi}_l^t | \boldsymbol{q})\right)^2 . \tag{3.140}$$

Bei der Bildung der partiellen Ableitungen dieser Kostenfunktion nach den zu optimierenden Parametern muss gemäß der Kettenregel die mögliche Abhängigkeit der einzelnen Komponenten des aktuellen Eingabevektors von den Modellparametern nach (3.139) berücksichtigt werden. Für die Ableitung des Modells nach einem beliebigen Parameter q_i im l-ten Iterationsschritt gilt deshalb

$$\frac{dg(\boldsymbol{\xi}_l^t | \boldsymbol{q})}{dq_i} = \frac{\partial g(\boldsymbol{\xi}_l^t | \boldsymbol{q})}{\partial q_i} + \sum_{d=1}^{D} \left(\frac{\partial g(\boldsymbol{\xi}_l^t | \boldsymbol{q})}{\partial \xi_{l,d}^t} \cdot \frac{\partial \xi_{l,d}^t}{\partial q_i} \right) , \tag{3.141}$$

wobei für die partielle Ableitung der d-ten Komponente der Eingabe nach q_i

analog zu (3.139) gilt

$$\frac{\partial \xi_{l,d}^t}{\partial q_i} = \begin{cases} \frac{d}{dq_i} g(\boldsymbol{\xi}_{l-r-(d-1)\tau}^t | \boldsymbol{q}) & \text{falls } l-r-(d-1)\tau \geq 0, \\ 0 & \text{falls } l-r-(d-1)\tau < 0 \end{cases} . \tag{3.142}$$

Die Ableitung im ersten Fall lautet mit den entsprechenden Indizes wieder genauso wie (3.141). Zu Beginn der freien Iteration ($l = 0$) konstituieren sich die totalen Ableitungen des Modells nach den Parametern nur aus den partiellen Ableitungen nach diesen Parametern, während für $l > r + (d-1)\tau$ durch den zweiten Term in (3.141) noch Ableitungen nach den Komponenten der Eingabe hinzukommen. Insgesamt ergeben sich damit die Ableitungen der Kostenfunktion nach den verschiedenen Modellparametern zu

$$\begin{aligned} \frac{\partial J_L}{\partial w_m} = -\frac{2}{L} \sum_{l=0}^{L-1} \sum_{t=1}^{N} \Bigg[y_{t+l} - g(\boldsymbol{\xi}_l^t) \Bigg] \Bigg[g_m(\boldsymbol{\xi}_l^t) \\ + \sum_{d=1}^{D} \Bigg(w_d + \sum_{k=D+1}^{D+M} w_k \frac{\partial g_k(\boldsymbol{\xi}_l^t)}{\partial \xi_{l,d}^t} \Bigg) \frac{\partial \xi_{l,d}^t}{\partial w_m} \Bigg] \end{aligned} \tag{3.143}$$

$$\begin{aligned} \frac{\partial J_L}{\partial c_{mn}} = -\frac{2}{L} \sum_{l=0}^{L-1} \sum_{t=1}^{N} \Bigg[y_{t+l} - g(\boldsymbol{\xi}_l^t) \Bigg] \Bigg[\frac{\partial g_m(\boldsymbol{\xi}_l^t)}{\partial c_{mn}} \\ + \sum_{d=1}^{D} \Bigg(w_d + \sum_{k=D+1}^{D+M} w_k \frac{\partial g_k(\boldsymbol{\xi}_l^t)}{\partial \xi_{l,d}^t} \Bigg) \frac{\partial \xi_{l,d}^t}{\partial c_{mn}} \Bigg] \end{aligned} \tag{3.144}$$

$$\begin{aligned} \frac{\partial J_L}{\partial r_{mn}} = -\frac{2}{L} \sum_{l=0}^{L-1} \sum_{t=1}^{N} \Bigg[y_{t+l} - g(\boldsymbol{\xi}_l^t) \Bigg] \Bigg[\frac{\partial g_m(\boldsymbol{\xi}_l^t)}{\partial r_{mn}} \\ + \sum_{d=1}^{D} \Bigg(w_d + \sum_{k=D+1}^{D+M} w_k \frac{\partial g_k(\boldsymbol{\xi}_l^t)}{\partial \xi_{l,d}^t} \Bigg) \frac{\partial \xi_{l,d}^t}{\partial r_{mn}} \Bigg] . \end{aligned} \tag{3.145}$$

Die Ableitungen der Basisfunktionen $g_m(\boldsymbol{x}, \boldsymbol{c}_m, \boldsymbol{r}_m)$ nach den c_{mn} bzw. r_{mn} können (3.136) entnommen werden, während sich die Ableitung von g_k nach der d-ten Komponente der Eingabe im l-ten Iterationsschritt zu

$$\frac{\partial g_k(\boldsymbol{\xi}_l^t, \boldsymbol{c}_k, \boldsymbol{r}_k)}{\partial \xi_{l,d}^t} = -2 \frac{\xi_{l,d}^t - c_{kd}}{r_{kd}^2} \cdot g_k(\boldsymbol{\xi}_l^t, \boldsymbol{c}_k, \boldsymbol{r}_k) \tag{3.146}$$

berechnet.

3.6 Einfluss der Termselektion auf das optimierte Modell

In diesem Abschnitt soll untersucht werden, welchen Einfluss eine vorherige systematische Termselektion auf ein RBF-Modell hat, wenn dieses Modell anschließend einer nichtlinearen Optimierung der Zentren und Breiten unterzogen wird. Dazu wird für die zu modellierenden Daten ein großer Pool an Kandidatentermen erzeugt, aus dem zum einen per Forward Selection (FOS) Terme ausgewählt wurden und zum anderen eine Anzahl von Termen einfach zufällig ausgewählt wurden. Natürlich liefert auch die Forward Selection als Greedy-Algorithmus i. Allg. nicht die bestmögliche Lösung. Nichtsdestotrotz erwartet man aber, dass die per FOS erzeugten Modelle besser sind als die per zufälliger Termwahl gewonnenen. Aber sind auch die nichtlinear optimierten FOS-Modelle noch besser als die nichtlinear optimierten Zufallsmodelle? Dies soll hier an numerisch generierten Daten des Chua-Oszillators und der Ramp-Hill-Funktion untersucht werden.

Der Chua-Oszillator ist ein nichtlineares dynamisches System, das beschrieben wird durch das System von Differentialgleichungen

$$\begin{aligned} C_1\dot{V}_{C_1} &= G \cdot (V_{C_2} - V_{C_1}) - g(V_{C_1}) \\ C_2\dot{V}_{C_2} &= G \cdot (V_{C_1} - V_{C_2}) + I_L \\ L\dot{I}_L &= -V_{C_2} \end{aligned} \tag{3.147}$$

mit $g(v) = m_0 v + 1/2(m_1 - m_0)|v + B_p| + 1/2(m_0 - m_1)|v - B_p|$. Die gewählten Parameter waren $C_1 = 1/9,\ C_2 = 1,\ L = 1/7,\ m_0 = -0{,}5,\ m_1 = -0{,}8,\ B_p = 1,\ G = 0{,}67$. Für die Modellierung dieses Systems wurde eine Zeitreihe der Länge 10000 Samples verwendet, auf der die Termselektion durchgeführt wurde. Eine weitere Zeitreihe gleicher Länge diente als Validierungsdatensatz. Die Einbettungsdimension betrug $D = 5$ mit einem Delay $\tau = 4\Delta t$ und einer Vorhersageschrittweite von $\Delta t = 0{,}1$. Die Termselektion wurde jeweils bei verschiedenen, zufällig gewählten Modellgrößen abgebrochen und das Modell anschließend einer nichtlinearen Optimierung des Einschritt-Vorhersagefehlers hinsichtlich aller seiner Parameter (Zentren, Breiten und lineare Koeffizienten) zugeführt. Die Ergebnisse zeigt Abb. 3.8. Zu erkennen ist zum einen, dass die MSE-Werte für die FOS-Methode bei einer festen Modellgröße alle viel weniger stark streuen als diejenigen der Zufallsauswahl. In allen Fällen konnte jedoch durch die nichtlineare Optimierung eine sehr starke Reduktion des Validierungsfehlers erreicht werden. Die Endwerte nach

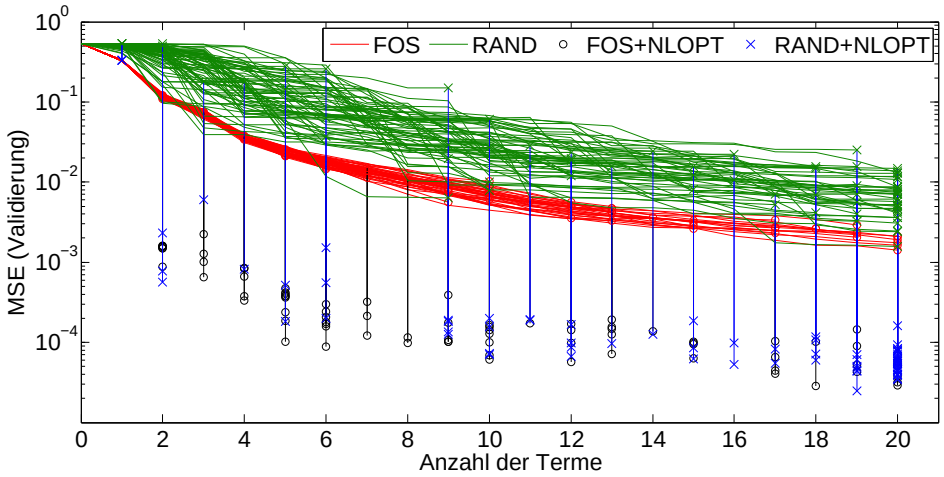


Abbildung 3.8: MSE auf den Testdaten des Chua-Systems für verschiedene Modellgrößen vor und nach der nichtlinearen Optimierung für durch Forward Selection erzeugte Modelle (FOS) und für durch zufällige Termauswahl erzeugte Modelle (RAND). Die vertikalen blauen bzw. schwarzen Linien kennzeichnen die Reduktion des MSE, die für das jeweilige Modell durch die nichtlineare Optimierung erzielt werden konnte.

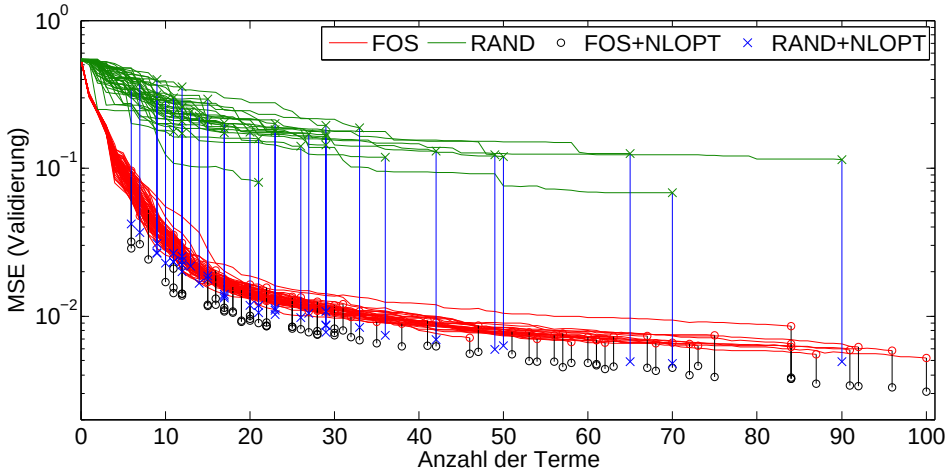


Abbildung 3.9: MSE auf den Testdaten der Ramp-Hill-Funktion für verschiedene Modellgrößen vor und nach der nichtlinearen Optimierung für durch Forward Selection erzeugte Modelle (FOS) und für durch zufällige Termauswahl erzeugte Modelle (RAND).

der Optimierung liegen für die Zufallsauswahl im Bereich der Werte, für die eine initiale Selektion durch FOS erreicht wurden.

Ein ähnliches Bild zeigt sich auch für die Daten der Ramp-Hill-Funktion. Hier wurden 5000 Samples zum Training benutzt und 5000 weitere als Validierungsdaten. Die Ergebnisse zeigt Abb. 3.9. Insbesondere bei kleinen Modellgrößen ergab

sich hier noch ein gewisser Vorteil einer systematischen Termselektion gegenüber der Zufallsauswahl, aber auch in diesem Fall bleibt der Unterschied nach Abschluss der nichtlinearen Optimierung relativ gering.

Kapitel 4

Modellierung von Parameterabhängigkeiten

Physikalische Systeme verfügen in der Regel über Parameter wie z.B. Druck, Temperatur, Ohm'scher Widerstand usw., durch die ihr Verhalten beeinflusst wird. Experimente haben u. a. zum Ziel, die Abhängigkeit des Systems von seinen Parametern zu bestimmen. Im Folgenden wird angenommen, dass es sich dabei um ein deterministisches dynamisches System handelt (vgl. Abschnitt 2.1). Die Zeitentwicklung eines solchen Systems wird durch den Fluss $\boldsymbol{\phi}$ (2.3) beschrieben, und die Parameterabhängigkeit ist hier gegeben durch die Abhängigkeit des Flusses von den Parametern $\boldsymbol{p}$. Sie äußert sich durch ihren Einfluss auf Lage und Geometrie von Attraktoren der Dynamik und auf deren Kenngrößen wie die Attraktordimension oder die Werte von Lyapunov-Exponenten. Die Parameter spannen den sogenannten *Parameterraum* auf. Die Bifurkationsanalyse [66, 67] untersucht diesen Raum, um in ihm räumliche Untermengen mit unterschiedlichem dynamischen Verhalten zu identifizieren, deren Berandungen die Orte von Bifurkationen wiedergeben. Handelt es sich nur um einen skalaren Parameter, so lässt sich das dynamische Verhalten des Systems in Abhängigkeit des Parameters grafisch mit Hilfe eines Bifurkationsdiagramms darstellen. Abb. 4.1 zeigt ein solches Bifurkationsdiagramm für die eindimensionale Gaußabbildung. Im weiteren Verlauf wird immer angenommen, dass die Gleichungen des betrachteten dynamischen Systems unbekannt sind und damit auch die funktionelle Form der Abhängigkeit des Flusses von den Parametern $\boldsymbol{p}$. Stattdessen werden lediglich Zeitreihen des Systems verwendet, die für verschiedene Werte der Parameter erzeugt bzw. aufgezeichnet wurden. Das Ziel ist dann, aus diesen Zeitreihen ein Modell zu konstruieren, das nicht nur die Dynamik des zugrundeliegenden Systems für eine gegebene Parameterkonstellation beschreibt, sondern auch die Abhängigkeit der Dynamik von den Parametern, und das somit auch in der Lage ist, die Systemdynamik für andere als die zum Training verwendeten Parameterwerte zu beschreiben. Ein solches

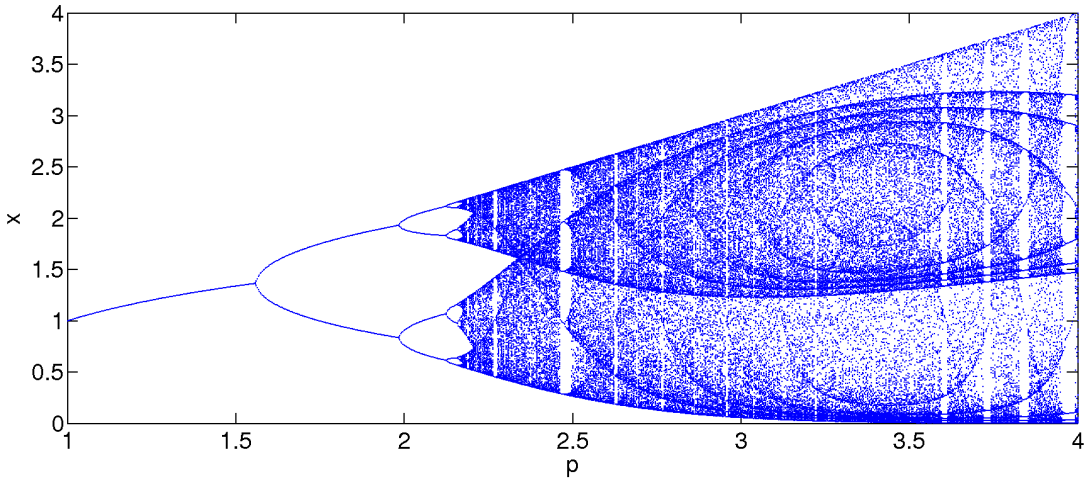


Abbildung 4.1: Bifurkationsdiagramm der Gaußabbildung $x_{n+1} = p \cdot \exp\bigl(-(x_n - 1)^2\bigr)$

Modell soll also gute Generalisierungseigenschaften sowohl in Bezug auf die Eingabedaten als auch in Bezug auf die Parameter besitzen. War bisher immer die Schätzung der Regression $\mathrm{E}[y|\boldsymbol{x}]$ durch ein Modell $g(\boldsymbol{x})$ das Ziel, so besteht dies nun in der Schätzung der zusätzlich von den Parametern $\boldsymbol{p}$ abhängigen Regression $\mathrm{E}[y|\boldsymbol{x}, \boldsymbol{p}]$ durch ein Modell $g(\boldsymbol{x}, \boldsymbol{p})$.

Von grundsätzlicher Bedeutung für die Vorgehensweise bei der Modellierung ist die Kenntnis der Parameter, d.h. ob Anzahl und Werte der Parameter bekannt sind oder nicht. Im letzten Fall müssen diese aus den verwendeten Zeitreihen selbst rekonstruiert werden. Weiterhin muss unterschieden werden zwischen unter stationären Bedingungen gewonnenen Zeitreihen, wo die Parameter während der Messung einer Zeitreihe jeweils konstant waren, und unter instationären Bedingungen gewonnenen Zeitreihen mit während der Messung veränderlichen Parameterwerten. In diesem letzten Fall besteht das Modellierungsziel darin, aus den instationären Zeitreihen Aussagen über die stationäre Dynamik bei festen Parameterwerten zu gewinnen.

In der Literatur werden im Wesentlichen zwei Ansätze zur Modellierung von Parameterabhängigkeiten beschrieben: die Verwendung einer parametrisierten Familie von Modellen [68–74] und die Konstruktion von Modellen mit erweitertem Zustandsraum [12, 74, 75]. Langer gibt in [35] und [74] eine ausführliche Beschreibung beider Varianten und stellt Anwendungsmöglichkeiten für die verschiedenen oben beschriebenen Fälle (bekannte/unbekannte Parameter, stationäre/instationäre Dynamik) dar. Er kommt zum Ergebnis, dass die Methode der erweiterten Zustandsraumvektoren (wenn anwendbar, s. u.) üblicherweise bessere

Ergebnisse liefert als die Methode der parametrisierten Familie von Modellen. In dieser Arbeit werden daher ausschließlich Modelle mit erweitertem Zustandsraum verwendet. In den folgenden beiden Abschnitten sollen nun beide Methoden kurz erläutert werden. Für eine ausführliche Darstellung siehe [35].

4.1 Parametrisierte Familien von Modellen

Es seien Zeitreihen $\{s_t^{(u)}|t = 1,\ldots,N\}$ für die verschiedenen Parameterwerte $\boldsymbol{p}_u \in \mathbb{R}^B$, $u = 1,\ldots,U$ gegeben. Die Modellierung der Dynamik und der Parameterabhängigkeit erfolgt hier in zwei Schritten. Zuerst wird eine Familie von Modellen konstruiert, die alle die gleiche Funktionenbasis verwenden und sich lediglich in ihren linearen Koeffizienten unterscheiden. Für ein Modell $g(\boldsymbol{x}) = \sum_{k=1}^{M} w_k g_k(\boldsymbol{x}, \boldsymbol{c}_k, \boldsymbol{r}_k)$ einer Linearkombination von radialen Basisfunktionen bedeutet das, dass sich das Modell schreiben lässt als

$$g(\boldsymbol{x}|\boldsymbol{w}(\boldsymbol{p}_u)) = \sum_{k=1}^{M} w_k(\boldsymbol{p}_u) g_k(\boldsymbol{x}, \boldsymbol{c}_k, \boldsymbol{r}_k)\,. \tag{4.1}$$

Die Annahme ist hier, dass stetige Veränderungen der Parameter $\boldsymbol{p}$ zu stetigen Veränderungen der Modellkoeffizienten führen. In einem zweiten Schritt erfolgt nun die Modellierung der eigentlichen Parameterabhängigkeit, d. h. hier der Abhängigkeit der Modellkoeffizienten von den Parametern: $\mathbb{R}^B \to \mathbb{R}^M$, $\boldsymbol{p} \mapsto w_k(\boldsymbol{p})$, $k = 1,\ldots,M$, wofür Langer [35] ein Polynom niedrigen Grades verwendet. Im Allgemeinen ist $M > B$ oder sogar $M \gg B$, und die Koeffizienten $\boldsymbol{w}$ liegen im M-dimensionalen Koeffizientenraum auf einer B-dimensionalen Untermannigfaltigkeit (siehe dazu auch [76]). Hat man eine Approximation der Abhängigkeit $\boldsymbol{w}(\boldsymbol{p})$ bestimmt, so lässt sich diese verwenden, um die Dynamik des zugrundeliegenden Systems bei einem anderen Parameterwert $\tilde{\boldsymbol{p}}$ als den zum Training verwendeten vorherzusagen, indem zuerst die Modellkoeffizienten $\boldsymbol{w}(\tilde{\boldsymbol{p}})$ bestimmt und diese dann in das Modell (4.1) eingesetzt werden. Die Schwierigkeit besteht bei dieser Methode darin, eine Funktionenbasis $\{g_1(\boldsymbol{x}),\ldots,g_M(\boldsymbol{x})\}$ zu finden, die eine gleichermaßen gute Approximation der Dynamiken aller Zeitreihen erlaubt. Dies gestaltet sich besonders schwierig, wenn der abgedeckte Parameterbereich so groß ist, dass die rekonstruierten Attraktoren kaum noch räumliche Überdeckungen aufweisen und sich stark in ihrem Zustandsraumvolumen unterscheiden. Probleme können sich allerdings auch schon dann einstellen, wenn der von den Trainingszeitreihen abgedeckte Parameterbereich sowohl chaotische als

auch periodische Orbits enthält. Dann kann es passieren, dass die zu periodischen Orbits gehörenden Zeitreihen zu wenige *verschiedene* Bereiche im rekonstruierten Zustandsraum abdecken und das Problem der Koeffizientenschätzung unterbestimmt ist. LANGER zeigt in [35] und [74], dass sich das Problem abmildern lässt, indem Zeitreihen verwendet werden, die nicht von eingeschwungenen Systemen stammen, sondern noch Transienten enthalten.

4.2 Modelle mit erweitertem Zustandsraum

Die hier beschriebene Methode geht auf eine Idee von CASDAGLI zur Rekonstruktion eines Bifurkationsdiagrammes aus einer durch einen Parametersweep[1] erzeugten, instationären Zeitreihe zurück [12]. Sie lässt sich auch zur Modellierung der Parameterabhängigkeit aus stationären Zeitreihen verwenden. Es seien also wieder Zeitreihen $\{s_t^{(u)} | t = 1, \ldots, N\}$ für die verschiedenen Parameterwerte $\boldsymbol{p}_u \in \mathbb{R}^B$, $u = 1, \ldots, U$ gegeben. Die Idee bei dieser Methode ist, nur *ein* Modell aus allen Zeitreihen zu konstruieren, das gleichzeitig die Dynamik des Systems und deren Abhängigkeit von den Parametern beschreibt. Um im rekonstruierten Zustandsraum zwischen Zuständen und Attraktoren zu verschiedenen Parameterwerten unterscheiden zu können, wird dieser um die Komponenten der Parametervektoren erweitert:

$$\tilde{\boldsymbol{x}}_t = (\boldsymbol{x}_t, \boldsymbol{p}) = (s_t^{(u)}, s_{t-\tau}^{(u)}, \ldots, s_{t-(D-1)\tau}^{(u)}, p_{u,1}, \ldots, p_{u,B}) \in \mathbb{R}^{D+B} \,. \tag{4.2}$$

Dabei sind immer die zu den Zeitreihensamples in den ersten D Komponenten gehörenden Parameterwerte hinten anzuhängen. Auf diese Art und Weise ergibt sich im erweiterten rekonstruierten Zustandsraum automatisch eine geometrische Anordnung der Zeitreihen, die dazu führt, dass zu benachbarten Parameterwerten gehörende Zeitreihen auch im erweiterten rekonstruierten Zustandsraum geometrisch benachbart sind. Das zu konstruierende Modell ist nun eine Abbildung

$$g : \mathbb{R}^D \times \mathbb{R}^B \to \mathbb{R}, \quad \tilde{\boldsymbol{x}}_t \mapsto \hat{y}_t = g(\tilde{\boldsymbol{x}}_t) \,. \tag{4.3}$$

Statt der Konstruktion einer parametrisierten Schar von Funktionen besteht die Modellierung nun in der Approximation einer $(D + B)$-dimensionalen Hyperfläche im $\mathbb{R}^{D+B+1}$. Abb. 4.2 verdeutlicht dies grafisch am Beispiel der Gaußabbildung, für die $D = B = 1$ ist. Beim Training des Modells, genauer bei der

[1] d.h. der Parameter steigt oder fällt während der Erzeugung der Zeitreihe linear mit der Zeit

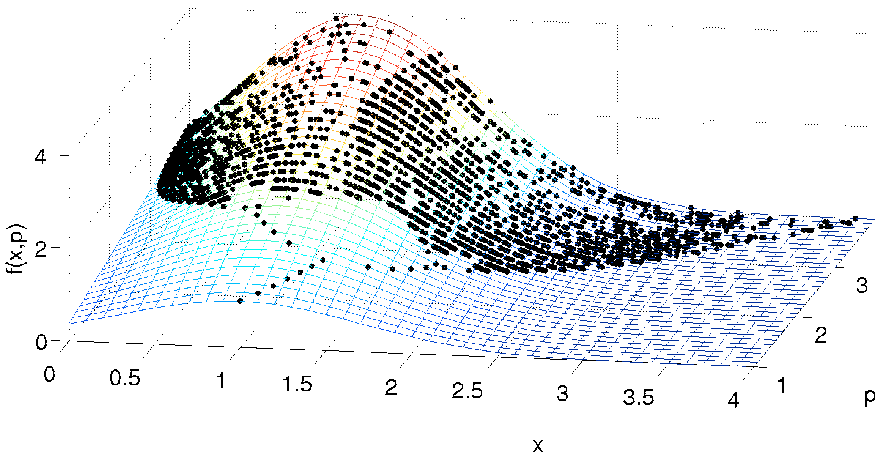


Abbildung 4.2: Der Fluss $f(x,p) = p \cdot \exp\bigl(-(x-1)^2\bigr)$ (durch das Gitter angedeutete Fläche) der Gaußabbildung als Funktion des Systemzustandes und des Parameters p. Die schwarzen Punkte auf den Gitterlinien parallel zur x-Achse sind die Zeitreihensamples zum entsprechenden Wert von p.

Termselektion und der anschließenden nichtlinearen Optimierung des Einschritt-Vorhersagefehlers spielt es keine Rolle, dass es sich bei den letzten B Komponenten der erweiterten Zustandsraumvektoren um Parameterwerte und nicht um Samples der Zeitreihe handelt. Die RBF-Terme sind bei der nichtlinearen Optimierung der Zentren und Breiten in allen $D+B$ Richtungen verschieb- und skalierbar. Lediglich bei der freien Iteration eines solchen Modells und damit auch bei der nichtlinearen Optimierung des Mehrschritt-Vorhersagefehlers zeigt sich ein Unterschied zur Modellierung mit ausschließlich aus Zeitreihensamples bestehenden Eingabevektoren, denn die Dynamik bzgl. eines festen Parametervektors $\boldsymbol{p}_0$ spielt sich nun nicht im gesamten erweiterten Zustandsraum ab, sondern nur in einem D-dimensionalen Unterraum $V_{\boldsymbol{p}_0} \subset \mathbb{R}^{D+B}$. Bei der freien Iteration des Modells müssen daher die Parameterkomponenten festgehalten werden, d.h. mit der Vorhersage $\hat{s}_{t+1} = \hat{y}_t = g(\tilde{\boldsymbol{x}}_t)$ für die Eingabe $\tilde{\boldsymbol{x}}_t = (s_t, s_{t-\tau}, \ldots, s_{t-(D-1)\tau}, p_1, \ldots, p_B)$ ist der nächste Eingabevektor des Modells $\hat{\tilde{\boldsymbol{x}}}_{t+1} = (\hat{s}_{t+1}, s_{t+1-\tau}, \ldots, s_{t+1-(D-1)\tau}, p_1, \ldots, p_B)$. Es ändern sich also nur die aus den Zeitreihensamples bestehenden ersten D Komponenten der Eingabe, während die übrigen B Komponenten mit den Parameterwerten festgehalten werden.

Ein großer Vorteil dieser Methode gegenüber den im vorigen Abschnitt besprochenen parametrisierten Familien ist, dass zur Approximation der Dynamik bei einem Parameterwert durch das geordnete Nebeneinander der Zeitreihen im er-

weiterten Zustandsraum auch Informationen aus den Zeitreihen zu benachbarten Parameterwerten herangezogen werden können. Dies reduziert insbesondere die Unterbestimmtheitsproblematik bei der Modellierung eingeschwungener periodischer Zeitreihen, wie sie bei den parametrisierten Familien auftreten kann. Weiterhin werden aus diesem Grund auch weniger Trainingsdaten pro Parameterwert benötigt. Im Vergleich zur Methode der parametrisierten Familien steigt die Dimension des Eingaberaumes zwar um die Anzahl der Parameterkomponenten, gleichzeitig steht jedoch die U-fache Anzahl an Trainingsdaten zur Konstruktion des Modells zur Verfügung. Die Modellierung der Dynamik auch für andere als zum Training verwendete Parameterwerte ist eine inhärente Eigenschaft der Methode der erweiterten Zustandsraumvektoren, da die durch die Modellierung approximierte $(D + B)$-dimensionale Hyperfläche auch die Parameterkomponenten enthält (vgl. Abb. 4.2). In Abschnitt 3.2.1 über die Wahl der Modellarchitektur wurde dargelegt, dass es nötig ist, die RBF-Terme so über den Eingaberaum zu verteilen und in ihrer Breite zu skalieren, dass zwischen benachbarten Termen eine gewisse Überlappung herrscht. Damit die Interpolation der Dynamik zwischen zum Training verwendeten Parameterwerten sinnvolle Ergebnisse liefern kann, muss eine solche Überlappung auch bzgl. der Richtungen der Parameterachsen herrschen. Aus numerischen Gründen und um dies mit ähnlichen Breitenskalierungen bzgl. aller Komponenten des Eingaberaumes erreichen zu können, ist es sinnvoll, die Zeitreihen sowie die Parameterwerte ggf. so zu skalieren, dass die Ausdehnung der Daten im erweiterten rekonstruierten Zustandsraum in allen $D + B$ Raumrichtungen (annähernd) gleich ist. Eine Vorhersage der Dynamik für andere als zum Training verwendete Parameterwerte erhält man schließlich einfach durch freie Iteration des Modells, indem die gewählten Parameterwerte in die letzten B Komponenten der $\tilde{\boldsymbol{x}}_t$ eingesetzt werden. Abb. 4.3 zeigt rekonstruierte Bifurkationsdiagramme des Chua-Oszillators (3.147), die mit der Methode der erweiterten Zustandsraumvektoren aus neun Zeitreihen an den durch rot gestrichelte Linien markierten Werten des Parameters G berechnet wurden. Für diese 9 Zeitreihen, die aus jeweils 3000 Samples bestanden, wurde ein RBF-Modell aus 50 Basisfunktionen konstruiert, die mit dem FOS-Algorithmus aus einem Termvorrat von 7749 Kandidatentermen ausgewählt wurden. Die Zentren dieser Kandidaten wurden zufällig den Trainingsdaten entnommen, als Breiten wurden zufällig gewählte Werte aus dem Intervall des 0,1- bis 3-fachen des mittleren euklidischen Abstandes zwischenden Kandidaten gewählt. Zum Vergleich ist in Abb. 4.3 oben ein durch numerische Integration des Chua-Systems berechnetes Original-Bifurkationsdiagramm dargestellt. Das zweite Diagramm wurde durch

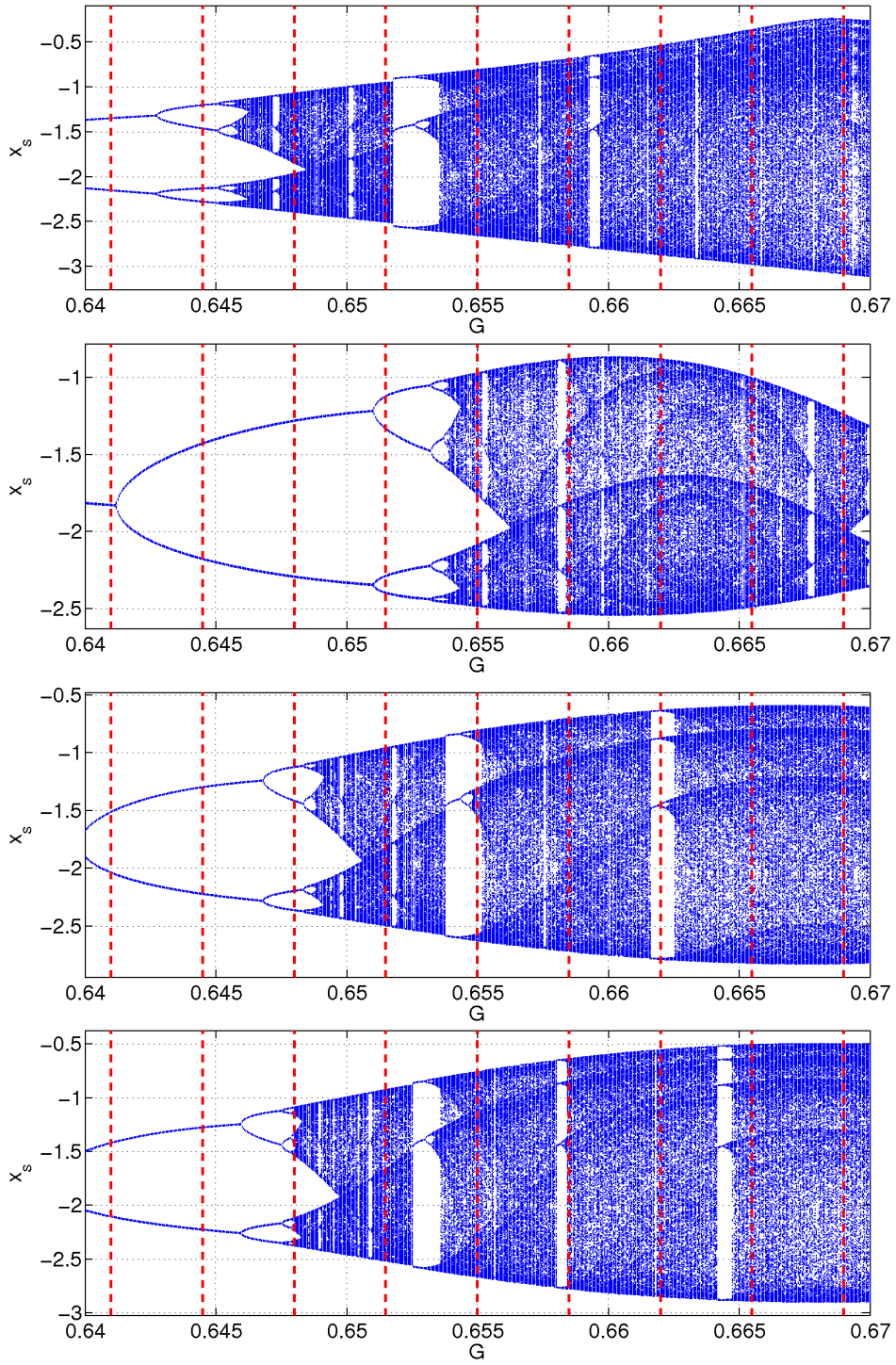


Abbildung 4.3: Oben: numerisch erzeugtes Original-Bifurkationsdiagramm des Chua-Oszillators. Darunter: rekonstruierte Bifurkationsdiagramme eines RBF-Modells nach Abschluss der Termselektion (2. Zeile), nach Abschluss der nichtlinearen Optimierung der Zentren und Breiten (3. Zeile) bzw. nach Minimierung des Mehrschritt-Vorhersagefehlers (4. Zeile). Die rot gestrichelten Linien geben die Parameterwerte an, für die Zeitreihen für das Training des Modells verwendet wurden.

freie Iteration des Modells gewonnen, wie es nach Abschluss der Termselektion vorlag. Anschließend wurde dieses Modell einer nichtlinearen Minimierung des Einschritt-Vorhersagefehlers bzgl. der Zentren und Breiten der RBF-Terme unterzogen, die hier zyklisch, d.h. termweise erfolgte (vgl. Abschnitt 3.5.1). Diese Optimierung bewirkte eine Reduktion des MSE auf den Validierungsdaten auf 32,1% des initialen Wertes. Nach Abschluss dieser Optimierung erfolgte wieder eine Rekonstruktion des Bifurkationsdiagrammes mit dem optimierten Modell, das in der dritten Zeile zu sehen ist. Schließlich wurde dieses Modell nochmals optimiert, dieses mal durch Minimierung des Mehrschritt-Vorhersagefehlers mit einer Akkumulation der Vorhersagefehler über 20 Iterationsschritte. Der Wert dieses 20-Schritt-Vorhersagefehlers konnte durch die Optimierung um 17,3% gesenkt werden. Das nach Abschluss dieser Optimierung rekonstruierte Bifurkationsdiagramm ist in der letzten Zeile in Abb. 4.3 dargestellt. Schon das nach Abschluss der Termselektion rekonstruierte Diagramm zeigt qualitativ viele Merkmale der Originaldynamik, wobei die Periodenverdopplungen bei deutlich zu großen Werten des Bifurkationsparameters G auftreten. Die nichtlinearen Optimierungen bringen eine sukzessive verbesserte Übereinstimmung der Modell- mit der Originaldynamik. Allerdings sind auch beim Mehrschritt-optimierten Modell noch deutliche Abweichungen sichtbar, die Periodenverdopplung $2 \rightarrow 4$ tritt z.B. verzögert (d.h. bei zu großem G) auf, und insbesondere ist der rekonstruierte Modell-Attraktor beim Parameterwert 0,6445, für den Trainingsdaten vorlagen, ein Orbit mit Periode 2, während der Originalattraktor dort die Periode 4 aufweist. Da die Dynamik des Systems empfindlich vom Wert von G abhängt, wie an den plötzlich auftretenden periodischen Fenstern inmitten eines chaotischen Regimes zu sehen ist, ist es zwar verständlich, dass die Lage eines solchen Fensters nicht exakt vom Modell reproduziert werden kann, insbesondere wenn es zwischen zwei Parameterstützstellen mit jeweils chaotischer Dynamik liegt. Allerdings würde man schon erwarten, dass das Modell wenigstens an den Parameterwerten, für die Trainingsdaten in Form von Zeitreihen vorliegen, die Dynamik qualitativ korrekt wiedergibt. Dieses Problem und Ansätze zur Lösung werden im nächsten Abschnitt behandelt.

4.3 Bewertung von rekonstruierten Bifurkationsdiagrammen

Im vorigen Abschnitt wurde erläutert, wie sich aus Zeitreihen zu verschiedenen Parametern eines dynamischen Systems ein Modell konstruieren lässt, das sowohl die Dynamik des zugrundeliegenden Systems als auch die Abhängigkeit dieser Dynamik von den Parametern approximiert. Im Falle eines skalaren Parameters kann diese Abhängigkeit vom Parameter mit Hilfe eines Bifurkationsdiagrammes dargestellt werden. Zur Rekonstruktion des Bifurkationsdiagramms kann dann das Modell verwendet werden, indem es für viele verschiedene Parameterwerte frei iteriert wird und aus den so enstandenen Zeitreihen durch Delay-Rekonstruktion und anschließende Berechnung von Poincaré-Schnitten ein Bifurkationsdiagramm erzeugt wird. Dabei tritt immer wieder der Fall ein, dass das rekonstruierte Bifurkationsdiagramm zwar qualitativ eine gute Übereinstimmung mit dem wahren Bifurkationsdiagramm aufweist und z.B. auch Periodenverdopplungen oder periodische Fenster innerhalb chaotischer Regimes auftreten. Sehr oft sind diese aber auf der Parameterachse gegenüber den wahren Orten verschoben, treten also bei etwas anderen Werten der Parameter auf. Häufig liefert das Modell sogar qualitativ falsche Dynamiken an den Stellen im Parameterraum, für die Stützstellen in Form von Trainingszeitreihen zur Verfügung standen, d.h. das Modell liefert bei freier Iteration an einzelnen Parameterorten z.B. periodische Orbits, obwohl die Trainingszeitreihe an diesem Parameterort einem chaotischen Attraktor entstammt und umgekehrt. Das Problem liegt darin, dass der bei der Modellkonstruktion durch Termselektion und anschließend nichtlinear minimierte *Einschritt*-Vorhersagefehler keinen guten Anhaltspunkt für das Verhalten des Modells bei freier Iteration liefert. Bei zwei auf den gleichen Trainingsdaten konstruierten Modellen $g_1(\boldsymbol{x})$ und $g_2(\boldsymbol{x})$ bedeutet ein kleinerer Einschritt-Vorhersagefehler von $g_1(\boldsymbol{x})$ gegenüber dem von $g_2(\boldsymbol{x})$ *nicht* automatisch, dass die Langzeitdynamik von $g_1(\boldsymbol{x})$ bei freier Iteration besser mit der Dynamik des gemessenen Systems übereinstimmt als die von $g_2(\boldsymbol{x})$. Durch Minimierung des Mehrschritt-Vorhersagefehlers (vgl. Abschnitt 3.5.2) kann man zwar versuchen, die Zeitreihe des frei iterierten Modells über mehrere Schritte hinweg an die Trainingsdaten zu binden und so eine bessere Übereinstimmung der rekonstruierten Attraktoren zu erreichen, jedoch ist dies wegen der sensitiven Abhängigkeit chaotischer Systeme von den Anfangsbedingungen und sich exponentiell mit der Zeit verstärkender kleinster Abweichungen nur über eine sehr begrenzte Anzahl an

Vorhersageschritten möglich. Auch diese begrenzte Anzahl freier Iterationen liefert oft keine zuverlässige Prognose über das Langzeitverhalten des frei iterierten Modells.

Es seien nun eine durch Messungen an einem dynamischen System gewonnene Trainings- oder Originalzeitreihe gegeben sowie eine durch freie Iteration eines auf diesen Trainingsdaten konstruierten Modells erzeugte Modellzeitreihe. Es wird ein Maß benötigt, das Aussagen über die Unterschiede der dynamischen Eigenschaften des Systems und des frei laufenden Modells auf der Basis dieser Zeitreihen liefert. Damit solche Aussagen zuverlässig getroffen werden können, müssen die Zeitreihen hinreichend lang sein und eine gute Abdeckung des Attraktors im rekonstruierten Zustandsraum liefern. Außerdem müssen die Zeitreihen natürlich frei von Transienten sein. Ein Sample-weiser Vergleich der Modell- mit der Originalzeitreihe ist wegen der sensitiven Abhängigkeit von den Anfangsbedingungen bei chaotischen Systemen nicht sinnvoll. In dieser Arbeit werden daher Möglichkeiten vorgeschlagen, direkt die Attraktoren im rekonstruierten Zustandsraum zu vergleichen und ein Maß für ihre Verschiedenheit zu finden. Die erste Möglichkeit basiert auf dem Vergleich der empirischen Wahrscheinlichkeitsverteilungen der Punktmengen im rekonstruierten Zustandsraum und verwendet die *Kullback-Leibler-Divergenz* [77, 78] zur Berechnung der Unterschiedlichkeit beider Verteilungen. Sie wird im Folgenden als KLD-Methode bezeichnet. Die zweite Methode ist eher heuristisch motiviert und basiert auf der Berechnung von Nächste-Nachbar-Abständen zwischen den Punktwolken. Diese Methode wird im Folgenden als NN-Methode bezeichnet. Die Verfahren werden in den beiden nächsten Unterabschnitten beschrieben.

4.3.1 KLD-Methode zur Bestimmung der Attraktor-Diskrepanz

Es seien $\boldsymbol{x}_t \in \mathbb{R}^D$ die durch Delay-Einbettung gewonnenen rekonstruierten Zustandsraumvektoren der Originalzeitreihe und $\hat{\boldsymbol{x}}_t \in \mathbb{R}^D$ die Delayvektoren, die aus dem frei iterierten Modell gewonnen wurden. P und $\hat{P}$ seien die Wahrscheinlichkeitsverteilungen von $\boldsymbol{x}_t$ bzw. $\hat{\boldsymbol{x}}_t$ und $p(\boldsymbol{u})$ bzw. $\hat{p}(\boldsymbol{u})$ die entsprechenden Wahrscheinlichkeitsdichtefunktionen. Dann ist die Kullback-Leibler-Divergenz (KLD) definiert durch

$$\mathrm{KLD}(P, \hat{P}) = \int\limits_{\mathbb{R}^D} p(\boldsymbol{u}) \log \left(\frac{p(\boldsymbol{u})}{\hat{p}(\boldsymbol{u})} \right) \mathrm{d}\boldsymbol{u} \,. \tag{4.4}$$

Die KLD ist ein Maß für die Verschiedenheit zweier Wahrscheinlichkeitsverteilungen. Aus informationstheoretischer Sicht beschreibt sie die Anzahl zusätzlicher Bits, die zur Codierung von Samples von P benötigt werden, wenn der Code auf $\hat{P}$ basiert. Sie wird oft als Abstand zwischen den beiden Wahrscheinlichkeitsverteilungen bezeichnet, obwohl es sich bei der KLD streng genommen nicht um eine Metrik handelt. Die KLD ist nicht symmetrisch, d.h. i. Allg. ist $\mathrm{KLD}(P, \hat{P}) \neq \mathrm{KLD}(\hat{P}, P)$.

Die Idee ist nun, die KLD zur Definition der *Diskrepanz* zwischen den Attraktoren zu verwenden, und wird motiviert durch die Tatsache, dass zwei Zeitreihen desselben Systems (z.B. die einer Messung und die eines perfekten Modells) zwar verschiedene Werte aufweisen können, im rekonstruierten Zustandsraum aber der gleichen Wahrscheinlichkeitsdichteverteilung gehorchen. Sind die Attraktoren im rekonstruierten Zustandsraum jedoch verschieden, so unterscheiden sich auch die zugrundeliegenden Wahrscheinlichkeitsdichteverteilungen. Letztere müssen nun zuerst für beide Attraktoren geschätzt werden. Hierzu kann der rekonstruierte Zustandsraum z.B. in Hyperkuben eingeteilt und ein *box counting* durchgeführt werden, d.h. man bestimmt in jedem Hyperkubus die Anzahl der in ihm liegenden rekonstruierten Zustände und normiert diesen Wert mit der Gesamtanzahl der Delayvektoren. Auf diese Art und Weise ergibt sich eine diskontinuierliche Approximation der Verteilung. Besser geeignet sind auf Kernfunktionen basierende Verfahren, die die Verteilung z.B. mittels Gaußfunktionen glatt approximieren. Solche Verfahren werden als Kerndichteschätzer bezeichnet (engl. *kernel density estimation*, KDE, [24]). Im Rahmen dieser Arbeit wird zur Schätzung der Wahrscheinlichkeitsdichten die KDE-Toolbox von Alexander Ihler [79] verwendet.

Basierend auf den so geschätzten empirischen Wahrscheinlichkeitsverteilungen wird nun die *Attraktordiskrepanz* definiert durch

$$\begin{aligned} d_{\mathrm{KLD}} &= \mathrm{KLD}(P, \hat{P}) + \mathrm{KLD}(\hat{P}, P) \\ &\equiv d_{\mathrm{KLD}}(\mathbf{X}, \hat{\mathbf{X}}) + d_{\mathrm{KLD}}(\hat{\mathbf{X}}, \mathbf{X})\,. \end{aligned} \tag{4.5}$$

Der erste Term der zweiten Zeile in (4.5) dient als Maß für die Diskrepanz der Verteilung der $\hat{\boldsymbol{x}}_t$ zur Verteilung der $\boldsymbol{x}_t$, während der zweite Term als Maß für die Diskrepanz der Verteilung der $\boldsymbol{x}_t$ zu der der $\hat{\boldsymbol{x}}_t$ dient.

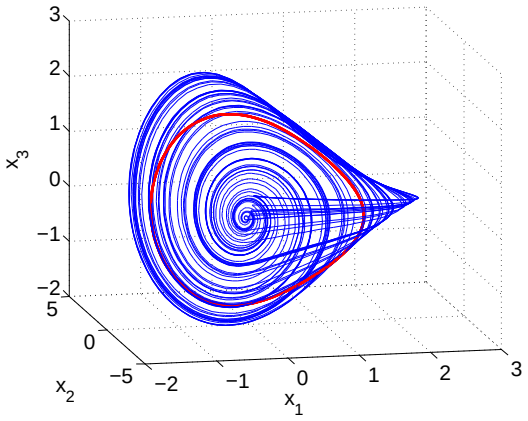


Abbildung 4.4: Gezeigt sind jeweils 5000 Punkte eines periodischen Orbits (rot) und eines chaotischen Attraktors (blau) des Rössler-Oszillators. Dieses Beispiel verdeutlicht die Asymmetrie der NN-Abstände: Zu jedem Punkt des periodischen Orbits existiert ein nächster Nachbar aus der Menge des chaotischen Attraktors mit einem kleinen euklidischen Abstand. Umgekehrt sind die Abstände meist deutlich größer.

4.3.2 NN-Methode zur Bestimmung der Attraktor-Diskrepanz

Die hier vorgestellte NN-Methode zur Quantifizierung der Verschiedenheit zweier Attraktoren basiert auf den euklidischen Abständen der Punkte des einen Attraktors zu deren nächsten Nachbarn aus der Menge der Punkte des anderen Attraktors und folgt einer einfachen Intuition: Wenn die zwei Attraktoren identisch sind und von beiden Zeitreihen hinreichend viele rekonstruierte Zustände $\boldsymbol{x}_t$ bzw. $\hat{\boldsymbol{x}}_t$ vorliegen, die jeweils den gesamten Attraktor bedecken, so sollte es zu jedem der Zustände $\boldsymbol{x}_t$ einen nächsten Nachbarn aus der Menge der $\hat{\boldsymbol{x}}_t$ geben, der einen kleinen Abstand von $\boldsymbol{x}_t$ hat, wobei mit „kleiner Abstand“ gemeint ist, dass dieser Abstand klein ist im Vergleich zur gesamten Ausdehnung des Attraktors. Umgekehrt sollte im Falle gleicher Attraktoren auch zu jedem der $\hat{\boldsymbol{x}}_t$ ein nächster Nachbar mit kleinem Abstand aus der Menge der $\boldsymbol{x}_t$ existieren. Unterscheiden sich jedoch beide Attraktoren deutlich in ihrer Gestalt, so werden auch die NN-Abstände größer sein. Wie bei der Kullback-Leibler-Divergenz sind auch diese Abstände zwischen nächsten Nachbarn i. Allg. asymmetrisch. Abb. 4.4 zeigt dies an einem Beispiel: Dargestellt sind dort zwei rekonstruierte Attraktoren aus numerisch generierten, aus jeweils 5000 Samples bestehenden Zeitreihen des Rössler-Oszillators, von denen der eine chaotisch und der andere periodisch ist. Zu jedem Punkt des periodischen Attraktors existiert ein nächster Nachbar aus der Menge des chaotischen Attraktors mit einem kleinen euklidischen Abstand. Der Mittelwert dieser 5000 NN-Abstände beträgt 0,056. Umgekehrt sind die Abstände der 5000 Punkte des chaotischen Attraktors zu ihren nächsten Nachbarn aus der Menge des periodischen Attraktors überwiegend deutlich größer, der Mittelwert beträgt hier 0,75.

Im Folgenden wird nun die Diskrepanz der beiden Attraktoren der $\boldsymbol{x}_t$ und $\hat{\boldsymbol{x}}_t$

definiert. Dazu sei zuerst $d_{\mathrm{NN}}(\mathbf{X}, \hat{\mathbf{X}})$ die mit der Anzahl der Delay-Vektoren normierte Quadratwurzel der Summe der quadrierten euklidischen Abstände der $\boldsymbol{x}_t$ zu ihrem jeweils nächsten Nachbarn aus der Menge der $\hat{\boldsymbol{x}}_t$, und $d_{\mathrm{NN}}(\hat{\mathbf{X}}, \mathbf{X})$ sei entsprechend die mit N normierte Quadratwurzel der Summe der quadrierten euklidischen Abstände der $\hat{\boldsymbol{x}}_t$ zu ihrem jeweils nächsten Nachbarn aus der Menge der $\boldsymbol{x}_t$:

$$\begin{aligned} d_{\mathrm{NN}}(\mathbf{X}, \hat{\mathbf{X}}) &= \frac{1}{N} \sqrt{\sum_{t=1}^{N} \|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{\mathrm{nn}_t}\|_2^2} \\ d_{\mathrm{NN}}(\hat{\mathbf{X}}, \mathbf{X}) &= \frac{1}{N} \sqrt{\sum_{t=1}^{N} \|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_{\mathrm{nn}_t}\|_2^2} \,. \end{aligned} \tag{4.6}$$

Dabei ist $\hat{\boldsymbol{x}}_{\mathrm{nn}_t}$ der aus der Menge der $\hat{\boldsymbol{x}}_t$ bestimmte nächste Nachbar von $\boldsymbol{x}_t$ und entsprechend $\boldsymbol{x}_{\mathrm{nn}_t}$ der aus der Menge der $\boldsymbol{x}_t$ bestimmte nächste Nachbar von $\hat{\boldsymbol{x}}_t$. Als Diskrepanz zwischen beiden Attraktoren wird nun die Summe der beiden Werte in (4.6) gesetzt:

$$d_{\mathrm{NN}} = d_{\mathrm{NN}}(\mathbf{X}, \hat{\mathbf{X}}) + d_{\mathrm{NN}}(\hat{\mathbf{X}}, \mathbf{X}) \,. \tag{4.7}$$

4.3.3 Vergleich der beiden Ansätze

In diesem Abschnitt sollen die beiden zuvor dargestellten Ansätze zur Quantifizierung der Diskrepanz zweier Attraktoren miteinander verglichen werden. Dies geschieht anhand zweier Modelle für die Abhängigkeit vom Parameter G beim Chua-Oszillator, von denen eins die Systemdynamik an allen Parameterstützstellen, für die Trainingsdaten verwendet wurden, richtig approximiert, während das andere die Dynamik an einigen Stützstellen inkorrekt beschreibt. Die Ergebnisse sind in Abb. 4.5 zu sehen. Die oberste Zeile zeigt zum Vergleich jeweils das Original-Bifurkationsdiagramm und die zweite Zeile die rekonstruierten Bifurkationsdiagramme des schlechten (links) bzw. des guten Modells (rechts). Darunter sind die zugehörigen, an den Parameterstützstellen berechneten Diskrepanzen zwischen den originalen ($\boldsymbol{x}$) und den Modell-Attraktoren ($\hat{\boldsymbol{x}}$) dargestellt. Die dritte Zeile zeigt die Diskrepanz-Werte, die mit der KLD-Methode und mit einer Bandbreite von 0,5 für die Schätzung der Wahrscheinlichkeitsdichten ermittelt wurden. $d(\boldsymbol{x}, \hat{\boldsymbol{x}})$ und $d(\hat{\boldsymbol{x}}, \boldsymbol{x})$ sind die Werte der Kullback-Leibler-Divergenz und $d_{\mathrm{KLD}} = d(\boldsymbol{x}, \hat{\boldsymbol{x}}) + d(\hat{\boldsymbol{x}}, \boldsymbol{x})$ ist die in (4.5) als Diskrepanz der beiden Attraktoren

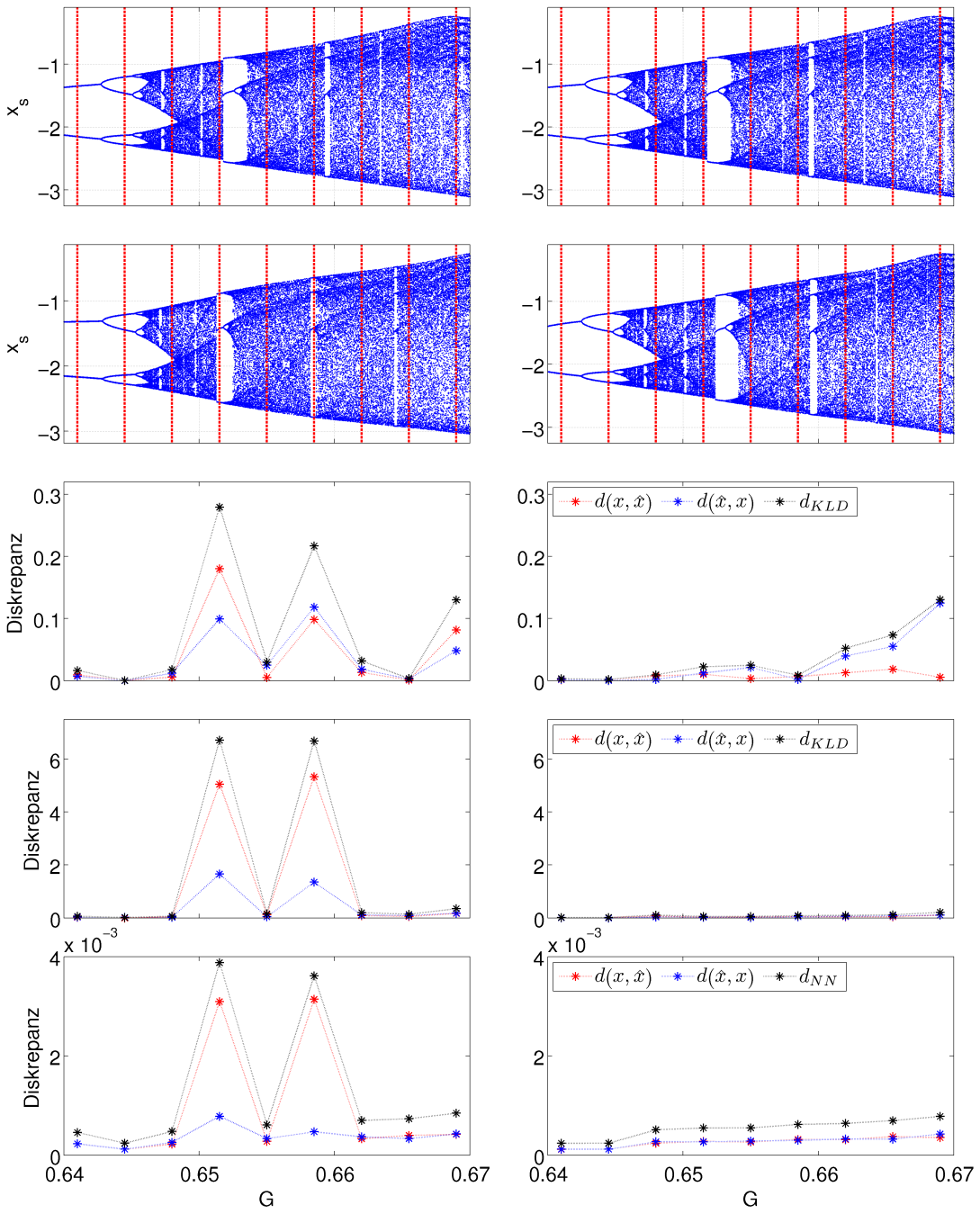


Abbildung 4.5: Zum Vergleich der Diskrepanzen: Ganz oben die Original-Bifurkationsdiagramme. 2. Zeile: Rekonstruierte Bifurkationsdiagramme eines schlechten (links) und eines guten (rechts) Modells. 3. Zeile: Zugehörige KLD-Diskrepanzen für die Bandbreite $0{,}5$. 4. Zeile: KLD-Diskrepanzen für die Bandbreite $0{,}09$. Ganz unten: NN-Diskrepanzen. $\boldsymbol{x}$ sind jeweils die Originaldaten und $\hat{\boldsymbol{x}}$ enstammen der freien Modelliteration.

definierte Summe beider Werte an den jeweiligen Parameterstützstellen. Die vierte Zeile zeigt die entsprechenden Werte, die sich ergeben, wenn für die Bandbreite der zur Schätzung der Wahrscheinlichkeitsdichten verwendeten Gauß-Funktionen ein Wert von 0,09 verwendet wird. Schließlich zeigt die letzte Zeile die mit der NN-Methode ermittelten Diskrepanzwerte zwischen beiden Attraktoren, wobei $d_{\mathrm{NN}} = d(\boldsymbol{x}, \hat{\boldsymbol{x}}) + d(\hat{\boldsymbol{x}}, \boldsymbol{x})$ die in (4.7) als Attraktor-Diskrepanz definierte Summe der beiden Einzel-Diskrepanzen ist. Zur besseren Vergleichbarkeit sind die Y-Achsen für beide Modelle jeweils gleich skaliert. Beim rekonstruierten Bifurkationsdiagramm des schlechten Modells treten die beiden periodischen Fenster, die beim Original zwischen $G = 0{,}65$ und $G = 0{,}655$ bzw. bei $G \approx 0{,}66$ liegen, zu früh auf, so dass dieses Modell für die vierte Parameterstützstelle bei $G = 0{,}6515$ einen Orbit mit Periode 3 liefert und für die sechste Parameterstützstelle bei $G = 0{,}6585$ einen Periode-4-Orbit. Für beide Parameterwerte ist die wahre Dynamik jedoch chaotisch. Entsprechend liefern sowohl die KLD-Methode für beide Werte der Bandbreite als auch die NN-Methode an diesen Stützstellen Werte für die Diskrepanz, die groß sind im Vergleich zu den Werten an den anderen Stützstellen, für die die Modelldynamik gut mit der wahren Dynamik übereinstimmt. Die Ergebnisse der KLD-Methode hängen jedoch von der gewählten Bandbreite der Gauß-Kerne und somit von der Güte der Wahrscheinlichkeitsdichteschätzung ab. Darin liegt auch eine Schwäche dieser Methode: Wie in der Definition der Kullback-Leibler-Divergenz (4.4) ersichtlich, müssen beide Dichteverteilungen für alle Datenpunkte echt größer Null sein, da andernfalls die KLD nicht definiert ist. Mit einer hinreichend kleinen Bandbreite beim Kerndichte-Schätzer ist zwar eine genaue Schätzung der Verteilung möglich, allerdings kann es dann passieren, dass für zwei komplett verschiedene, disjunkte Attraktoren an Orten, die zum einen Attraktor gehören, die Wahrscheinlichkeitsdichteverteilung des anderen Attraktors praktisch Null ist. Mit einer zu großen Bandbreite ist hingegen keine hinreichend genaue Schätzung der Verteilungen möglich. Dies zeigt sich in Abb. 4.5 bei den KLD-Werten für die (zu große) Bandbreite 0,5 (dritte Zeile), die an der letzten Parameterstützstelle bei $G = 0{,}669$ fast ebenso große Werte annehmen wie bei $G = 0{,}6585$, obwohl im ersten Fall das Modell die chaotische Dynamik korrekt beschreibt, beide Attraktoren also (näherungsweise) übereinstimmen, während im zweiten Fall das Modell fälschlicherweise eine periodische Dynamik liefert, die sich im rekonstruierten Zustandsraum komplett von der wahren, chaotischen Dynamik unterscheidet. Die NN-Methode zeigt insgesamt eine recht gute Übereinstimmung mit der KLD-Methode, wenn bei letzterer eine geeignete Bandbreite wie in diesem Fall 0,09 gewählt wird. Allerdings ist bei der NN-Methode eine

leicht ansteigende Tendenz der Werte vom kleinsten zum größten Parameterwert festzustellen, die in den unterschiedlichen Attraktorgeometrien begründet liegt: Die Werte links im Diagramm gehören zu einem Periode-1-Attraktor, während die Werte rechts im Diagramm zu einem chaotischen Attraktor gehören, der ein wesentlich größeres Gebiet im rekonstruierten Zustandsraum überdeckt, so dass dort bei gleicher Anzahl der Datenpunkte die Abstände zu den nächsten Nachbarn im Mittel größer sind. An den großen Diskrepanzwerten beim schlechten Modell lässt sich auch sehr schön die Asymmetrie der Einzeldiskrepanzen vom Modell zum Original und ungekehrt erkennen: An den beiden Stützstellen, an denen das Modell fälschlicherweise eine periodische Dynamik liefert, ist die Diskrepanz vom Modell- zum Originalattraktor deutlich größer als die des Originalattraktors zum Modellattraktor. Die Situation ist vergleichbar mit dem Beispiel in Abb. 4.4: Zu jedem Punkt des periodischen Modellattraktors existieren in der Nähe Punkte des Originalattraktors, so dass die Diskrepanz in dieser Richtung klein ist. Im Gegensatz dazu tragen die weit vom periodischen Attraktor entfernten Punkte des chaotischen Attraktors zu einer großen Diskrepanz in der umgekehrten Richtung bei. Ein Maß für die Unterschiedlichkeit zweier Attralktoren sollte allerdings bei geometrisch verschiedenen Attraktoren wie einem chaotischen und einem periodischen in jedem Fall einen großen Wert liefern, unabhängig davon, ob der wahre Attraktor nun der chaotische oder der periodische ist, was die Symmetrisierung durch die Summation der Diskrepanzen in beiden Richtungen motiviert.

Ein weiterer Vorteil der NN-Methode neben der Unabhängigkeit von Parametern wie der Bandbreite bei der KLD-Methode ist der deutlich geringere Rechenaufwand der Bestimmung des nächsten Nachbarn jedes Datenpunktes gegenüber der Kerndichte-Schätzung. Zur Berechnung der nächsten Nachbarn existieren effiziente Verfahren wie der in dieser Arbeit verwendete ATRIA-Algorithmus[2] [80,81], bei dem die Laufzeit im Wesentlichen von der fraktalen Dimension des Attraktors abhängt und nur unwesentlich von der meist (deutlich) größeren Dimension des rekonstruierten Zustandsraumes.

[2] **A**dvanced **TR**iangle **I**nequality **A**lgorithm

4.4 Optimierte Rekonstruktion von Bifurkationsdiagrammen

Die zuvor eingeführten Maße der Attraktor-Diskrepanz können verwendet werden, um aus einem Vorrat an Modellen, die mit den in Kapitel 3 vorgestellten Methoden konstruiert wurden, ein Modellensemble zu konstruieren, indem für jede Parameterstützstelle das Modell mit der kleinsten Attraktor-Diskrepanz ermittelt wird. Diese Modelle werden anschließend zu einem Ensemble verbunden, das als Ausgabe eine gewichtete Summe der Ausgaben der Einzelmodelle liefert. Wenn im Modellvorrat für jede Stützstelle ein Modell enthalten ist, das die Dynamik an dieser Stützstelle richtig approximiert und die Gewichtung der einzelnen Modelle im Ensemble parameterabhängig gewählt wird und zwar so, dass an jeder Stützstelle nur das jeweils beste Modell ein Gewicht von Eins erhält und alle anderen Modelle an dieser Stelle im Parameterraum nicht zum Ensemble beitragen, so ist damit sichergestellt, dass das Ensemble zumindest an allen Stützstellen die Dynamik korrekt wiedergibt. Über andere Parameterwerte lässt sich allerdings keine sichere Aussage treffen. Abb. 4.6 zeigt dies an einem Beispiel für den Chua-Oszillator. Aus einem Vorrat von insgesamt 126 Modellen, die durch Termselektion und anschließende Minimierung des Einschritt- und des Mehrschritt-Vorhersagefehlers erzeugt wurden, wurde mit Hilfe der NN-Methode für jede Parameterstützstelle das jeweils beste Modell als dasjenige mit der kleinsten Attraktordiskrepanz ausgewählt und aus diesen ein Ensemble gebildet. Dieses bestand nur aus 7 statt 9 verschiedenen Modellen, da 2 der Einzelmodelle an jeweils 2 Parameterstützstellen als die besten identifiziert wurden. Über den gesamten modellierten Parameterbereich hinweg betrachtet liefert das Ensemble bessere Ergebnisse als jedes Einzelmodell. Die Dynamik wird an allen Stützstellen durch das Ensemble korrekt wiedergegeben. Allerdings liegt auch das Ensemble an einigen Werten direkt neben einer Stützstelle schon wieder falsch.

Problematisch sowohl bei der KLD- als auch bei der NN-Methode ist die Interpretation der Absolutwerte der ermittelten Diskrepanzen. Diese hängen von der Anzahl der Datenpunkte, der Geometrie der Attraktoren und bei der KLD-Methode auch von der Bandbreite der Kernfunktionen ab und erlauben daher immer nur einen relativen Vergleich zwischen Modellen in dem Sinne, dass man aus dem Wert der Diskrepanz entscheiden kann, dass Modell 1 an einem Ort im Parameterraum die Dynamik besser approximiert als Modell 2. Der Diskrepanzwert an sich erlaubt jedoch nicht ohne Weiteres eine Aussage über die absolute

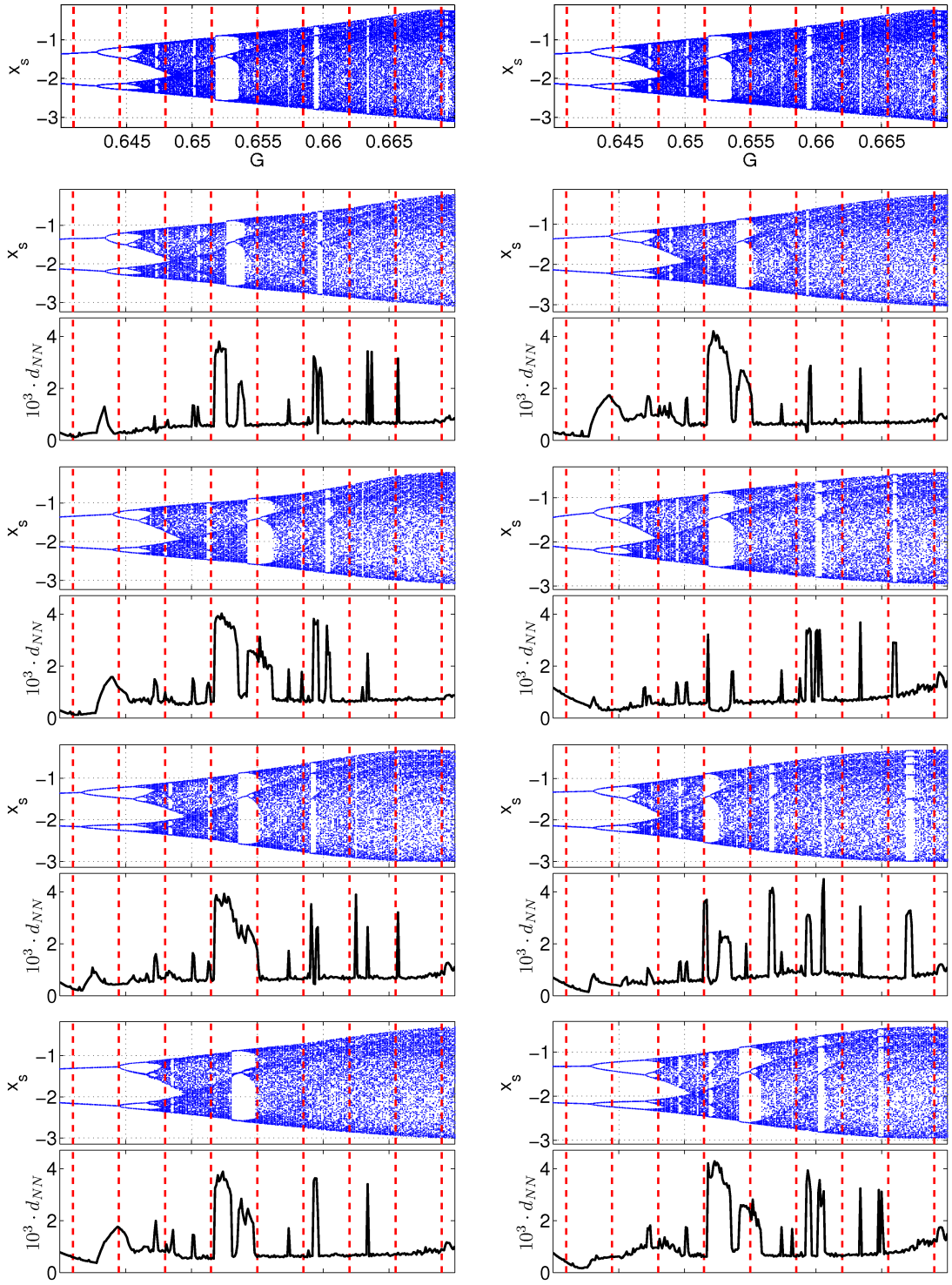


Abbildung 4.6: Die erste Zeile zeigt zum Vergleich die Original-Bifurkationsdiagramme. In den übrigen Zeilen sind rekonstruierte Bifurkationsdiagramme sowie darunter jeweils die NN-Diskrepanzen dargestellt. Die zweite Zeile zeigt links das Ergebnis des Ensembles. Die übrigen Diagramme und Diskrepanzkurven entsprechen den Einzelmodellen des Ensembles.

Übereinstimmung mit dem Original-Attraktor: Ergibt sich für Modell 1 ein niedrigerer Diskrepanzwert als für Modell 2, so bedeutet das nur, dass Modell 1 die Dynamik besser beschreibt, aber nicht, dass Modell 1 die Dynamik auch *richtig* beschreibt bzw. dass Modell 2 eine inkorrekte Approximation der Dynamik liefert.

Im Folgenden soll darum ein NN-basierter Ansatz vorgestellt werden, der eine solche Aussage über die Richtigkeit der vorhergesagten Langzeitdynamik eines Modells erlaubt. Dazu seien jeweils N Delay-Vektoren $\boldsymbol{x}_t$ der Original-Zeitreihen bzw. $\hat{\boldsymbol{x}}_t$ der freien Modelliteration gegeben. Die Idee ist nun, die Mengen X bzw. $\hat{X}$ jeweils in 2 Hälften X_1, X_2 bzw. $\hat{X}_1$, $\hat{X}_2$ einzuteilen und X_1 und $\hat{X}_1$ zu einer gemischten Menge G zu vereinen. Anschließend wird zu jedem der Delayvektoren in den verbleibenden Mengen X_2 bzw. $\hat{X}_2$ der nächste Nachbar aus der gemischten Menge G bestimmt. Nun werden aber nicht die Abstände dieser nächsten Nachbarn betrachtet, sondern nur die relative Häufigkeit dafür, dass der ermittelte nächste Nachbar dem jeweils anderen Attraktor entstammt. Bei der Suche der nächsten Nachbarn für die Zustände in X_2 aus der Menge G wird also die relative Häufigkeit bestimmt, mit der dieser Nachbar aus dem ursprünglichen $\hat{X}_1$ stammt, während bei der Suche der Nachbarn von $\hat{X}_2$ die relative Häufigkeit bestimmt wird, mit der der nächste Nachbar aus X_1 stammt. Sind beide Attraktoren gleich, sollten die relativen Häufigkeiten bei etwa 50% liegen. Unterscheiden sich die Attraktoren jedoch deutlich, stammen die nächsten Nachbarn (fast) ausschließlich vom gleichen Attraktor, die relativen Häufigkeiten liegen also annähernd bei Null. Abb. 4.7 zeigt die Anwendung dieser Methode auf das zuvor beschriebene Ensemble von Modellen für die Parameterabhängigkeit beim Chua-System. Es wurden die NN-Diskrepanzen sowie die relativen Häufigkeiten, dass die nächsten Nachbarn jeweils aus dem anderen Attraktor stammen, für alle Parameterwerte berechnet, für die auch Punkte im Bifurkationsdiagramm aufgetragen sind. Die rote Kurve bezeichnet dabei die relative Häufigkeit, einen nächsten Nachbarn von Punkten des Originalattraktors aus den Zuständen des Ensemble-Attraktors zu finden. Die blaue Kurve stellt entsprechend den umgekehrten Fall dar. Die relativen Häufigkeiten korrespondieren recht gut mit den NN-Diskrepanzen: Dort, wo große Diskrepanzen auftreten, sind die relativen Häufigkeiten sehr klein, während sie an Parameterwerten, für die die Dynamiken gut übereinstimmen, oft zwischen 0,4 und 0,5 liegen. Allerdings reagieren diese relativen Häufigkeiten ziemlich empfindlich schon auf kleine Unterschiede in den Attraktorgeometrien, was sich z.B. bei der kleinsten Parameterstützstelle äußert, für die zwar auch das Ensemble einen Orbit der Periode 1 erzeugt, der aber im

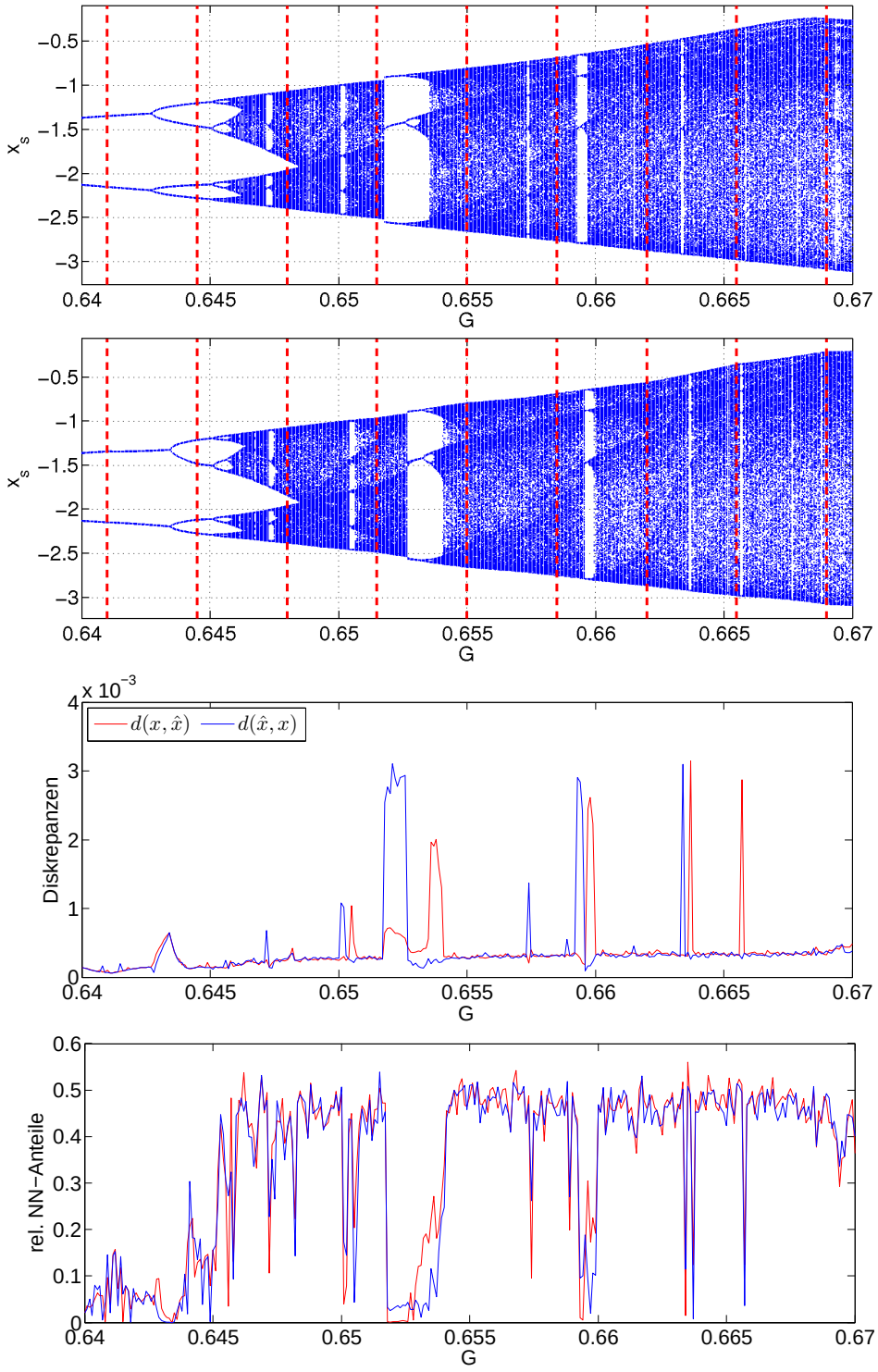


Abbildung 4.7: Zum Vergleich: Oben das originale und darunter das aus den Zeitreihen rekonstruierte Bifurkationsdiagramm des Chua-Systems unter Verwendung eines Ensembles aus 7 Modellen. 3. Zeile: NN-Diskrepanzen. 4. Zeile: relative Anteile der nächsten Nachbarn, gemittelt über 50 zufällige Aufteilungen.

rekonstruierten Zustandsraum offenbar nicht völlig deckungsgleich mit dem Originalattrator ist.

Als zweites Beispiel dient die Modellierung der Parameterabhängigkeit beim Rössler-System, das beschrieben wird durch das System von Differentialgleichungen

$$\begin{aligned} \dot{x}_1 &= -x_2 - x_3 \\ \dot{x}_2 &= x_1 + a x_2 \\ \dot{x}_3 &= b + x_3(x_1 - c)\,, \end{aligned} \tag{4.8}$$

wobei der Parameter a als Bifurkationsparameter verwendet und $b = 2,\ c = 4$ gesetzt wurden. Es wurden 8 Zeitreihen $\{s_t\}$ der Länge 3000 Samples für die Werte $a = \{0{,}2775,\ 0{,}3125,\ 0{,}3475,\ 0{,}3825,\ 0{,}4175,\ 0{,}4525,\ 0{,}4875,\ 0{,}5225\}$ erzeugt und zusätzlich mit einem mittelwertfreien, normalverteilten Rauschsignal $\{\epsilon_t\}$ überlagert, wobei das Verhältnis der Standardabweichungen $\sigma_\epsilon/\sigma_s = 0{,}10$ betrug. Mittels Termselektion wurden zuerst 7 RBF-Modelle mit konstantem und linearem Term konstruiert. Dabei diente die Cross-Validation zur Bestimmung der optimalen Termanzahl, die bei den Modellen zwischen 35 und 44 Termen lag. Anschließend erfolgte eine Optimierung der Zentren und Breiten der RBF-Terme durch Minimierung des Einschritt-Vorhersagefehlers mit verschiedenen Optimierungsstrategien (gleichzeitige Optimierung aller Zentren- und Breitenkoordinaten bzw. zyklische Optimierung aller Zentren- und Breitenwerte immer nur eines Terms). Schließlich wurde noch eine Minimierung des Mehrschritt-Vorhersagefehlers bzgl. aller Modellparameter mit verschiedenen Schrittweiten durchgeführt (20, 50 und 80 Vorhersageschritte). Die Modelle auch aller Zwischenschritte dieser Optimierungskaskade wurden mit der NN-Methode auf die Übereinstimmung der Langzeitdynamik mit der der Originalzeitreihen verglichen, das jeweils beste Modell bestimmt und aus den so erhaltenen 7 Modellen (ein Modell erwies sich wieder an zwei Stützstellen als das Beste) ein Ensemble konstruiert. Das mit diesem Ensemble rekonstruierte Bifurkationsdiagramm ist in Abb. 4.9 dargestellt. Es zeigt eine sehr gute Übereinstimmung an allen Parameterstützstellen. Allerdings tritt beim Ensemble-Diagramm unmittelbar neben der größten Parameterstützstelle ein kleines periodisches Fenster auf, das dort beim Original nicht zu sehen ist. Weiterhin ist die beim Original auftretende Einkerbung im oberen Bereich des Diagramms bei $a \gtrapprox 0{,}43$ sowie der „Überhang“ ganz oben rechts im Original-Diagramm nicht im rekonstruierten enthalten und das große periodische Fenster fällt zu groß aus. Abb. 4.9 zeigt wieder die beiden

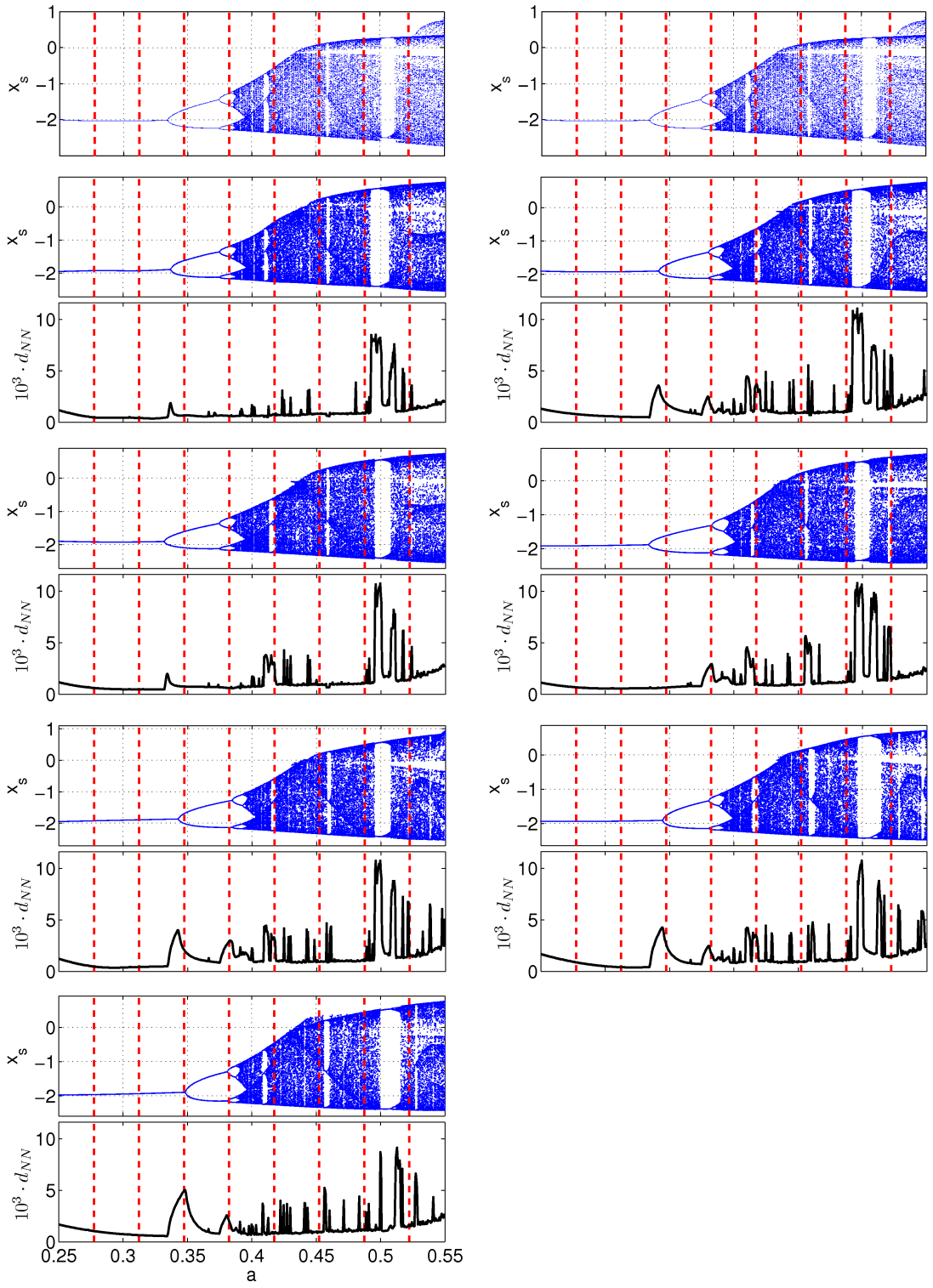


Abbildung 4.8: Die erste Zeile zeigt zum Vergleich die Original-Bifurkationsdiagramme des Rössler-Systems. In den übrigen Zeilen sind rekonstruierte Bifurkationsdiagramme sowie darunter jeweils die NN-Diskrepanzen dargestellt. Die zweite Zeile zeigt links das Ergebnis des Ensembles. Die übrigen Diagramme und Diskrepanzkurven entsprechen den Einzelmodellen des Ensembles.

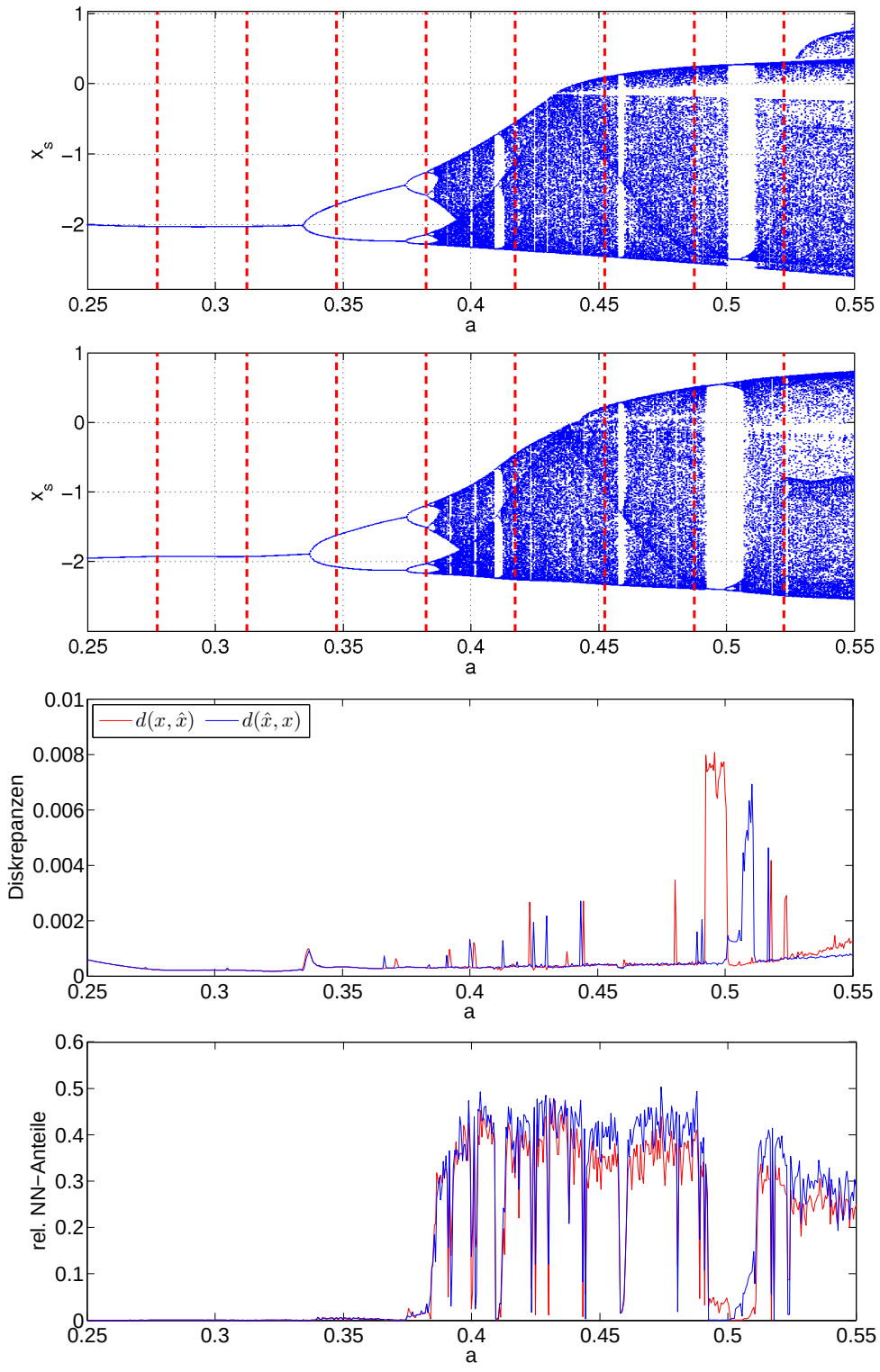


Abbildung 4.9: Zum Vergleich: Oben das originale und darunter das aus den verrauschten Zeitreihen rekonstruierte Bifurkationsdiagramm des Rössler-Systems unter Verwendung eines Ensembles aus 6 Modellen. 3. Zeile: NN-Diskrepanzen. 4. Zeile: relative Anteile der nächsten Nachbarn aus dem jeweils gleichen Attraktor, gemittelt über 50 zufällige Aufteilungen der Daten.

Bifurkationsdiagramme sowie die NN-Diskrepanzen und die relativen Häufigkeiten der nächsten Nachbarn. Hier tritt die oben angesprochene Empfindlichkeit der relativen Häufigkeiten auch bei dicht beieinanderliegenden, sich aber nicht überdeckenden Attraktoren besonders deutlich zutage: Die Lage der Orbits mit Periode 1 und 2 wird zwar durch das Ensemble richtig vorhergesagt, jedoch unterscheiden sich die Geometrien leicht, wodurch die relativen Häufigkeiten nahezu Null sind.

Kapitel 5
Zusammenfassung und Ausblick

Die vorliegende Arbeit hat sich mit der Modellierung dynamischer Prozesse mit Hilfe von radialen Basisfunktionen (RBF) befasst. Die der Modellbildung zugrundeliegende statistische Regressionstheorie wurde ausführlich dargelegt. Dabei wurden verschiedene Methoden der Termselektion diskutiert, die der Erzeugung kompakter Modelle mit guten Generalisierungseigenschaften dienen. Im Rahmen dieser Arbeit konnten zwei effiziente Algorithmen zur Termselektion, die *Fast Orthogonal Search*-Methode von Korenberg und der *Backward Greedy Algorithm* von Reeves, um die Ridge-Regression erweitert werden, was eine bessere Steuerung der Modellkomplexität bei gleichzeitiger hoher Ausführungsgeschwindigkeit erlaubte. Da radiale Basisfunktionen im Gegensatz zu bspw. den Monomen polynomialer Modelle noch zusätzliche Parameter in Form ihrer Zentren und Breiten besitzen, die nichtlinear in die Modellausgabe eingehen, wurde untersucht, welcher Einfluss der Termselektion zukommt, wenn das so erzeugte Modell anschließend hinsichtlich dieser nichtlinearen Parameter nachoptimiert wird. Dabei konnte gezeigt werden, dass die Wahl der Termselektionsmethode zur Konstruktion eines initialen Modells nur einen sehr begrenzten Einfluss auf das finale, der nichtlinearen Optimierung entstammende Modell hat. Es erwies sich daher als sinnvoller, mehr Rechenzeit in die nichtlineare Optimierung zu investieren als in eine ausgeklügelte Termselektion. Die im Rahmen dieser Arbeit um die Ridge-Regression erweiterte Forward Orthogonal Search-Methode stellte sich als ein guter Kompromiss aus erzielter Genauigkeit des Modells und der dazu investierten Rechenzeit heraus.

Ein besonderes Augenmerk wurde auf die korrekte Erfassung der Langzeitdynamik bei der Modellierung dynamischer Systeme gelegt. Das Ziel war hier die Modellierung der Dynamik chaotischer Systeme und der Abhängigkeit dieser Dynamik von externen Parametern zur Rekonstruktion ihrer Bifurkationsstruktur

aus Messungen in Form von Zeitreihen, die für einige wenige Werte der Parameter vorgenommen wurden. Es zeigte sich, dass die üblichen Methoden der Minimierung des Einschritt-Vorhersagefehlers nur einen ungenügenden Anhaltspunkt für die korrekte Beschreibung der Langzeitdynamik bei freier Iteration des Modells liefern. Dies äußerte sich z.B. darin, dass das Modell bei freier Iteration häufig für bestimmte Werte der Parameter eine periodische Dynamik lieferte, obwohl das zu modellierende System für diese Parameterwerte eine chaotische Dynamik aufwies und umgekehrt. Dies trat oft sogar für Werte der Parameter auf, für die Trainingsdaten für das Modell in Form von Zeitreihen vorlagen. Von Langer wurde diesbezüglich der Weg beschritten, die Dynamik des frei laufenden Modells durch eine Minimierung des Mehrschritt-Vorhersagefehlers längerfristig an die Dynamik des zu modellierenden Systems zu binden. Während Langer die Optimierung dabei nur bezüglich der RBF-Koeffizienten durchführte, wurden in dieser Arbeit auch die Zentren und Breiten der RBF-Terme in die Optimierung mit einbezogen. Dies führte zwar zu einer Verbesserung hinsichtlich der Problematik der korrekten Wiedergabe der Langzeitdynamik durch das frei laufende Modell, stieß aber u. a. wegen der sensitiven Abhängigkeit chaotischer dynamischer Systeme von ihren Anfangsbedingungen und der komplizierten, nichtlinear von den Parametern abhängigen Fehlerlandschaft mit vielen lokalen Minima schnell an prinzipielle Grenzen. Im Rahmen dieser Arbeit wurden daher Möglichkeiten untersucht, direkt den Attraktor des frei laufenden Systems mit dem des zu modellierenden Systems im rekonstruierten Zustandsraum zu vergleichen und so ein Maß für die *Diskrepanz* beider Attraktoren zu erhalten.

Als eine etablierte Methode zur Bestimmung der Verschiedenheit zweier Wahrscheinlichkeitsverteilungen wurde die Eignung der *Kullback-Leibler-Divergenz* (KLD) als Maß für die Verschiedenheit zweier Attraktoren untersucht. Diesem Ansatz lag die Idee zugrunde, die rekonstruierten Zustandsvektoren der Modellausgabe und der Trainingsdaten als Realisierung jeweils eines stochastischen Prozesses aufzufassen, eine Schätzung der den Realisierungen zugrundeliegenden Wahrscheinlichkeitsdichten vorzunehmen und mit der KLD die Verschiedenheit dieser Verteilungen und damit der Attraktoren zu quantifizieren. Die grundsätzlichen Probleme dieses Vorgehens, die sich in der Wahl geeigneter Parameter des Dichte-Schätzers (hier der Bandbreiten der zur Schätzung verwendeten Gauß-Kerne) äußerten, wurden aufgezeigt. Wurden diese zu klein gewählt, ergab sich oft ein unendlich großer Wert der KLD, obwohl die verglichenen Attraktoren nicht völlig verschieden waren (z.B. ein periodischer Attraktor, der in einen chaotischen Attraktor eingebettet war). Dieses Problem ergab sich oft auch, wenn die Band-

breiten automatisch z.B. mittels Cross-Validation bestimmt wurden. Hingegen war die Dichteschätzung bei einer zu großen Wahl der Bandbreite zu ungenau und die KLD-Werte waren unplausibel.

Als Alternative zu diesem etablierten Verfahren wurde daher eine neue Methode zur Quantifizierung der Diskrepanz zwischen Attraktoren entwickelt, die auf den Abständen der Punkte eines Attraktors zu ihren nächsten Nachbarn aus der Punktmenge des anderen Attraktors basiert und die keine Parameter enthält, von deren Werten sie kritisch abhängt. Es konnte eine gute Übereinstimmung der mit dieser NN-Methode (NN steht für *nächste Nachbarn*) ermittelten Diskrepanzwerte mit den durch die KLD ermittelten festgestellt werden, wenn für letztere geeignete Bandbreiten gewählt wurden. Aufgrund der Verfügbarkeit effizienter Algorithmen zur Bestimmung nächster Nachbarn erfordert diese Methode darüber hinaus einen wesentlich geringeren Rechenaufwand als bspw. die Berechnung der Kullback-Leibler-Divergenz. Mit Hilfe dieser Methode war es möglich, bei der Modellierung von Parameterabhängigkeiten in Bezug auf die Problematik inkorrekt erfasster Dynamik an Orten im Parameterraum, für die Zeitreihen als Trainingsdaten vorlagen, aus einem Vorrat konstruierter Modelle automatisch jeweils das Modell für jede Parameterstützstelle zu bestimmen, das die Dynamik für diesen Parameterwert am besten beschreibt. Die Idee war dann, diese Informationen zu verwenden, um aus den jeweils besten Modellen ein Ensemble von Modellen mit parameterabhängiger Gewichtung zu konstruieren. Enthielt nun der Vorrat an Modellen für jede Stützstelle mindestens ein Modell, das die Dynamik bei dieser Stützstelle korrekt erfasste, so konnte bei geeigneter Gewichtung sicher gewährleistet werden, dass das Ensemble die Parameterabhängigkeit zumindest an allen Stützstellen richtig wiedergab.

Sowohl die per KLD ermittelten Diskrepanzwerte als auch diejenigen, die mit der NN-Methode bestimmt wurden, ließen sich ausschließlich *relativ* untereinander vergleichen, um z.B. das Modell zu bestimmen, dessen Dynamik an einer Parameterstützstelle am besten mit der wahren Dynamik übereinstimmte. Die *Absolutwerte*, die von den Attraktorgeometrien und im Fall der KLD auch von der Bandbreite des Dichteschätzers abhingen, erlaubten hingegen keine Aussage darüber, wie exakt diese Übereinstimmung der Langzeitdynamik war, ob also das als das beste identifizierte Modell nicht einfach nur das am wenigsten schlechte war.

Ein Schritt in Richtung einer quantitativen Aussage über das Maß der Übereinstimmung der Attraktoren wurde durch eine Abwandlung der NN-Methode

unternommen. Diese bestand darin, in einer Vereinigung von rekonstruierten Zustandsraumvektoren der Trainingsdaten und der Modellausgaben die nächsten Nachbarn zu den jeweils nicht in dieser Vereinigung enthaltenen Delay-Vektoren der Trainingsdaten bzw. Modellausgaben zu suchen und die relativen Anteile bzgl. der Anzahlen der Nachbarn zu bestimmen, die vom jeweils anderen System (Original oder Modell) stammten. Anstelle der eigentlichen Nachbar-Abstände wurde hier also nur untersucht, wie hoch der Anteil der jeweils nächsten Nachbarn ist, die vom anderen Attraktor stammen. Ein Wert von etwa 50% deutete auf eine sehr gute Übereinstimmung der Attraktoren hin, während ein Wert von nahezu 0% auf eine deutliche Verschiedenheit der Attraktoren hinwies.

Eine direkte Beschränkung der vorgestellten Methodik zur Quantifizierung der Verschiedenheit von Attraktoren hinsichtlich der Erzeugung von Modellen oder Ensembles von Modellen, die die Parameterabhängigkeit der Dynamik im gesamten betrachteten Parameterraum korrekt wiedergeben, liegt allerdings auf der Hand: Die Methoden eignen sich lediglich dazu, fertig konstruierte Modelle *im Nachhinein* auf die Übereinstimmung ihrer vorhergesagten Langzeitdynamik zu bewerten. Diese Methoden bieten hingegen keinen Anhaltspunkt zur Modellkonstruktion selbst, der eine solche Übereinstimmung der Dynamiken gewährleisten könnte. Das eigentliche Problem der Konstruktion von Modellen mit korrekter Nachbildung der asymptotischen Dynamik bleibt also weiter ungelöst. Eine Idee könnte z.B. sein, die Attraktordiskrepanz als Kostenfunktion zu verwenden und diese mit Optimierungsverfahren zu minimieren. Eigene erste Versuche in dieser Richtung brachten jedoch keine zufriedenstellenden Ergebnisse, zumal die wiederholte Auswertung einer solchen Kostenfunktion durch einen Optimierungsalgorithmus sehr rechenintensiv ist und dieser außerdem ohne Gradienteninformationen auskommen muss. Hier bieten sich vielfältige Möglichkeiten für zukünftige Forschungen.

Literaturverzeichnis

[1] J. Dittmar: *Nichtlineare Dimensionsreduktionsmethoden in der Datenanalyse und Signalverarbeitung.* Diplomarbeit, Georg-August-Universität Göttingen, 2002.

[2] J. A. Lee und M. Verleysen: *Nonlinear Dimensionality Reduction.* Springer, 2007.

[3] W.-H. Steeb: *Chaos und Quantenchaos in Dynamischen Systemen.* BI Wissenschaftsverlag, Mannheim, 1994.

[4] J. Argyris, G. Faust und M. Haase: *Die Erforschung des Chaos.* Vieweg, 1995.

[5] E. Ott: *Chaos in Dynamical Systems.* Cambridge University Press, 1993.

[6] F. Takens: *Detecting strange attractors in turbulence.* In: *Dynamical Systems and Turbulence*, Band 898 der Reihe *Lecture Notes in Mathematics*, Seiten 366–381. Springer, Berlin, Heidelberg, 1981.

[7] T. Sauer, J. A. Yorke und M. Casdagli: *Embedology.* J. Stat. Phys. **65**(3–4), 579–616, 1991.

[8] H. D. I. Abarbanel: *Analysis of Observed Chaotic Data.* Springer, New York, 1996.

[9] H. D. I. Abarbanel und U. Parlitz: *Nonlinear Analysis of Time Series Data.* In: B. Schelter, M. Winterhalder und J. Timmer (Herausgeber): *Handbook of Time Series Analysis*, Seiten 5–37. Wiley-VCH, Weinheim, 2006.

[10] M. Small: *Applied Nonlinear Time Series Analysis: Applications in Physics, Physiology and Finance*, Band 52 der Reihe *World Scientific Series on Nonlinear Science, Series A.* World Scientific, 2005.

[11] J. D. Farmer und J. J. Sidorowich: *Predicting chaotic time series.* Phys. Rev. Lett. **59**(8), 845–848, 1987.

[12] M. Casdagli: *Nonlinear prediction of chaotic time series.* Physica D **35**(3), 335–356, 1989.

[13] S. A. Billings, H. B. Jamaluddin und S. Chen: *Properties of neural networks with applications to modelling non-linear dynamical systems.* Int. J. Control **55**(1), 193–224, 1992.

[14] S. Chen und S. A. Billings: *Representations of non-linear systems: the NARMAX model.* Int. J. Control **49**(3), 1013–1032, 1989.

[15] L. A. Aguirre und S. A. Billings: *Retrieving dynamical invariants from chaotic data using narmax models.* Int. J. of Bifurcation and Chaos **5**(2), 449–474, 1995.

[16] S. Chen, S. A. Billings und W. Luo: *Orthogonal least squares methods and their application to non-linear system identification.* Int. J. Control **50**(5), 1873–1896, 1989.

[17] S. A. Billings, S. Chen und M. J. Korenberg: *Identification of mimo non-linear systems using a forward-regression orthogonal estimator.* Int. J. Control **49**(6), 2157–2189, 1989.

[18] S. Chen und S. A. Billings: *Recursive prediction error parameter estimator for non-linear models.* Int. J. Control **49**(2), 569–594, 1989.

[19] M. J. Korenberg, S. A. Billings, Y. P. Liu und P. J. McIlroy: *Orthogonal parameter estimation algorithm for non-linear stochastic systems.* Int. J. Control **48**(1), 193–210, 1988.

[20] L. A. Aguirre: *Recovering map static nonlinearities from chaotic data using dynamical models.* Physica D **100**(1), 41–57, 1997.

[21] E. M. A. M. Mendes und S. A. Billings: *On overparametrization of nonlinear discrete systems.* Int. J. of Bifurcation and Chaos **8**(3), 535–556, 1998.

[22] L. A. Aguirre und S. A. Billings: *Dynamical effects of overparametrization in nonlinear models.* Physica D **80**(1), 26–40, 1995.

[23] K. H. Chon, J. K. Kanters, R. J. Cohen und N.-H. Holstein-Rathlou: *Detection of chaotic determinism in time series from randomly forced maps.* Physica D **99**(4), 471–486, 1997.

[24] T. Hastie, R. Tibshirani und J. Friedman: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer, 2. Auflage, 2009.

[25] M. V. Corrêa, L. A. Aguirre und S. A. Billings: *Modeling chaotic dynamics with discrete nonlinear rational models*. Int. J. of Bifurcation and Chaos **10**(5), 1019–1032, 2000.

[26] Q. M. Zhu und S. A. Billings: *Parameter estimation for stochastic nonlinear rational models*. Int. J. Control **57**(2), 309–333, 1993.

[27] S. Chen, S. A. Billings, C. F. N. Cowan und P. M. Grant: *Nonlinear systems identification using radial basis functions*. Int. J. Syst. Sci. **21**(12), 2513–2539, 1990.

[28] G. L. Zheng und S. A. Billings: *Radial basis function network configuration using mutual information and the orthogonal least squares algorithm*. Neural Networks **9**(9), 1619–1637, 1996.

[29] A. I. Mees: *Parsimonious dynamical reconstruction*. Int. J. of Bifurcation and Chaos **3**(3), 669–675, 1993.

[30] Q. M. Zhu und S. A. Billings: *Fast orthogonal identification of nonlinear stochastic models and radial basis function neural networks*. Int. J. Control **64**(5), 871–886, 1996.

[31] W. Luo und S. A. Billings: *Structure selective updating for nonlinear models and radial basis function neural networks*. Int. J. Adapt. Control Signal Process. **12**(4), 325–345, 1998.

[32] T. Nakamura und M. Small: *Modeling nonlinear time series using improved least squares method*. Int. J. of Bifurcation and Chaos **16**(2), 445–464, 2006.

[33] L. Cao, Y. Hong, H. Fang und G. He: *Predicting chaotic time series with wavelet networks*. Physica D **85**(2), 225–238, 1995.

[34] S. A. Billings und H.-L. Wei: *A new class of wavelet networks for nonlinear system identification*. IEEE Transactions on Neural Networks **16**(4), 862–874, 2005.

[35] G. Langer: *Modellierung von Parameterabhängigkeiten nichtlinearer dynamischer Systeme*. Doktorarbeit, Georg-August-Universität Göttingen, 2002.

[36] V. Kecman: *Learning and Soft Computing*, Band 1. The MIT Press, 2001.

[37] M. J. D. Powell: *Radial Basis Functions for Multivariable Interpolation: A Review*. In: *Algorithms for Approximation*, Seiten 143–167, Oxford, 1987. Clarendon Press.

[38] M. J. D. Powell: *Radial Basis Function Approximations to Polynomials.* In: *Proc. 12th Biennial Numerical Analysis Conf. 1987, Dundee*, Seiten 223–241, New York, 1989. Longman Publishing Group.

[39] G. H. Golub und C. F. Van Loan: *Matrix Computations.* The Johns Hopkins University Press, 3. Auflage, 1996.

[40] C. Schaffer: *Overfitting avoidance as bias.* Machine Learning **10**(2), 153–178, 1993.

[41] S. Geman, E. Bienenstock und R. Doursat: *Neural networks and the bias/variance dilemma.* Neural Computation **4**(1), 1–58, 1992.

[42] V. Cherkassky und F. Mulier: *Learning from Data.* Wiley-IEEE Press, 2. Auflage, 2007.

[43] M. Stone: *Cross-validatory choice and assessment of statistical predictions.* Journal of the Royal Statistical Society. Series B **36**(2), 111–147, 1974.

[44] D. M. Allen: *The Relationship between Variable Selection and Data Augmentation and a Method for Prediction.* Technometrics **16**(1), 125–127, 1974.

[45] A. E. Hoerl und R. W. Kennard: *Ridge regression: biased estimation for nonorthogonal problems.* Technometrics **12**(1), 55–67, 1970.

[46] Å. Björck: *Numerical Methods for Least Squares Problems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

[47] R. Tibshirani: *Regression shrinkage and selection via the lasso.* Journal of the Royal Statistical Society. Series B **58**(1), 267–288, 1996.

[48] A. J. Miller: *Subset Selection in Regression.* CRC Press, 2. Auflage, 2002.

[49] S. A. Billings, M. J. Korenberg und S. Chen: *Identification of nonlinear output-affine systems using an orthogonal least-squares algorithm.* Int. J. Syst. Sci. **19**(8), 1559–1568, 1988.

[50] M. J. Korenberg: *Identifying nonlinear difference equation and functional expansion representations: The fast orthogonal algorithm.* Annals of Biomedical Engineering **16**(1), 123–142, 1988.

[51] W. Ahmed: *Fast Orthogonal Search for Training Radial Basis Function Neural Networks.* Diplomarbeit, The Graduate School, University of Maine, 1994.

[52] M. J. Korenberg: *A robust orthogonal algorithm for system identification and time-series analysis.* Biological Cybernetics **60**(4), 267–276, 1989.

[53] M. J. L. Orr: *Regularised centre recruitment in radial basis function networks.* Technischer Bericht 59, Centre for Cognitive Science, University of Edinburgh, Scotland, 1993.

[54] S. Chen, E. S. Chng und K. Alkadhimi: *Regularized orthogonal least squares algorithm for constructing radial basis function networks.* Int. J. Control **64**(5), 829–837, 1996.

[55] S. J. Reeves: *An Efficient Implementation of the Backward Greedy Algorithm for Sparse Signal Reconstruction.* IEEE Signal Processing Letters **6**(10), 266–268, 1999.

[56] R. A. Horn und C. R. Johnson: *Matrix Analysis.* Cambridge University Press, 1990.

[57] X. Hong, P. M. Sharkey und K. Warwick: *Automatic nonlinear predictive model-construction algorithm using forward regression and the PRESS statistic.* IEE Proc.-Control Theory Appl. **150**(3), 245–254, 2003.

[58] X. Hong, S. Chen und P. M. Sharkey: *Automatic Kernel Regression Modelling Using Combined Leave-One-Out Test Score and Regularised Orthogonal Least Squares.* International Journal of Neural Systems **14**(1), 27–37, 2004.

[59] S. Chen, X. Hong, C. J. Harris und P. M. Sharkey: *Sparse Modelling Using Orthogonal Forward Regression With PRESS Statistic and Regularization.* IEEE Trans. Systems, Man and Cybernetics, Part B **34**(2), 898–911, 2004.

[60] S. Perkins, K. Lacker und J. Theiler: *Grafting: Fast, Incremental Feature Selection by Gradient Descent in Function Space.* Journal of Machine Learning Research **3**, 1333–1356, 2003.

[61] S. Perkins und J. Theiler: *Online Feature Selection using Grafting.* In: *International Conference on Machine Learning*, Seiten 592–599. ACM Press, 2003.

[62] W. H. Press, S. A. Teukolsky, W. T. Vetterling und B. P. Flannery: *Numerical Recipes in C.* Cambridge University Press, Cambridge, New York, Port Chester, Melbourne, Sydney, 2. Auflage, 1992.

[63] J. Werner: *Numerische Mathematik*, Band 2. Vieweg, 1992.

[64] P. Lancaster und K. Šalkauskas: *Curve and Surface Fitting: An Introduction.* Academic Press, London, 1986.

[65] J. McNames: *Innovations in Local Modelling for Time Series Prediction*. Doktorarbeit, Stanford University, 1999.

[66] R. Seydel: *Practical Bifurcation and Stability Analysis*, Band 2. Springer, 1994.

[67] G. Langer und U. Parlitz: *Robust method for experimental bifurcation analysis*. Int. J. of Bifurcation and Chaos **12**(8), 1909–1913, 2002.

[68] R. Tokunaga, S. Kajiwara und T. Matsumoto: *Reconstructing bifurcation diagrams only from time-waveforms*. Physica D **79**(2–4), 348–360, 1994.

[69] I. Tokuda, S. Kajiwara, R. Tokunaga und T. Matsumoto: *Recognizing chaotic time-waveforms in terms of a parametrized family of nonlinear predictors*. Physica D **95**(3–4), 380–395, 1996.

[70] E. Bagarinao, Jr., K. Pakdaman, T. Nomura und S. Sato: *Time series-based bifurcation diagram reconstruction*. Physica D **130**(3–4), 211–231, 1999.

[71] E. Bagarinao, Jr., K. Pakdaman, T. Nomura und S. Sato: *Reconstructing bifurcation diagrams from noisy time series using nonlinear autoregressive models*. Phys. Rev. E **60**(1), 1073–1076, 1999.

[72] E. Bagarinao, Jr., T. Nomura, K. Pakdaman und S. Sato: *Generalized one-parameter bifurcation diagram reconstruction using time series*. Physica D **124**(1–3), 258–270, 1998.

[73] L. Cao: *Extraction of dynamics from non-stationary time series data*. In: J. B. Kadtke und A. Bulsara (Herausgeber): *Applied Nonlinear Dynamics and Stochastic Systems Near the Millenium*, Band 411 der Reihe *AIP Conference Proceedings*, Seiten 69–74. AIP, 1997.

[74] G. Langer und U. Parlitz: *Modeling parameter dependence from time series*. Phys. Rev. E **70**(5), 056217–1–9, 2004.

[75] K. Judd und A. Mees: *Modeling chaotic motions of a string from experimental data*. Physica D **92**(3–4), 221–236, 1996.

[76] S. Güttler, H. Kantz und E. Olbrich: *Reconstruction of the parameter spaces of dynamical systems*. Phys. Rev. E **63**(5), 056215–1–11, 2001.

[77] S. Kullback und R. A. Leibler: *On Information and Sufficiency*. The Annals of Mathematical Statistics **22**(1), 79–86, 1951.

[78] C. M. Bishop: *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, 2006.

[79] *Homepage der KDE-Toolbox*. URL: `http://www.ics.uci.edu/~ihler/code/`.

[80] C. Merkwirth: *Nächste-Nachbar basierte Methoden in der nichtlinearen Zeitreihenanalyse*. Doktorarbeit, Georg-August-Universität Göttingen, 2000.

[81] C. Merkwirth, U. Parlitz und W. Lauterborn: *Fast nearest-neighbor searching for nonlinear signal processing*. Phys. Rev. E **62**(2), 2089–2097, 2000.

Danksagung

Ich danke Herrn Prof. Dr. W. Lauterborn für die Aufnahme an das Dritte Physikalische Institut und der Ermöglichung der Finanzierung dieser Arbeit. Weiterhin danke ich ihm für die Übernahme des Amtes des Korreferenten.

Mein besonderer Dank gebührt Herrn Prof. Dr. U. Parlitz für die stets vorbildliche Betreuung und Ermöglichung dieser Arbeit.

Ich bedanke mich bei Herrn Prof. Dr. D. Ronneberger für die Aufnahme in das Graduiertenkolleg Strömungsinstabilitäten und Turbulenz.

Weiterhin bedanke ich mich bei meinen Zimmer- und Institutsgenossen David Engster, Karsten Köhler, Daniel Schanz, Philipp Koch, Sebastian Berg, Jan Schumann-Bischoff, Andreas Hüper und allen anderen namentlich nicht genannten Mitgliedern des Instituts für die angenehme Arbeitsatmosphäre und die vielen interessanten Gespräche auch abseits der Physik.

Bedanken möchte ich mich noch bei Christian Schneider und insbesondere bei meinem Bruder Michael für die aufgeopferte Zeit zum Korrekturlesen des Manuskripts.

Schließlich bedanke ich mich noch bei meinen Eltern für die Ermöglichung meines Studiums.

Lebenslauf

Name:	Jörg Dittmar
Geboren:	14. Februar 1976, Kassel
Eltern:	Werner und Heidrun Dittmar
Staatsangehörigkeit:	deutsch

1982 - 1986	Besuch der Grundschule Neuenbrunslar
1986 - 1992	Besuch der Gesamtschule Felsberg
1992 - 1995	Besuch der Jacob-Grimm-Schule, Kassel
1995	Abschluss mit der Allgemeinen Hochschulreife
1995 - 1996	Wehrdienst beim Panzergrenadierbataillon 152, Schwarzenborn
1996 - 2002	Diplomstudiengang Physik an der Georg-August-Universität Göttingen
Juni 2002	Erwerb des Hochschulgrades Diplom-Physiker
Juli 2002 - März 2003	Anstellung am Dritten Physikalischen Institut der Georg-August-Universität Göttingen als wissenschaftliche Hilfskraft
April 2003	Beginn der Promotion in Physik an der Georg-August-Universität Göttingen
Apr. 2003 - März 2005	Stipendiat des Graduiertenkollegs *Strömungsinstabilitäten und Turbulenz*
Mai 2005 - Mai 2006	Anstellung am Dritten Physikalischen Institut der Georg-August-Universität Göttingen als wissenschaftlicher Mitarbeiter

seit Nov. 2005	Anstellung am Zentrum Anaesthesiologie, Rettungs- und Intensivmedizin der Universitätsmedizin Göttingen als wissenschaftlicher Mitarbeiter